DENSO ROBOTER

Vertikal ausgerichtet

V*-D-SERIE

Horizontal ausgerichtet

H*-D-SERIE

Kartesische Koordinate

XYC-4D-SERIE

Bildverarbeitungsgerät

μVision-21-SERIE

PROGRAMMIERUNGSHANDBUCH

(Ver. 1.4)

Copyright © DENSO, 1999

Alle Rechte vorbehalten. Kein Teil dieses Handbuches darf in irgendeiner Form oder auf irgendeine Weise ohne die schriftliche Genehmigung des Herausgebers reproduziert werden.

Änderungen der Spezifikationen ohne vorherige Ankündigung vorbehalten.

Alle erwähnten Produkt- und Firmennamen sind Warenzeichen oder eingetragene Warenzeichen ihrer entsprechenden Inhaber.

Vorwort

Vielen Dank für den Kauf dieses Hochgeschwindigkeits- und Hochpräzisions-Montageroboters.

Bevor Sie den Roboter in Betrieb nehmen, lesen Sie bitte sorgfältig dieses Handbuch durch, damit Sie aus dem Roboter den größtmöglichen Nutzen für Ihre Montagearbeiten erhalten.

Folgende Roboterserien und/oder -modelle werden in diesem Handbuch behandelt

- Vertikal ausgerichteter Roboter V*-D-Serie
- Horizontal ausgerichteter Roboter H*-D-Serie
- Kartesischer Koordinaten-Roboter XYC-4D-Serie
- Bildverarbeitungsgerät μVision-21-Serie

Wichtig

Zur Gewährleistung der Sicherheit muß das Bedienpersonal mit den Vorsichtsmaßnahmen und Anleitungen im Abschnitt "SICHERHEITS- UND VORSICHTSMASSNAHMEN" (Seite 1 bis 9) vertraut sein.

Der Dokumentationssatz ist wie folgt aufgebaut

Der Dokumentationssatz umfaßt die folgenden sechs Bücher. Wenn Sie mit dieser Roboterserie nicht vertraut sind, sollten Sie alle diese Bücher gelesen und verstanden haben, bevor Sie mit dem Roboter arbeiten.

EINFÜHRUNGSHANDUCH

Einführung in den DENSO Roboter. Anhand eines Gerätebeispiels führt Sie dieses Buch durch die Bedienung Ihres Roboter mit dem Programmiergerät, das Erstellen eines Programmes in WINCAPSII und die automatische Bedienung des Roboters.

INSTALLATIONS- UND WARTUNGSHANDBUCH

Beschreibt den Roboteraufbau, gibt die Anleitungen für die Installation der Roboterelemente sowie die Wartungs- und Prüfungsarbeiten.

EINSTELLUNGSHANDBUCH - dieses Buch -

Beschreibt das Einstellen oder Programmieren des Roboters mit Hilfe des Programmiergeräts oder des Bedienfeldes.

WINCAPSII

Enthält Anleitungen zum Einsatz des auf dem PC installierten und mit dem Roboter und dem Roboter-Controller verbundenen Programmiersystems für die Programmentwicklung und -verwaltung.

PROGRAMMIERUNGSHANDBUCH - dieses Buch -

Beschreibt die Programmiersprache PAC, die Schritte der Programmentwicklung in PAC und Befehlsspezifikationen.

FEHLERCODETABELLEN

Enthält eine Auflistung der Fehlercodes, die am Programmiergerät, am Bedienfeld oder auf dem PC-Bildschirm angezeigt werden, wenn ein Fehler an einem Roboter oder in WINCAPSII auftritt. Enthält außerdem eine Zusammenfassung über die Fehlerbeschreibung und die auszuführenden Maßnahmen.

Das vorliegende Handbuch ist wie folgt aufgebaut:

Dieses Handbuch ist nur ein Teil des Dokumentationssatzes. Es enthält die SICHERHEITS- UND VORSCIHTSMASSNAHMEN sowie die Kapitel 1 bis 5.

SICHERHEITS- UND VORSICHTSMASSNAHMEN

Erläutert sicherheitsrelevante Begriffe und Symbole und enthält zu beachtende Vorsichtsmaßnahmen. Bitte lesen Sie diesen Abschnitt unbedingt, bevor Sie anfangen, mit dem Roboter zu arbeiten.

TEIL 1 PROGRAMMKONZEPTION

Kapitel 1 Musterprogramm

Dieses Kapitel verdeutlicht die Verwendung der einzelnen Befehle anhand eines einfachen Musterprogrammes.

Kapitel 2 Programmablauf

Dieses Kapitel beschreibt die Regeln zum Erstellen eines Programmes, damit dieses Programm in der PAC-Sprache läuft.

Kapitel 3 Roboterbewegung

Dieses Kapitel erläutert die verschiedenen Bewegungen des Roboters. Die Roboterbewegung ist je nach Bezugsstellung und der zum Erreichen des Zielpunktes gewählten Entscheidungsmethode unterschiedlich.

Kapitel 4 Bestimmen von Geschwindigkeit, Beschleunigung und Verzögerung

Dieses Kapitel erläutert die Bedeutung und die Einstellungen von Geschwindigkeit, Beschleunigung und Verzögerung.

Kapitel 5 Bildverarbeitung

Dieses Kapitel erläutert die auf die Bildverarbeitung bezogenen Begriffe, die bei der Programmerstellung benötigt werden.

TEIL 2 BEFEHLSREFERENZ

Kapitel 6 Anleitung zur Befehlsreferenz

Dieses Kapitel enthält Befehlsbeschreibungen und eine Befehlsliste für den PAC-Roboter.
Verwenden Sie die Befehlsliste, um Informationen zu den einzelnen Befehlen schnell zu finden.

Kapitel 7 Konfigurationselemente der PAC-Sprache

Dieses Kapitel erläutert die Elementregeln (Kennung, Variable, Konstante, Bediener, Ausdruck und Befehl), aus denen sich die PAC-Sprache aufbaut.

Kapitel 8 Syntax der PAC-Sprache

Dieses Kapitel erläutert die Syntaxregeln, die beim Schreiben eines Programmes in der PAC-Sprache zu beachten sind.

Kapitel 9 bis 21

Diese Kapitel erläutern die einzelnen Befehle der PAC-Robotersprache. Die Befehle sind nach Funktion geordnet.

Benutzen Sie die Befehlsliste aus KAPITEL 1, um schnell die Erläuterung eines bestimmten Befehls zu finden.

Kapitel 22 PAC-Bibliothek

Dieses Kapitel erläutert die als Standardzubehör an WINCAPSII angehängten Programmbibliotheken.

Anhänge

Zeichencodetabelle

Abbildungen von Schulter, Ellbogen und Handgelenk

Umgebungseinstellwerte

Verwenden von Zustandsparametern

Liste der reservierten Worte

Befehlsentsprechungstabelle zu herkömmlichen Sprachbefehlen (VS)

Kompatibilitätstabelle

Parametereinstelltabelle

Index

SICHERHEITS- UND VORSICHTSMASSNAHMEN

Bitte stellen Sie die Einhaltung der folgenden Sicherheits- und Vorsichtsmaßnahmen sicher.

Die strikte Einhaltung der Warn- und Vorsichtshinweise ist ein MUSS zur Verhütung von Unfällen, die zu körperlichen Schäden und erheblichem Sachschaden führen können. Stellen Sie sicher, daß Sie die unten aufgeführten Definitionen dieser Begriffe und die dazugehörigen, weiter unten dargestellten Symbole verstehen, bevor Sie mit dem Text fortfahren.

⚠ **WARNUNG**	Warnung vor Situationen, die zu schweren körperlichen Schäden oder Tod führen können, wenn die Anleitungen nicht korrekt befolgt werden.
⚠ **VORSICHT**	Warnung vor Situationen, die zu geringerem körperlichen Schaden oder erheblichem Sachschaden führen können, wenn die Anleitungen nicht korrekt befolgt werden.

Terminologie und Definitionen

Maximaler Roboterraum: Bezieht sich auf die Ausdehnung des Raums, der von den maximal möglichen Bewegungen aller Roboterbauteile, einschließlich Endwerkzeug, Werkstück und Anbaueinheiten, eingeschlossen wird. (Zitiert nach "RIA* Committee Draft".)

Sicherheitsraum: Bezieht sich auf den Teil des maximalen Roboterraums, auf den ein Roboter durch Begrenzungsvorrichtungen (z. B. mechanische Anschläge) eingeschränkt wird. Der maximale Weg, den der Roboter, das Endwerkzeug und das Werkstück nach dem Ansprechen der Begrenzungsvorrichtung zurücklegen können, definiert die Grenzen des Sicherheitsraums des Roboters. (Zitiert nach "RIA* Committee Draft".)

Bewegungsraum: Der Teil des Sicherheitsraums, auf den der Roboter durch Software-Bewegungseinschränkungen begrenzt ist. Der maximale Weg, den der Roboter, das Endwerkzeug und das Werkstück nach dem Einstellen der Software-Bewegungsgrenzen zurücklegen können, definiert die Grenzen des Bewegungsraums des Roboters. (Der Begriff "Bewegungsraum" ist eine spezifische Terminologie von Denso.)

Arbeitsraum: Bezieht sich auf den Teil des Sicherheitsraums (oder Bewegungsraums in der Terminologie von Denso), der vom Roboter während des Ausführens seines Aufgabenprogrammes tatsächlich genutzt wird. (Zitiert nach "RIA* Committee Draft".)

Aufgabenprogramm: Bezieht sich auf die Instruktionen für die Bewegung und die Hilfsfunktionen, die die jeweilige Aufgabe des Robotersystems definieren. (Zitiert nach "RIA* Committee Draft".)

(*RIA: Robotic Industries Association/Vereinigung der Roboterindustrie)

1. Einführung

In diesem Abschnitt sind die Sicherheits- und Vorsichtsmaßnahmen beschrieben, die während der Installation, des Programmierens, der Prüfung, der Anpassung und der Wartung des Roboters einzuhalten sind.

2. Vorsichtsmaßnahmen bei der Installation

2.1 Sicherstellen der geeigneten Installationsumgebung

2.1.1 Standardausführung

Die Standardausführung ist nicht explosionsgeschützt, staubdicht oder spritzwassergeschützt ausgelegt. Demzufolge darf sie nicht in Umgebungen mit:

(1) entzündlichen Gasen oder Flüssigkeiten,
(2) herumfliegenden Spänen der Metallbearbeitung oder sonstigen elektrisch leitendendem Material,
(3) säurehaltigen, alkalihaltigen oder sonstigen korrosiven Gasen,
(4) Ölnebeln vom Schneiden oder Schleifen,
(5) der Gefahr des Eintauchens in Flüssigkeiten,
(6) schwefelhaltigen Ölnebeln vom Schneiden oder Schleifen oder
(7) großen Umrichtern, Transmittern hoher Ausgangsleistung und hoher Frequenz, großen Schützen, Schweißgeräten oder sonstigen Quellen elektrischer Störungen installiert werden.

2.1.2 Staubdichte, spritzwassergeschützte Ausführung

Die staubdichte, spritzwassergeschützte Ausführung ist eine Struktur entsprechend IP54, die gegen Staub und Spritzwasser geschützt ist, nicht jedoch gegen Explosionen. (Das Handgelenk des VM-D-W entspricht einer gegen Staub und Spritzwasser geschützten IP65-Struktur.)

Beachten Sie, daß der Roboter-Controller keine staubdichte oder spritzwassergeschützte Roboterkomponente ist. Aus diesem Grund ist der Roboter-Controller in einer durch Ölnebel belasteten Umgebung in einem optionalen Schutzgehäuse unterzubringen.

Die staubdichte, spritzwassergeschützte Ausführung darf nicht in einer Umgebung mit:

(1) entzündlichen Gasen oder Flüssigkeiten,
(2) säurehaltigen, alkalihaltigen oder sonstigen korrosiven Gasen,
(3) großen Umrichtern, Transmittern hoher Ausgangsleistung und hoher Frequenz, großen Schützen, Schweißgeräten oder sonstigen Quellen elektrischer Störungen,
(4) der Gefahr des Eintauchens in Flüssigkeiten,
(5) Partikeln oder Spänen vom Schleifen oder Zerspanen,
(6) der Verwendung eines von DENSO nicht zugelassenen Öls oder
Anmerkung: DENSO-zugelassenes Öl: Yushiron-Öl Nr. 4C (unlöslich)
(7) schwefelhaltigen Ölnebeln vom Schneiden oder Schleifen installiert werden.

2.2 Servicebereich

Der Roboter und die Peripheriegeräte sind so zu installieren, daß ein ausreichender Servicebereich für das sichere Programmieren, die Wartung und die Prüfung zu Verfügung steht.

2.3 Steuergeräte außerhalb des Sicherheitsraums des Roboters

Der Roboter-Controller, das Programmiergerät und das Bedienfeld sollten außerhalb des Sicherheitsraums des Roboters installiert werden und an einer Stelle, von der aus Sie alle Bewegungen des Roboters beobachten können, wenn Sie mit dem Roboter-Controller, dem Programmiergerät oder dem Bedienfeld arbeiten.

2.4 Anordnung von Meß- und Anzeigegeräten

Druckmesser, Öldruckmesser andere Anzeiger sind an einem leicht zu kontrollierenden Ort zu montieren.

2.5 Schutz der Elektro-installation und der Hydraulik-/Druckluft-Rohrleitungen

Die Elektroinstallationen und Hydraulik-/Druckluft-Rohrleitungen sind durch eine Abdeckung oder eine entsprechende Maßnahme zu schützen, wenn die Möglichkeit einer Beschädigung besteht.

2.6 Anordnung der NOT-AUS-Schalter

Die NOT-AUS-Schalter sind an einer leicht erreichbaren Stelle anzubringen, um den Roboter ggf. unverzüglich anhalten zu können.

(1) Die NOT-AUS-Schalter müssen rot sein.

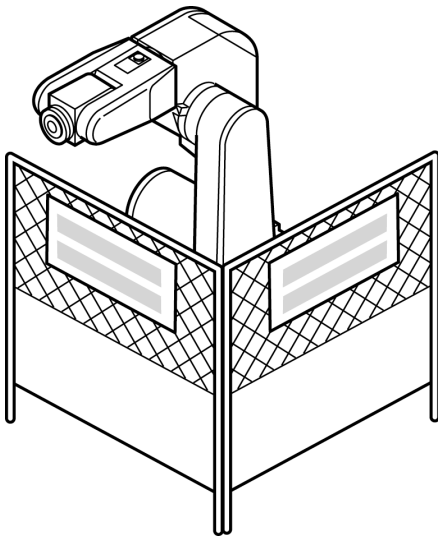
(2) Die NOT-AUS-Schalter müssen so konstruiert sein, daß sie nach dem Auslösen weder automatisch noch versehentlich durch eine Bedienperson freigegeben werden können.

(3) Die NOT-AUS-Schalter müssen vom Netzschalter klar zu unterscheiden sein.

2.7 Anordnung der Betriebszustandsan-zeigen

Die Betriebszustandsanzeigen sind so anzuordnen, daß das Bedienpersonal leicht erkennt, ob der Roboter vorübergehend, durch einen NOT-AUS oder durch eine Fehlfunktion stillsteht.

2.8 Aufstellen der Sicherheitsumzäunung oder -verkleidung



Eine Sicherheitsumzäunung oder -verkleidung ist so aufzustellen, daß niemand den Sicherheitsraum des Roboters betreten kann. Sollte dies nicht möglich sein, verwenden Sie andere Schutzeinrichtungen wie in Abschnitt 2.9 beschrieben.

(1) Die Umzäunung oder Verkleidung muß so aufgebaut sein, daß man sie nicht ohne weiteres bewegen oder entfernen kann.

(2) Die Umzäunung oder Verkleidung muß so aufgebaut sein, daß man sie durch äußere Krafteinwirkung nicht ohne weiteres beschädigen oder verformen kann.

(3) In der Umzäunung oder Verkleidung ist ein Ein-/Ausgang einzurichten. Errichten Sie die Umzäunung oder Verkleidung so, daß niemand sie ohne weiteres durch Klettern über die Umzäunung oder Verkleidung überwinden kann.

(4) Die Umzäunung oder Verkleidung ist so zu errichten, daß man Hände oder andere Körperteile nicht hindurchstecken kann.

(5) Ergreifen Sie eine der folgenden Schutzmaßnahmen für den Ein-/Ausgang der Umzäunung oder Verkleidung:

1) Bringen Sie eine Tür, ein Seil oder eine Kette mitten im Ein-/Ausgang der Umzäunung oder Verkleidung an. Montieren Sie eine Sperre, die das automatische Ansprechen des NOT-AUS-Gerätes sicherstellt, wenn die Sperre geöffnet oder entfernt wird.

2) Bringen Sie einen Warnhinweis am Ein-/Ausgang der Umzäunung oder Verkleidung mit der Aufschrift "In Betrieb! Zutritt verboten!" oder "Es wird gearbeitet! Nicht bedienen!" an, und sorgen Sie dafür, daß diese Anweisungen vom Personal zu jeder Zeit befolgt werden.

Positionieren Sie bei Durchführung eines Testlaufs vor dem Einrichten der Umzäunung oder Verkleidung einen Beobachter an einer Stelle außerhalb des Sicherheitsraums des Roboters, von wo aus er alle Bewegungen des Roboters beobachten kann. Die Aufsichtsperson soll verhindern, daß Personal den Sicherheitsraum des Roboters betritt, und soll sich ausschließlich dieser Aufgabe widmen.

2.9 Anbringen eines Seils oder einer Kette

Falls keine Umzäumung oder Verkleidung wie in Abschnitt 2.8 beschrieben angebracht werden kann, hängen Sie ein Seil oder eine Kette um den Sicherheitsraum des Roboters, um sicherzustellen, daß niemand den Sicherheitsraum des Roboters betritt.

(1) Sorgen Sie dafür, daß die Stützpfosten nicht leicht verschoben werden können.

(2) Stellen Sie sicher, daß sich das Seil bzw. die Kette farblich deutlich von der Umgebung abhebt.

(3) Bringen Sie an einer deutlich sichtbaren Stelle einen Warnhinweis mit der Aufschrift "In Betrieb! Zutritt verboten!" oder "Es wird gearbeitet! Nicht bedienen!" an, und sorgen Sie dafür, daß diese Anweisungen vom Personal zu jeder Zeit befolgt werden.

(4) Legen Sie den Ein-/Ausgang fest und befolgen Sie die Anweisungen (3) bis (5) von Abschnitt 2.8.

2.10 Einstellen des Bewegungsraums des Roboters

Das für den Betrieb des Roboters erforderliche Volumen wird als Arbeitsraum bezeichnet.

Wenn der Bewegungsraum des Roboters größer ist als der Bedienungsraum, sollten Sie einen kleineren Bewegungsraum definieren, um zu verhindern, daß der Roboter andere Geräten oder Komponenten berührt und eventuell beschädigt.

Siehe hierzu auch die Anweisungen in Kapitel 4 des "INSTALLATIONS- UND WARTUNGSHANDBUCHES".

2.11 Änderungen am Roboter sind nicht erlaubt

Verändern Sie niemals den Roboteraufbau, den Roboter-Controller, das Programmiergerät oder andere Geräte.

2.12 Reinigen der Werkzeuge

Wenn der Roboter mit Schweißpistolen, Farbspritzdüsen oder anderen Endwerkzeugen arbeitet, die gereinigt werden müssen, wird die automatische Durchführung des Reinigungsvorganges empfohlen.

2.13 Beleuchtung

Für den sicheren Roboterbetrieb muß ausreichende Beleuchtung vorhanden sein.

2.14 Schutz vor Gegenständen, die vom Endwerkzeug weggeschleudert werden

Bei Verletzungsgefahr des Personals durch vom Endwerkzeug fallengelassene oder weggeschleuderte Objekte sind die Abmessungen, das Gewicht, die Temperatur und die chemischen Eigenschaften der Objekte zu berücksichtigen und geeignete Sicherheitsvorkehrungen zu treffen, um die Sicherheit des Personals zu garantieren.

2.15 Anbringen des Warnschildes

Bringen Sie das mit dem Roboter mitgelieferte Warnschild am Ein-/Ausgang der Sicherheitsumzäunung oder an einer anderen gut sichtbaren Stelle an.

3. Vorsichtsmaßnahmen, während der Roboter aktiv ist

⚠ Warnung

Das Berühren des Roboters während seines Betriebs kann zu schweren Verletzungen führen. Stellen Sie sicher, daß die folgenden Vorschriften eingehalten und die in Abschnitt 3.1 aufgezählten Vorsichtsmaßnahmen bei der Durchführung aller Arbeiten befolgt werden.

1) Betreten Sie den Sicherheitsraum des Roboters nicht, wenn der Roboter in Betrieb oder der Antriebsmotor eingeschaltet ist.
2) Als Vorsichtsmaßnahme gegen Fehlfunktionen sorgen Sie dafür, daß das NOT-AUS-Gerät so aktiviert ist, daß die Spannung des Roboterantriebs beim Betreten des Sicherheitsraums des Roboters abgeschaltet wird.
3) Wenn der Sicherheitsraum des Roboters betreten werden muß, um eine Programmierung oder Wartung durchzuführen, während der Roboter aktiv ist, stellen Sie sicher, daß die in Abschnitt 3.3 "Gewährleistung der Sicherheit des Personals bei der Durchführung von Arbeiten im Sicherheitsraum des Roboters" beschriebenen Maßnahmen befolgt werden.

3.1 Erstellen von Arbeitsrichtlinien und Sicherstellen der Einhaltung durch das Personal

Falls der Sicherheitsraum des Roboters zur Programmierung oder Wartung des Roboters betreten werden muß, definieren Sie "Arbeitsrichtlinien" zu folgenden Punkten, und stellen Sie sicher, daß die Mitarbeiter sich an diese Richtlinien halten.

(1) Notwendige Bedienungsabläufe für den Roboterbetrieb.
(2) Robotergeschwindigkeit während des Lernvorganges.
(3) Verwendete Signalisierungsverfahren, wenn die Arbeiten von mehr als einem Arbeiter durchgeführt werden.
(4) Vorgehensweise des Personals im Störungsfall je nach Art der Störung.
(5) Notwendige Schritte zur Überprüfung der Freigabe und der Sicherheit während des Störungszustandes, um den Roboter nach dem Anhalten der Roboterbewegung infolge der Betätigung des NOT-AUS-Gerätes neu zu starten.
(6) Nachfolgende Schritte sind unabhängig von den vorher genannten Punkten notwendig, um Gefahren infolge unerwarteter Roboterbewegung oder Störung des Roboters zu verhindern.
   1) Anzeige der Steuertafel (siehe Abschnitt 3.2 auf der nächsten Seite)
   2) Gewährleistung der Sicherheit des Personals, das Arbeiten im Sicherheitsraum des Roboters durchführt (siehe Abschnitt 3.3 auf der nächsten Seite)
   3) Die vom Personal einzunehmende Position und Haltung

      Die Position und Haltung, die dem Personal ermöglicht, den normalen Roboterbetrieb zu bestätigen und bei Auftreten einer Störung sofort Schutz zu suchen.

4) Realisierung von Maßnahmen für den Lärmschutz
5) Signalisierungsverfahren für Personal an angeschlossenen Geräten
6) Arten der Störungen und wie diese zu unterscheiden sind

Stellen Sie sicher, daß die definierten "Arbeitsrichtlinien" für den Robotertyp, den Standort der Installation und die Art der Arbeit angemessen sind.

Achten Sie darauf, daß beim Aufstellen dieser Arbeitsrichtlinien auch die betroffenen Mitarbeiter, Techniker, die Hersteller der Geräte und der Beauftragte für die Arbeitssicherheit gehört werden.

3.2 Anzeige des Bedienfeldes

Erstellen Sie eine Anzeige, die auf dem Bediengerät oder Programmiergerät erscheint, um darauf hinzuweisen, daß der Roboter in Betrieb ist und um zu verhindern, daß der EIN-Schalter oder Umschalter während des Betriebs durch andere Personen als durch den Bediener zufällig betätigt werden. Ergreifen Sie zusätzliche geeignete Maßnahmen wie das Verriegeln der Maschinenverkleidung.

3.3 Gewährleistung der Sicherheit des Personals bei der Durchführung von Arbeiten im Sicherheitsraum des Roboters

Beachten Sie bei der Durchführung von Arbeiten innerhalb des Sicherheitsraums des Roboters die folgenden Schritte, um sicherzustellen, daß der Betrieb des Roboters bei einer Fehlfunktion sofort gestoppt werden kann.

(1) Positionieren Sie einen Beobachter an einer Stelle außerhalb des Sicherheitsraums des Roboters, von wo aus er alle Bewegungen des Roboters beobachten kann. Der Beobachter darf nicht gleichzeitig mit anderen Aufgaben beschäftigt sein.

   ① Bei einer Störung ist sofort ein NOT-AUS-Gerät zu aktivieren.

   ② Achten Sie darauf, daß ausschließlich Mitarbeiter, die für diese Aufgabe ausgewählt wurden, den Sicherheitsraum des Roboters betreten.

(2) Stellen Sie sicher, daß ein Fachmann innerhalb des Sicherheitsraums des Roboters einen tragbaren NOT-AUS-Schalter (Schaltfläche "Roboterstop" auf dem Programmiergerät) bei sich hat, den er im Notfall sofort betätigen kann.

3.4 Prüfungen vor Beginn von Arbeiten wie z. B. das Programmieren

Vor dem Beginn von Arbeiten wie die Programmierung sind evtl. Reparaturen sofort nach dem Erkennen einer Störung und sonstige notwendige Maßnahmen durchzuführen.

(1) Überprüfen Sie die Umhüllung oder Abdeckung der externen Verdrahtung oder der externen Geräte auf evtl. Beschädigung.

(2) Überprüfen Sie die normale Funktionsweise des Roboters (außergewöhnlicher Lärm oder Vibrationen während des Betriebs).

(3) Überprüfen Sie die Funktionstüchtigkeit der NOT-AUS-Gerätes.

(4) Überprüfen Sie die Druckluft oder Öl führenden Leitungen auf Lecks.

(5) Stellen Sie sicher, daß der Sicherheitsraum des Roboters nicht durch Hindernisse verstellt ist.

3.5 Entlüftung

Vor der Demontage oder dem Auswechseln von Pneumatik-Bauteilen ist der Antriebszylinder zu entlüften.

3.6 Vorsichtsmaßnahmen für Testläufe

Während der Durchführung von Testläufen sollte sich das Personal nach Möglichkeit außerhalb des Sicherheitsraums des Roboters aufhalten.

3.7 Vorsichtsmaßnahmen für den Automatikbetrieb

(1) Bei Inbetriebnahme

Vor der Inbetriebnahme des Roboters sind zuerst die nachstehenden Punkte zu überprüfen. Außerdem müssen die zu benutzenden Signale festgelegt und die Signalisierung mit dem gesamten betroffenen Personal überprüft werden.

1) Stellen Sie sicher, daß sich niemand im Sicherheitsraum des Roboters aufhält.

2) Überprüfen Sie, daß das Programmiergerät und die Werkzeuge an ihren vorgesehenen Plätzen sind.

3) Überprüfen Sie, ob Störmeldeleuchten am Roboter oder dazugehörigen Ausrüstungen leuchten.

(2) Überprüfen Sie, ob die Anzeigelampe für automatischen Betrieb im Automatikbetrieb leuchtet.

(3) Maßnahmen im Störungsfall

Wenn eine Störung am Roboter oder den zugehörigen Geräten auftritt und der Sicherheitsraum des Roboters für eine Notfallwartung betreten werden muß, stoppen Sie den Betrieb des Roboters durch Aktivieren des NOT-AUS-Geräts. Ergreifen Sie alle notwendigen Maßnahmen wie z. B. das Anbringen eines Warnschildes am Start-Schalter, um auf die laufenden Arbeiten hinzuweisen und zu verhindern, daß jemand Zugang zum Roboter bekommt.

3.8 Vorsichtsmaßnahmen bei Reparaturen

(1) Reparaturen dürfen nicht außerhalb des dafür vorgesehenen Bereiches durchgeführt werden.

(2) Unter keinen Umständen dürfen Sperrvorrichtungen entfernt werden.

(3) Beim Öffnen der Abdeckung des Roboter-Controller zum Auswechseln der Batterie oder aus anderen Gründen ist die Stromversorgung des Roboter-Controller abzuschalten und das Stromkabel zu ziehen.

(4) Verwenden Sie nur von DENSO zugelassene Ersatzwerkzeuge.

4. Tägliche und regelmäßige Prüfungen

(1) Stellen Sie sicher, daß die täglichen und regelmäßigen Prüfungen durchgeführt werden. Überprüfen Sie immer vor Arbeitsbeginn, daß keine Probleme mit dem Roboter und der dazugehörigen Ausrüstungen vorliegen. Bei Problemen sind alle notwendigen Maßnahmen zu ergreifen, um diese zu beheben.

(2) Die Durchführung regelmäßiger Prüfungen oder evtl. Reparaturen sind zu protokollieren. Die Protokolle sind mindestens 3 Jahre aufzubewahren.

5. Verwaltung von Disketten

(1) Gehen Sie mit den zusammen mit dem Roboter gelieferten Disketten sorgfältig um und bewahren Sie sie an einem sicheren Ort auf. Auf diesen Disketten sind die Anfangseinstellungen sowie speziell für Ihren Roboter vorbereitete Einstellungen gespeichert.

(2) Nach beendetem Programmiervorgang oder etwaigen Änderungen sind die Programme und die Daten stets auf Disketten zu sichern.

Datensicherungen helfen Ihnen, die im Roboter-Controller gespeicherten Daten wiederherzustellen, wenn diese aufgrund einer leeren Backup-Batterie verlorengegangen sind.

(3) Beschriften Sie alle für das Speichern von Aufgabenprogrammen verwendete Disketten, um das Laden falscher Disketten im Roboter-Controller auszuschließen.

(4) Bewahren Sie die Disketten an einem von Staub, Luftfeuchtigkeit und Magnetfeldern freien Ort auf. Diese Einflüsse könnten die Disketten oder die auf den Disketten gespeicherten Daten zerstören.

INHALT

Vorwort ..... **i**

Der Dokumentationssatz ist wie folgt aufgebaut ..... **ii**

Das vorliegende Handbuch ist wie folgt aufgebaut: ..... **iii**

SICHERHEITS- UND VORSICHTSMASSNAHMEN **1**

Liste mit Anweisungen in alphabetischer Reihenfolge (Nach dem Inhaltsverzeichnis.)

Liste mit Anweisungen in der Reihenfolge der Funktionen
(Nach der Liste mit Anweisungen in alphabetischer Reihenfolge.)

Kapitel 1 Musterprogramm

1.1 Modellfall-Anwendung 1-1
1.2 Programmablauf 1-2
1.3 Programmliste 1-3

Kapitel 2 Programmablauf

2.1 Aufrufen eines Programmes und eines Unterprogrammes 2-1
2.1.1 Aufrufen einer Unterroutine 2-2
2.1.2 Aufrufen eines Programmes 2-3
2.1.3 Rekursiver Programmaufruf 2-4
2.2 Ausführen eines Programmes 2-5
2.2.1 Ausführen über das Programmiergerät 2-5
2.2.2 Ausführen über das Bedienfeld 2-5
2.2.3 Ausführen über ein externes Gerät 2-5
2.3 Multitasking 2-6
2.3.1 Priorität 2-6
2.3.2 Kommunikation zwischen Tasks 2-6
2.4 Serielle Kommunikation 2-8
2.4.1 Schaltkreisnummer 2-8
2.4.2 Kommunikationsbefehl 2-8
2.4.3 Bereinigen des Kommunikationspuffers 2-8
2.4.4 Musteranwendung 2-9
2.5 Bibliothek 2-13
2.5.1 Programmbank 2-13
2.5.2 Palettierungsbibliothek 2-14

Kapitel 3 Roboterbewegung

3.1 Absolute Bewegung und relative Bewegung 3-1
3.1.1 Absolute Bewegung 3-1
3.1.2 Relative Bewegung 3-1
3.1.3 Beispiele für die absolute Bewegung und die relative Bewegung 3-1
3.2 Bestätigen der erreichten Position 3-3
3.2.1 Durchlaufbewegung 3-3
3.2.2 Endebewegung 3-3
3.2.3 Codiererwert-Prüfbewegung 3-3
3.2.4 Bewegungsbeispiele zu Durchlauf, Ende und Codiererwertprüfung 3-4
3.2.5 Unterschiede der Ausführungszeit bei Durchlauf-, End- und Codiererwert-Prüfbewegung 3-5
3.2.6 Wenn Durchlaufbewegung nicht ausgeführt wird 3-6
3.2.7 Wenn der Effekt der Durchlaufbewegung reduziert ist 3-7
3.2.8 Wenn die Beschleunigung den Durchlaufbewegungspfad beeinflußt 3-8
3.2.9 Durchlaufstartverschiebung 3-9

3.3 Interpolationssteuerung 3-11
3.3.1 Punkt-zu-Punkt-Steuerung 3-11
3.3.2 Linearsteuerung 3-12
3.3.3 Bogeninterpolationssteuerung 3-12
3.4 Wenn Ausgabebefehl nach der Bewegungsanweisung vorhanden ist 3-13
3.5 Nachgiebigkeits-Steuerungsfunktion 3-14
3.5.1 Überblick 3-14
3.5.2 Stromgrenzwertfunktion für einzelne Achsen [V1.2 oder neuer] 3-14
3.5.3 Steuerfunktion für die Spitzen-Nachgiebigkeit [V1.4 oder neuer] 3-16

Kapitel 4 Bestimmen von Geschwindigkeit, Beschleunigung und Verzögerung

4.1 Externe Geschwindigkeit und interne Geschwindigkeit 4-1
4.2 Angabe der Geschwindigkeit 4-1
4.3 Externe Beschleunigung, Externe Verzögerung, Interne Beschleunigung und Interne Verzögerung 4-1
4.4 Festlegen der Beschleunigung und Verzögerung 4-2
4.5 Beispiel zum Einstellen der Geschwindigkeit und Beschleunigung 4-2
4.6 Steuersätze der Bewegungsoptimierung 4-5
4.6.1 Steuersatz 0 4-5
4.6.2 Steuersatz 1 4-8
4.6.3 Steuersatz 2 4-9
4.6.4 Steuersatz 3 4-10
4.6.5 Verwenden von Bedingungen, die Sie setzen müssen 4-10
4.6.6 Hinweise zum Einstellen 4-11
4.7 Einstellen der Master-Steuerparameter in den Benutzereinstellungen 4-12
4.7.1 Einstellen der Master-Steuerparameter für die Masse der Nutzlast, den Schwerpunkt und den Steuersatz der Bewegungsoptimierung 4-12
4.7.2 Einstellen der internen Lastbedingungswerte (Masse de Nutzlast und Schwerpunkt) und Interner Modus 4-16
4.7.3 Einstellen der Roboterinstallationsbedingung 4-17
4.7.4 Einstellen der Initialisierung der optimalen Lastkapazität [V1.4 oder neuer] 4-20
4.8 Sicherheitsfunktionen 4-23
4.8.1 ndTc (Anweisung) [V1.2 oder neuer] 4-23
4.8.2 ndTs (Anweisung) [V1.2 oder euer] 4-23

Kapitel 5 Bildverarbeitungssteuerung

5.1 Bildverarbeitungssteuerung 5-1
5.1.1 Begriffe der Bildverarbeitungssteuerung 5-1

Kapitel 6 Anleitung zur Befehlsreferenz

6.1 Erläuterungen zum Format der Befehlsbeschreibungen 6-1
6.2 Befehlsliste 6-2
6.2.1 Befehle in alphabetischer Reihenfolge 6-2
6.2.2 Befehle nach Funktionen geordnet 6-2

Kapitel 7 Konfigurations- elemente der PAC-Sprache

7.1 Die neue Robotersprache PAC 7-1
7.2 Beziehung zwischen der PAC-Robotersprache und konventionellen Sprachen 7-2
7.3 Sprachelemente 7-3
7.4 Name 7-3
7.5 Kennung 7-4
7.5.1 Variable 7-4
7.5.2 Funktion 7-8
7.5.3 Kennung 7-8
7.5.4 Programm 7-9

7.6 Datentyp 7-10
7.6.1 Zeichenfolge 7-10
7.6.2 Numerischer Wert 7-10
7.6.3 Vektor 7-10
7.6.4 Stellung 7-10
7.6.5 E/A (EIN/AUS) 7-10
7.7 Datentypumwandlung 7-11
7.7.1 Numerischer Wert 7-11
7.7.2 Zeichen und Numerischer Wert 7-11
7.7.3 Stellungstyp-Daten 7-11
7.8 Konstante 7-12
7.8.1 Konstante vom Typ Numerischer Wert 7-12
7.8.2 Konstante vom Typ Zeichenfolge 7-14
7.8.3 Konstante vom Typ Vektor 7-14
7.8.4 Konstante vom Typ Stellung 7-14
7.9 Ausdruck und Operator 7-16
7.9.1 Zuweisungs-Operator 7-16
7.9.2 Arithmetischer Operator 7-16
7.9.3 Relationaler Operator 7-17
7.9.4 Logischer Operator 7-18
7.9.5 Zeichenfolge-Operator 7-18
7.9.6 Vektor-Operation 7-19
7.9.7 Positions-Operation 7-19
7.9.8 Gelenk-Operation 7-20
7.9.9 Homogene Transformations-Array-Operation 7-20
7.10 Einheiten für die PAC-Sprache 7-21

Kapitel 8 Syntax der PAC-Sprache

8.1 Anweisung und Zeile 8-1
8.2 Programmname und Deklaration 8-2
8.3 Kennung 8-3
8.4 Zeichensatz 8-4
8.5 Reserviertes Wort 8-5
8.6 Deklarationsanweisung 8-6
8.6.1 Typdeklaration 8-6
8.6.2 Funktions-/Programmdeklaration 8-9
8.6.3 Benutzerkoordinatensystem-Deklaration 8-9
8.7 Zuweisungsanweisung 8-10
8.7.1 Zuweisungsanweisung für numerischen Wert 8-10
8.7.2 Zuweisungsanweisung für Zeichenfolge 8-10
8.7.3 Zuweisungsanweisung für Vektor 8-10
8.7.4 Zuweisungsanweisung für Stellung 8-11
8.8 Flußsteuerungsanweisung 8-13
8.8.1 Unbedingte Verzweigung 8-13
8.8.2 Bedingte Verzweigung 8-13
8.8.3 Auswahl 8-13
8.8.4 Wiederholung 8-14
8.8.5 Aufruf eines definierten Prozesses 8-15
8.9 Robotersteuerungsanweisung 8-17
8.9.1 Steuerungsanweisung für Bewegung 8-17
8.9.2 Steuerungsanweisung für Stop 8-17
8.9.3 Steuerungsanweisung für Geschwindigkeit 8-18
8.9.4 Steuerungsanweisung für Zeit 8-18
8.9.5 Anweisung für Koordinatentransformation 8-18

8.10 Steuerungsanweisung für die Ein/Ausgabe ... 8-19
8.10.1 DI/DO-Steuerungsanweisung ... 8-19
8.10.2 RS232C-Steuerungsanweisung ... 8-19
8.10.3 Steuerungsanweisung für Programmiergerät ... 8-19
8.11 Multitasking-Steuerungsanweisung ... 8-20
8.11.1 Task-Steuerungsanweisung ... 8-20
8.11.2 Semaphor-Steuerungsanweisung ... 8-20
8.11.3 Spezielle Semaphor-Steueranweisungen ... 8-20
8.12 Steuerung von Zeit und Datum ... 8-21
8.13 Fehlerprüfung ... 8-22
8.14 Systeminformationen ... 8-23
8.15 Präprozessor ... 8-24
8.16 Aufrufen mit einem Wert und einem Bezug ... 8-25
8.16.1 Aufruf mit einem Wert ... 8-25
8.16.2 Aufruf mit Bezug ... 8-26
8.17 Bildverarbeitungssteuerung ... 8-27
8.17.1 Bildeingabe/Ausgabe ... 8-27
8.17.2 Fenstereinstellung ... 8-27
8.17.3 Zeichnen ... 8-28
8.17.4 Bildverarbeitung ... 8-28
8.17.5 Codeerkennung ... 8-29
8.17.6 Bezeichung ... 8-29
8.17.7 Suchfunktion ... 8-29
8.17.8 Abrufen von Ergebnissen ... 8-30
8.17.9 Kalibrierung der Bildverarbeitung ... 8-30

Kapitel 9 Deklarations- anweisungen

9.1 Programmname ... 9-1
9.2 Koordinaten des Interferenzbereichs ... 9-2
9.3 Benutzerfunktion ... 9-4
9.4 Ausgangskoordinaten ... 9-5
9.5 Werkzeugkoordinaten ... 9-6
9.6 Arbeitskoordinaten ... 9-7
9.7 Lokale Variablen ... 9-8
9.8 Array ... 9-17

Kapitel 10 Zuweisungs- anweisungen

10.1 Variablen ... 10-1
10.2 Vektor ... 10-2
10.3 Armstellung ... 10-5
10.4 Verbindungswinkel ... 10-6
10.5 Haltung ... 10-7
10.6 Rotationskomponente ... 10-8
10.7 Achsenkomponente ... 10-12

Kapitel 11 Flußsteuerungsanweisungen

11.1 Programmstop ... 11-1
11.2 Aufruf ... 11-4
11.3 Wiederholung ... 11-9
11.4 Bedingte Verzweigung ... 11-17
11.5 Unbedingte Verzweigung ... 11-21
11.6 Kommentar ... 11-23

Kapitel 12 Roboter-Steueranweisungen

12.1 Bewegungssteuerung 12-1
12.2 Steuerung der Armstellung 12-31
12.3 Steuerungsanweisung für Stop 12-36
12.4 Steuerung der Geschwindigkeit 12-40
12.5 Zeitsteuerung 12-55
12.6 Koordinatentransformation 12-57
12.7 Interferenzprüfung 12-61

Kapitel 13 Steuerungs- anweisungen für die Ein-/ Ausgabe

13.1 E/A-Anschluß 13-1
13.2 RS232C Port 13-8
13.3 Programmiergerät 13-13

Kapitel 14 Die Multitasking-Steueranweisungen

14.1 Task-Steuerung 14-1
14.2 Semaphor 14-6
14.3 Arm-Semaphor 14-14

Kapitel 15 Funktionen

15.1 Arithmetische Funktion 15-1
15.2 Trigonometrische Funktion 15-12
15.3 Winkelumwandlung 15-19
15.4 Geschwindigkeitsumwandlung 15-22
15.5 Zeitfunktion 15-23
15.6 Vektor 15-24
15.7 Transformation von Daten des Typs Stellung 15-28
15.8 Distanzextraktion 15-35
15.9 Armstellungskomponente 15-36
15.10 Winkelkomponente 15-37
15.11 Achsenkomponente 15-38
15.12 Rotationskomponente 15-41
15.13 Armstellungskomponente 15-45
15.14 Positionsfunktion 15-46
15.15 Zeichenfolgefunktion 15-50

Kapitel 16 Konstanten

16.1 Integrierte Konstanten 16-1

Kapitel 17 Steuerung von Zeit/Datum

17.1 Zeit/Datum 17-1

Kapitel 18 Fehlersteuerungen

18.1 Fehlerinformationen 18-1
18.2 Fehlerunterbrechung 18-4

Kapitel 19 Systeminformationen

19.1 System 19-1
19.2 Protokoll 19-4

Kapitel 20 Präprozessor

20.1 Symbolkonstanten · Makrodefinitionen 20-1
20.2 Datei einbinden 20-4
20.3 Optimierung 20-5

Kapitel 21 Bildverarbeitungssteuerung (Option)

21.1 Kompatibilität mit der konventionellen µVision-15 21-1
21.2 Bildeingabe und -ausgabe 21-3
21.3 Fenstereinstellung 21-14
21.4 Zeichnen 21-24
21.5 Bildverarbeitung 21-41
21.6 Codeerkennung 21-64
21.7 Kennzeichnung 21-67
21.8 Suchfunktion 21-76
21.9 Ergebnisse abrufen 21-93

Kapitel 22 PAC-Bibliothek

22.1 Konventionelle Sprache 22-2
22.2 Palettierung 22-18
22.3 Werkzeugoperation 22-44
22.4 Eingabe/Ausgabe 22-49
22.5 Armbewegung 22-50
22.6 Bildverarbeitung 22-92

Anhänge

1 Zeichencodetabelle Anhang-1
2 Abbildungen von Schulter, Ellbogen und Handgelenk Anhang-2
3 Umgebungseinstellwerte Anhang-13
4 Verwenden von Zustandsparametern Anhang-14
5 Liste der reservierten Wörter Anhang-20
6 Befehlsentsprechungstabelle zu herkömmlichen Sprachbefehlen (VS) Anhang-22
7 Kompatibilitätstabelle Anhang-27
8 Parametereinstelltabelle Anhang-28

INDEX

Befehle in alphabetischer Reihenfolge

4 Achsen	6 Achsen	Bildverarbeitungsgerät	
⊙	⊙	⊙	Verfügbar mit allen Serien von Robotern und Bildverarbeitungsgeräten.
○	○	○	Verfügbar mit allen Serien von Robotern. Befehlsspezifikationen unterscheiden sich zwischen Robotern mit 4 oder 6 Achsen und dem Bildverarbeitungsgerät.
⊙	V1.2		Verfügbar mit den Robotern mit 4 Achsen oder 6 Achsen ab Version 1.2 oder neuer.

Befehle	Funktionen	4 Achsen	6 Achsen	Bildverarbeitungsgerät	Siehe:
#					
#define	Ersetzt eine angegebene Konstante oder einen Makronamen in dem Programm durch eine angegebene Zeichenfolge.	⊙	⊙	⊙	20-1
#error	Erzwingt das Generieren eines Kompilierungsfehlers bei Ausführung des Befehls #error.	⊙	⊙	⊙	20-3
#include	Bindet das Präprozessorprogramm ein.	⊙	⊙	⊙	20-4
#pragma optimize	Gibt die auszuführende Optimierung für jedes Programm an.	⊙	⊙	⊙	20-5
#undef	Macht eine mit #define oder einer Makrodefinition definierte Symbolkonstante ungültig.	⊙	⊙	⊙	20-2
A					
ABS	Ruft den absoluten Wert eines Ausdruckwertes ab.	⊙	⊙	⊙	15-1
ACCEL	Gibt die interne Beschleunigung und die interne Verzögerung an.	⊙	⊙		12-42
ACOS	Ermittelt einen Arkuskosinus.	⊙	⊙	⊙	15-12
APPROACH	Führt die im Werkzeugkoordinatensystem angegebene absolute Bewegung aus.	○	○		12-1
AREA	Deklariert den Bereich, in dem eine Interferenzprüfung ausgeführt wird.	○	○		9-2
AREAPOS	Gibt die Mittelpunktposition und die Richtung eines Quaders mit dem Positionstyp für einen Bereich zurück, für den eine Interferenzprüfung durchgeführt wird.	⊙	⊙		15-46
AREASIZE	Gibt die Größe (jede Seitenlänge) eines Quaders zurück, der den Bereich für die Interferenzprüfung mit dem Vektortyp definiert.	⊙	⊙		15-47
ARRIVE	Definiert das Bewegungsverhältnis relativ zu der programmierten vollständigen Distanz bis zum Zielpunkt, um das Programm zur Ausführung des nächsten Schritts bereit zu machen, bis der Roboter das definierte Bewegungsverhältnis erzielt.	⊙	V1.2		12-29
ASC	Führt eine Umwandlung in einen Zeichencode durch.	⊙	⊙	⊙	15-50
ASIN	Ermittelt einen Arkussinus.	⊙	⊙	⊙	15-13
aspACLD	Ändert die internen Lastbedingungswerte. Zu den Lastbedingungswerten gehören die Masse der Nutzlast in Gramm (g) und der Nutzlastschwerpunkt (Angabe in mm). Geben Sie beide Werte an.	⊙	⊙		22-2
aspChange	Wählt den internen Modus für die richtige Steuereinstellung der Bewegungsoptimierung aus.	⊙	⊙		22-4
ATN	Ermittelt einen Arkustangens.	⊙	⊙	⊙	15-14
ATN2	Ermittelt den Arkustangens von Ausdruck 1, geteilt durch Ausdruck 2.	⊙	⊙	⊙	15-15
AVEC	Extrahiert einen Anfahrvektor.	⊙	⊙		15-24

4 Achsen	6 Achsen	Bildverarbei-tungsgerät	
⊙	⊙	⊙	Verfügbar mit allen Serien von Robotern und Bildverarbeitungsgeräten.
○	○	○	Verfügbar mit allen Serien von Robotern. Befehlsspezifikationen unterscheiden sich zwischen Robotern mit 4 oder 6 Achsen und dem Bildverarbeitungsgerät.
⊙	V1.2		Verfügbar mit den Robotern mit 4 Achsen oder 6 Achsen ab Version 1.2 oder neuer.

Befehle	Funktionen	4 Achsen	6 Achsen	Bildverarbei-tungsgerät	Siehe:
B					
BIN$	Wandelt den Wert eines Ausdrucks in eine binäre Zeichenfolge um.	⊙	⊙	⊙	15-51
BLOB	Kennzeichnung ausführen.	⊙	⊙	⊙	21-67
BLOBCOPY	Kopiert eine Objektkennungsnummer.	⊙	⊙	⊙	21-74
BLOBLABEL	Ruft die Kennungsnummer für angegebene Koordinaten ab.	⊙	⊙	⊙	21-72
BLOBMEASURE	Führt eine Funktionsmessung der Objektkennungsnummer aus.	⊙	⊙	⊙	21-70
BUZZER	Gibt einen Buzzer-Ton aus.	⊙	⊙		13-15
C					
CALL	Ruft ein Programm auf und führt es aus.	⊙	⊙	⊙	11-4
CAMIN	Speichert ein Abbild von der Kamera im Bildspeicher (Prozeßfenster).	⊙	⊙	⊙	21-3
CAMLEVEL	Stellt die Eingangsstufe für Kamerabilder ein.	⊙	⊙	⊙	21-6
CAMMODE	Stellt die Funktion zum Speichern eines Kameraabbildes ein.	⊙	⊙	⊙	21-4
CHANGETOOL	Ändert das Werkzeugkoordinatensystem.	⊙	⊙		12-57
CHANGEWORK	Ändert das Benutzerkoordinatensystem.	⊙	⊙		12-58
CHR$	Wandelt einen ASCII-Code in ein Zeichen um.	⊙	⊙	⊙	15-52
CLEARLOG	Initialisiert die Aufzeichnung des Servo-Steuerprotokolls.	⊙	⊙		19-5
COS	Ermittelt einen Kosinus.	⊙	⊙	⊙	15-16
CREATESEM	Erstellt einen Semaphor.	⊙	⊙		14-7
CURACC	Ruft den aktuellen internen Beschleunigungswert ab.	⊙	⊙		12-46
CURDEC	Ruft den aktuellen internen Verzögerungswert ab.	⊙	⊙		12-48
CURFIG	Ruft den aktuellen Wert der Roboterarmstellung ab.	⊙	⊙		12-31
CURJACC	Ruft den aktuellen internen Achsenbeschleunigungswert ab.	⊙	⊙		12-47
CURJDEC	Ruft den aktuellen internen Achsenverzögerungswert ab.	⊙	⊙		12-49
CUREXTACC	Ruft den aktuellen externen Beschleunigungswert ab.	V1.4	V1.4		12-52
CUREXTDEC	Ruft den aktuellen externen Verzögerungswert ab.	V1.4	V1.4		12-53
CUREXTSPD	Ruft den aktuellen externen Geschwindigkeitswert ab.	V1.4	V1.4		12-54
CURJNT	Ruft den aktuellen Winkel des Roboters mit Typ J ab.	○	○		12-23
CURJSPD	Ruft die interne Bewegungsgeschwindigkeit der momentan eingestellten Achse ab.	⊙	⊙		12-50
CURPOS	Ruft die aktuelle Position in dem Werkzeugkoordinatensystem mit Typ P ab.	○	○		12-24
CURSPD	Ruft den aktuellen Wert der internen Bewegungsgeschwindigkeit an der Hand ab.	⊙	⊙		12-51
CURTOOL	Ruft die Nummer des momentan angegebenen Werkzeugs ab.	V1.4	V1.4		12-59
CURTRN	Ruft die aktuelle Position in dem Werkzeugkoordinatensystem mit Typ T ab.	⊙	⊙		12-25
CURWORK	Ruft die Nummer der momentan angegebenen Arbeit ab.	V1.4	V1.4		12-60
D					
DATE$	Ruft das aktuelle Datum ab.	⊙	⊙		17-1
DECEL	Gibt die interne Verzögerung an.	⊙	⊙		12-44
DEF FN	Deklariert eine benutzerdefinierte Funktion.	⊙	⊙	⊙	9-4

4 Achsen	6 Achsen	Bildverarbeitungsgerät	
⊙	⊙	⊙	Verfügbar mit allen Serien von Robotern und Bildverarbeitungsgeräten.
○	○	○	Verfügbar mit allen Serien von Robotern. Befehlsspezifikationen unterscheiden sich zwischen Robotern mit 4 oder 6 Achsen und dem Bildverarbeitungsgerät.
⊙	V1.2		Verfügbar mit den Robotern mit 4 Achsen oder 6 Achsen ab Version 1.2 oder neuer.

Befehle	Funktionen	4 Achsen	6 Achsen	Bildverarbeitungsgerät	Siehe:
DEFDBL	Deklariert eine Variable vom Typ Real mit doppelter Genauigkeit. Der gültige Bereich für Real-Variablen mit doppelter Genauigkeit reicht von - 1,79769313486231D + 308 bis 1,79769313486231D + 308.	⊙	⊙	⊙	9-10
DEFEND	Verteidigt eine Task.	⊙	⊙		14-4
DEFINT	Deklariert eine ganzzahlige Variable. Der gültige Bereich für die Ganzzahl reicht von -2147483648 bis 2147483647.	⊙	⊙	⊙	9-8
DEFIO	Deklariert eine E/A-Variable ensprechend dem Ein-/Ausgabeanschluß.	⊙	⊙	⊙	9-16
DEFJNT	Deklariert eine Variable vom Typ Gelenk.	○	○		9-14
DEFPOS	Deklariert eine Variable vom Typ Position.	○	○		9-13
DEFSNG	Deklariert eine Variable vom Typ Real mit einfacher Genauigkeit. Der gültige Bereich für Real-Variablen mit einfacher Genauigkeit reicht von -3,402823E+38 bis 3,402823E+38.	⊙	⊙	⊙	9-9
DEFSTR	Deklariert eine Variable vom Typ Zeichenfolge. Sie können maximal 247 Zeichen als Zeichenfolge eingeben.	⊙	⊙	⊙	9-11
DEFTRN	Deklariert eine Variable mit einem homogenen Transformationstyp.	⊙	⊙		9-15
DEFVEC	Deklariert eine Variable vom Typ Vektor.	⊙	⊙		9-12
DEGRAD	Wandelt die Einheit in Radianten um.	⊙	⊙	⊙	15-19
DELAY	Unterbricht die Programmverarbeitung für die angegebene Dauer.	⊙	⊙	⊙	12-55
DELETESEM	Löscht einen Semaphor.	⊙	⊙		14-10
DEPART	Führt die relative Bewegung in dem Werkzeugkoordinatensystem aus.	○	○		12-4
DESTJNT	Ermittelt die Zielposition der aktuellen Bewegungsanweisung mit Typ J. Die aktuelle Position (Wert der Anweisung) wird abgerufen, wenn der Roboter stoppt.	○	○		12-26
DESTPOS	Ermittelt die Zielposition der aktuellen Bewegungsanweisung mit Typ P. Wenn der Roboter stoppt, wird der aktuelle Wert (Wert der Anweisung) abgerufen.	○	○		12-27
DESTTRN	Ermittelt die Zielposition der aktuellen Bewegungsanweisung mit Typ T. Wenn der Roboter stoppt, wird die aktuelle Position (Wert der Anweisung) abgerufen.	⊙	⊙		12-28
DIM	Deklariert einen Array.	⊙	⊙	⊙	9-17
dioSync	Führt eine Synchronisation mit einem an DIO angeschlossenen externen Gerät (z. B. einem Sequenzierer) durch.	⊙	⊙		22-49
DIST	Gibt die Distanz zwischen zwei Punkten zurück.	○	○		15-35
DO-LOOP	Führt eine Entscheidungsiteration (Wiederholung) durch.	⊙	⊙	⊙	11-9
DRAW	Führt die im Arbeitskoordinatensystem angegebene relative Bewegung aus.	⊙	⊙		12-7
DRIVE	Führt die relative Bewegung der einzelnen Achsen aus.	⊙	⊙		12-9
DRIVEA	Führt die absolute Bewegung der einzelnen Achsen aus.	⊙	⊙		12-11
E					
END	Deklariert das Bewegungsende über ein Programm.	⊙	⊙	⊙	11-1
ERL	Ruft die Zeilennummer ab, in der ein Fehler auftrat.	⊙	⊙		18-1

4 Achsen	6 Achsen	Bildverarbeitungsgerät	
⊙	⊙	⊙	Verfügbar mit allen Serien von Robotern und Bildverarbeitungsgeräten.
○	○	○	Verfügbar mit allen Serien von Robotern. Befehlsspezifikationen unterscheiden sich zwischen Robotern mit 4 oder 6 Achsen und dem Bildverarbeitungsgerät.
⊙	V1.2		Verfügbar mit den Robotern mit 4 Achsen oder 6 Achsen ab Version 1.2 oder neuer.

Befehle	Funktionen	4 Achsen	6 Achsen	Bildverarbeitungsgerät	Siehe:
ERR	Ruft die Nummer eines aufgetretenen Fehlers ab.	⊙	⊙		18-2
ERRMSG$	Legt eine Fehlermeldung fest.	⊙	⊙	⊙	18-3
EXIT DO	Erzwingt das Beenden der DO-Schleife.	⊙	⊙	⊙	11-11
EXIT FOR	Erzwingt das Beenden des FOR-NEXT-Zweigs.	⊙	⊙	⊙	11-14
EXP	Ruft eine Exponentialfunktion mit einem natürlichen Logarithmus als Basis auf.	⊙	⊙	⊙	15-2
F					
FALSE	Setzt einen Booleschen Wert auf falsch (0).	⊙	⊙	⊙	16-4
FIG	Extrahiert eine Armstellung.	○	○		15-36
FIGAPRL	Berechnet Armstellungen in einer Anfahrposition und in einer Standardposition, die für die Linearbewegung verfügbar ist.	○	○		12-33
FIGAPRP	Berechnet eine Anfahrposition, bei der die Punkt-zu-Punkt-Bewegung verfügbar ist, und eine Armstellung einer Referenzposition.	○	○		12-35
FLUSH	Löscht den Eingabepuffer.	⊙	⊙	⊙	13-12
FLUSHSEM	Gibt die auf einen Semaphor wartenden Tasks frei.	⊙	⊙		14-11
FOR-NEXT	Führt wiederholt eine Reihe von Anweisungen zwischen den Abschnitten FOR-NEXT aus.	⊙	⊙	⊙	11-12
G					
GETENV	Ruft die Werte für die Umgebungseinstellung des Systems ab.	⊙	⊙	⊙	19-1
GIVEARM	Gibt die Priorität der Robotersteuerung frei.	⊙	⊙		14-17
GIVESEM	Gibt eine auf einen Semaphor wartende Task frei.	⊙	⊙		14-12
GIVEVIS	Gibt die Priorität der visuellen Verarbeitung frei.	⊙	⊙		14-19
GOHOME	Bewegt den Roboter an die mit der HOME-Anweisung definierte Position (die Ausgangsposition).	⊙	⊙		12-13
GOSUB	Ruft eine Unterroutine auf.	⊙	⊙	⊙	11-6
GOTO	Führt eine unbedingte Programmverzweigung aus.	⊙	⊙	⊙	11-21
H					
HALT	Stoppt die Ausführung eines Programmes.	⊙	⊙		12-37
HEX$	Ruft einen Wert, der von einer dezimalen in eine hexadezimale Zahl umgewandelt wurde, als Zeichenfolge ab.	⊙	⊙	⊙	15-56
HOLD	Hält die Programmverarbeitung für eine bestimmte Zeit an.	⊙	⊙		12-36
HOME	Deklariert zufällige Koordinaten als Ausgangsposition.	○	○		9-5
I					
IF-END IF	Führt eine bedingte Entscheidung eines bedingten Ausdrucks zwischen IF-END IF aus.	⊙	⊙	⊙	11-17
IF-THEN-ELSE	Führt eine bedingte Entscheidung eines logischen Ausdrucks aus.	⊙	⊙	⊙	11-18
IN	Liest Daten aus dem E/A-Anschluß, der mit einer E/A-Variablen angegeben wurde.	⊙	⊙	⊙	13-1
INPUT	Ruft Daten von dem RS232C-Anschluß ab.	⊙	⊙	⊙	13-8
INT	Ruft den maximalen Wert aus einem angegebenen Wert ab.	⊙	⊙	⊙	15-3
INTERRUPT ON/OFF	Unterbricht eine Roboterbewegung.	⊙	⊙		12-38
IOBLOCK ON/OFF	Führt während der Ausführung einer Bewegungsanweisung gleichzeitig eine Nicht-Bewegungsanweisung wie beispielsweise eine E/A- oder Berechnungsanweisung durch.	⊙	⊙		13-3

4 Achsen	6 Achsen	Bildverarbeitungsgerät	
⊙	⊙	⊙	Verfügbar mit allen Serien von Robotern und Bildverarbeitungsgeräten.
○	○	○	Verfügbar mit allen Serien von Robotern. Befehlsspezifikationen unterscheiden sich zwischen Robotern mit 4 oder 6 Achsen und dem Bildverarbeitungsgerät.
⊙	V1.2		Verfügbar mit den Robotern mit 4 Achsen oder 6 Achsen ab Version 1.2 oder neuer.

Befehle	Funktionen	4 Achsen	6 Achsen	Bildverarbeitungsgerät	Siehe:
J					
J2P	Wandelt Daten vom Typ Gelenk in Daten des Typs Position um.	○	○		15-28
J2T	Wandelt Daten vom Typ Gelenk in Daten des Typs Homogene Transformation um.	○	○		15-29
JACCEL	Gibt die interne Achsenbeschleunigung und die interne Achsenverzögerung an.	⊙	⊙		12-43
JDECEL	Gibt die interne Achsenverzögerung an.	⊙	⊙		12-45
JOINT	Extrahiert einen Winkel aus Koordinaten des Typs Gelenk.	○	○		15-37
JSPEED	Gibt die Bewegungsgeschwindigkeit der Roboterachse an.	⊙	⊙		12-41
K					
KILL	Erzwingt das Beenden einer Task.	⊙	⊙		14-2
L					
LEFT$	Extrahiert den linken Teil einer Zeichenfolge.	⊙	⊙	⊙	15-57
LEN	Ruft die Länge einer Zeichenfolge in Byte ab.	⊙	⊙	⊙	15-58
LET	Weist einer Variablen einen Wert zu.	○	○	○	10-1
LETA	Weist einem Anfahrvektor des homogenen Transformationstyps einen Wert zu.		⊙		10-2
LETENV	Definiert die Werte für die Umgebungseinstellung des Systems.	⊙	⊙	⊙	19-2
LETF	Weist einer Armstellungskomponente des Typs Position oder homogene Transformation einen Wert zu.	⊙	⊙		10-5
LETJ	Weist einem angegebenen Verbindungswinkel des Typs Gelenk einen Wert zu.	⊙	⊙		10-6
LETO	Weist einem Orientierungsvektor des homogenen Transformationstyps einen Wert zu.	⊙	⊙		10-3
LETP	Weist einem Positionsvektor des Typs Position oder homogene Transformation einen Wert zu.	⊙	⊙		10-4
LETR	Weist drei Rotationskomponenten des Typs Position einen Wert zu.		⊙		10-7
LETRX	Weist der X-Achsen-Rotationskomponente des Typs Position einen Wert zu.		⊙		10-8
LETRY	Weist der Y-Achsen-Rotationskomponente des Typs Position einen Wert zu.		⊙		10-9
LETRZ	Weist der Z-Achsen-Rotationskomponente des Typs Position einen Wert zu.		⊙		10-10
LETT	Weist der T-Achsenkomponente des Typs Position einen Wert zu.	⊙			10-11
LETX	Weist der X-Achsenkomponente des Typs Vektor/Position/ homogene Transformation einen Wert zu.	⊙	⊙		10-12
LETY	Weist der Y-Achsenkomponente des Typs Vektor/Position/ homogene Transformation einen Wert zu.	⊙	⊙		10-13
LETZ	Weist der Z-Achsenkomponente des Typs Vektor/Position/homogene Transformation einen Wert zu.	⊙	⊙		10-14
LINEINPUT	Liest Daten über einen RS232C-Anschluß in einem Begrenzer ein und weist sie einer Variablen des Typs Zeichenfolge zu.	⊙	⊙	⊙	13-9
LOG	Ermittelt einen natürlichen Logarithmus.	⊙	⊙	⊙	15-4
LOG10	Ermittelt einen Zehnerlogarithmus.	⊙	⊙	⊙	15-5
M					

4 Achsen	6 Achsen	Bildverarbei-tungsgerät	
⊙	⊙	⊙	Verfügbar mit allen Serien von Robotern und Bildverarbeitungsgeräten.
○	○	○	Verfügbar mit allen Serien von Robotern. Befehlsspezifikationen unterscheiden sich zwischen Robotern mit 4 oder 6 Achsen und dem Bildverarbeitungsgerät.
⊙	V1.2		Verfügbar mit den Robotern mit 4 Achsen oder 6 Achsen ab Version 1.2 oder neuer.

Befehle	Funktionen	4 Achsen	6 Achsen	Bildverarbei-tungsgerät	Siehe:
MAGNITUDE	Ruft die Vektorgröße ab.	⊙	⊙		15-27
MAX	Extrahiert den maximalen Wert.	⊙	⊙	⊙	15-7
MID$	Extrahiert eine Zeichenfolge für die angegebene Anzahl Zeichen aus einer Zeichenfolge.	⊙	⊙	⊙	15-59
MIN	Extrahiert den minimalen Wert.	⊙	⊙	⊙	15-8
MOVE	Bewegt den Roboter an die angegebene Koordinate.	○	○		12-14
MPS	Wandelt einen Ausdruck für die Geschwindigkeit um.	⊙	⊙		15-22
mvResetPulseWidth	Stellt die Impulswerte des Standardcodierers für die zulässige Positionsabweichung wieder her.	⊙	⊙		22-50
mvResetTimeOut	Stellt den Zeitüberschreitungswert für das Ende der Standardbewegung wieder her.	⊙	⊙		22-51
mvReverseFlip	Umkehrung der Armstellung bei 4 Achsen	⊙	⊙		22-52
mvSetPulseWidth	Definiert die zulässige Breite des Stopimpulses.	⊙	⊙		22-53
mvSetTimeOut	Definiert den Zeitüberschreitungswert für das Ende der Bewegung.	⊙	⊙		22-54
N					
ndApra	Führt eine absolute Operation mit dem angegebenen Werkzeugkoordinatensystem aus (ausschließlich für Roboter mit 4 Achsen).	⊙			22-16
ndDepa	Führt eine absolute Operation mit dem angegebenen Werkzeugkoordinatensystem aus (ausschließlich für Roboter mit 4 Achsen).	⊙			22-17
ndInb	Wandelt die Eingabe des angegebenen Anschlusses in eine Dezimalzahl um und behandelt sie wie eine Binärzahl.	⊙	⊙		22-5
ndJf	Eine bedingte Verzweigung nach dem Erhalten von OK/NG von einem externen Gerät (RS232C Eingabe/Ausgabe).	⊙	⊙		22-6
ndOnb	Wandelt eine Dezimalzahl in eine Binärzahl um und gibt sie an einen angegebenen Anschluß aus.	⊙	⊙		22-7
ndOnbI	Wandelt eine Dezimalzahl in eine Binärzahl um und gibt sie an einen angegebenen Anschluß aus.	⊙	⊙		22-8
ndTC	Stellt die WM-Zeit ein.	⊙	V1.2		4-23
ndTS	Legt die Dauer der WM-Zeit fest und die langsame Geschwindigkeit.	⊙	V1.2		4-23
ndVType	Legt das Protokoll für die Kommunikation mit einem externen Gerät fest (RS232C-Ausgabe).	⊙	⊙		22-14
ndVcom	Kommunikation mit einem externen Gerät (Kernel) (RS232C Eingabe/Ausgabe)	⊙	⊙		22-15
ndVdt	Ruft eine Variable ab, die von einem externen Gerät übertragen wurde (RS232C Eingabe/Ausgabe).	⊙	⊙		22-9
ndVis	Überträgt eine angegebene ganze Zahl mit 2 Ziffern an ein externes Gerät (RS232C Eingabe/Ausgabe).	⊙	⊙		22-10
ndVput	Überträgt die Haltungsangabe an ein externes Gerät (RS232C Eingabe/Ausgabe).	⊙	⊙		22-11
ndVrst	Initialisiert ein externes Gerät (RS232C Eingabe/Ausgabe).	⊙	⊙		22-12
ndVset	Empfängt Daten von einem externen Gerät (RS232C-Eingabe/Ausgabe).	⊙	⊙		22-13
O					
OFF	Stellt den Wert OFF (0) ein.	⊙	⊙	⊙	16-1
OffPWM	Gibt die PWM-Umschaltung an der angegebenen Achse frei (ausschließlich für Roboter mit 4 Achsen).	⊙			22-67

4 Achsen	6 Achsen	Bildverarbei-tungsgerät	
⊙	⊙	⊙	Verfügbar mit allen Serien von Robotern und Bildverarbeitungsgeräten.
○	○	○	Verfügbar mit allen Serien von Robotern. Befehlsspezifikationen unterscheiden sich zwischen Robotern mit 4 oder 6 Achsen und dem Bildverarbeitungsgerät.
⊙	V1.2		Verfügbar mit den Robotern mit 4 Achsen oder 6 Achsen ab Version 1.2 oder neuer.

Befehle	Funktionen	4 Achsen	6 Achsen	Bildverarbei-tungsgerät	Siehe:
OffSrvLock	Gibt die Servo-Sperre für die angegebene Achse frei. (Ausschließlich für Roboter mit 4 Achsen)	⊙			22-65
ON	Stellt den Wert ON (1) ein.	⊙	⊙	⊙	16-2
ON ERROR GOTO	Unterbricht die Verarbeitung, wenn ein Fehler auftritt.	⊙	⊙		18-4
ON-GOSUB	Ruft bei einem bestimmten Wert eines Ausdrucks eine entsprechende Unterroutine auf.	⊙	⊙	⊙	11-7
ON-GOTO	Führt bei einem bestimmten Wert eines Ausdrucks eine unbedingte Verzweigung aus.	⊙	⊙	⊙	11-22
OnPWM	Führt eine PWM-Umschaltung an einer angegebenen Achse aus (ausschließlich für Roboter mit 4 Achsen).	⊙			22-66
OnSrvLock	Führt eine Servo-Sperre einer angegebenen Achse aus (ausschließlich für Roboter mit 4 Achsen).	⊙			22-64
ORD	Führt eine Umwandlung in einen Zeichencode durch.	⊙	⊙	⊙	15-60
OUT	Gibt Daten an den mit einer E/A-Variablen angegebenen E/A-Anschluß aus.	⊙	⊙	⊙	13-2
OVEC	Extrahiert einen Orientierungsvektor.	⊙	⊙		15-25
P					
P2J	Wandelt Daten vom Typ Position in Daten des Typs Gelenk um.	○	○		15-30
P2T	Wandelt Daten vom Typ Position in Daten des Typs Homogene Transformation um.	○	○		15-31
PI	Stellt einen Wert π ein.	⊙	⊙	⊙	16-3
pltDecCnt	Verringert die Zahl im Paletierungs-Gesamtzähler.	⊙	⊙		22-18
pltGetCnt	Ruft den Inhalt des Paletierungs-Gesamtzählers ab.	⊙	⊙		22-19
pltGetK	Ruft den Wert K des Paletierungssatzes ab.	⊙	⊙		22-20
pltGetK1	Ruft den Inhalt des Paletierungszählers K1 ab.	⊙	⊙		22-21
pltGetM	Ruft den Wert M des Paletierungssatzes ab.	⊙	⊙		22-22
pltGetM1	Ruft den Inhalt des Paletierungszählers M1 ab.	⊙	⊙		22-23
pltGetN	Ruft den Wert N des Palettierungssatzes ab.	⊙	⊙		22-24
pltGetN1	Ruft den Inhalt des Palettierungszählers N1 ab.	⊙	⊙		22-25
pltGetNextPos	Ruft die nächste Position ab.	⊙	⊙		22-26
pltGetPLT1END	Ruft eine Markierung Palettierung 1. Zeile ab.	⊙	⊙		22-27
pltGetPLTEND	Ruft eine Markierung Palettierung alle Zeilen ab.	⊙	⊙		22-28
pltIncCnt	Erhöht den Inhalt des Palettierungsgesamtzählers.	⊙	⊙		22-29
pltInit1	Palettierungsinitialisierung Schablone 1.	⊙	⊙		22-30
pltInitialize	Palettierungsinitialisierung	⊙	⊙		22-31
pltKernel	Palettierungsbewegung (Kernel)	⊙	⊙		22-32
pltLetCnt	Definiert den Inhalt des Palettierungsgesamtzählers.	⊙	⊙		22-33
pltLetK1	Definiert den Inhalt des Palettierungszählers K1.	⊙	⊙		22-34
pltLetM1	Definiert den Inhalt des Palettierungszählers M1.	⊙	⊙		22-35
pltLetN1	Definiert den Inhalt des Palettierungszählers N1.	⊙	⊙		22-36
pltMain1	Palettierung Schablone 1	⊙	⊙		22-37
pltMain2	Palettierung Schablone 2	⊙	⊙		22-38
pltMove	Standardpalettierung Schablone 1	⊙	⊙		22-39
pltMove0	Standardpalettierung Bewegung 1	⊙	⊙		22-40
pltResetAll	Setzt alle Palettierungszähler zurück.	⊙	⊙		22-41
pltResetPLT1END	Setzt eine Markierung Palettierung 1. Zeile zurück.	⊙	⊙		22-42
pltResetPLTEND1	Setzt eine Markierung Palettierung alle Zeilen zurück.	⊙	⊙		22-43
POSRX	Extrahiert die X-Achsen-Rotationskomponente.		⊙		15-41
POSRY	Extrahiert die Y-Achsen-Rotationskomponente.		⊙		15-42

4 Achsen	6 Achsen	Bildverarbei-tungsgerät	
⊙	⊙	⊙	Verfügbar mit allen Serien von Robotern und Bildverarbeitungsgeräten.
○	○	○	Verfügbar mit allen Serien von Robotern. Befehlsspezifikationen unterscheiden sich zwischen Robotern mit 4 oder 6 Achsen und dem Bildverarbeitungsgerät.
⊙	V1.2		Verfügbar mit den Robotern mit 4 Achsen oder 6 Achsen ab Version 1.2 oder neuer.

Befehle	Funktionen	4 Achsen	6 Achsen	Bildverarbei-tungsgerät	Siehe:
POSRZ	Extrahiert die Z-Achsen-Rotationskomponente.		⊙		15-43
POST	Extrahiert die T-Achsen-Rotationskomponente.	⊙			15-44
POSX	Extrahiert die X-Komponente.	○	○		15-38
POSY	Extrahiert die Y-Komponente.	○	○		15-39
POSZ	Extrahiert die Z-Komponente.	○	○		15-40
POW	Ermittelt einen Exponenten.	⊙	⊙	⊙	15-6
PRINT	Gibt Daten an einen RS232C-Anschluß aus.	⊙	⊙	⊙	13-10
PRINTDBG	Gibt Daten an das Debug-Fenster aus.	⊙	⊙		13-14
PRINTLBL	Definiert eine Kennung (Beschriftung) für eine benutzerdefinierte Schaltfläche.	⊙	⊙		13-16
PRINTMSG	Zeigt eine Meldung mit einer Beschriftung und einem Symbol auf der Farb-LCD des Programmiergeräts an.	⊙	⊙		13-13
PROGRAM	Deklariert einen Programmnamen.	⊙	⊙	⊙	9-1
PVEC	Extrahiert einen Positionsvektor.	○	○		15-26
R					
RAD	Wandelt einen in Radianten angegebenen Wert in Grad um.	⊙	⊙	⊙	15-20
RADDEG	Wandelt die Einheit in Grad um.	⊙	⊙	⊙	15-21
REM	Gibt einen Kommentar an.	⊙	⊙	⊙	11-23
REPEAT-UNTIL	Führt eine Endentscheidungsiteration (Wiederholung) durch.	⊙	⊙	⊙	11-15
RESET	Stellt einen E/A-Anschluß auf OFF ein.	⊙	⊙	⊙	13-7
RESETAREA	Initialisiert eine Interferenzprüfung.	⊙	⊙		12-62
ResetCompJLimit	Initialisiert den aktuellen Grenzwert unter der Nachgiebigkeitssteuerung.		V1.4		22-83
ResetCompRate	Initialisiert die Nachgiebigkeitsrate unter der Nachgiebigkeitssteuerung.		V1.4		22-79
ResetCompVMode	Deaktiviert den Geschwindigkeits-Steuermodus unter der Nachgiebigkeitssteuerung.		V1.4		22-85
ResetCurLmt	Setzt den Motorstromgrenzwert für die angegebene Achse zurück.	⊙	V1.2		22-61
Resetcycloid	Bewirkt einen Übergang vom Zykloidmodus in den normalen Betriebsmodus.	⊙	V1.4		22-69
ResetDampRate	Initialisiert die Dämpfungsrate unter der Nachgiebigkeitssteuerung.		V1.4		22-89
ResetEralw	Setzt den Wert für die zulässige Abweichung der angegebenen Achse auf den Anfangswert zurück.	⊙	V1.2		22-63
ResetFrcAssist	Initialisiert die Kraftunterstützung unter der Nachgiebigkeitssteuerung.		V1.4		22-81
ResetFrcLimit	Initialisiert die Rate der Kraftbegrenzung.		V1.4		22-77
ResetGravity	Deaktiviert die Balanceeinstellung zwischen dem begrenzten Motordrehmoment und dem Schwerkraftdrehmoment, die mit SetGravity definiert wurde.		V1.2		22-56
ResetGrvOffset	Deaktiviert die Schwerkraft-Offsetfunktion.		V1.2		22-58
ResetVibControl	Kehrt vom Steuermodus zur Reduzierung der Restvibration zurück in den normalen Steuermodus.		V1.4		22-91
RESUME	Kehrt von einer Prozessunterbrechungsroutine zurück.	⊙	⊙		18-5
RETURN	Kehrt aus einer Unterroutine zurück.	⊙	⊙	⊙	11-8

4 Achsen	6 Achsen	Bildverarbeitungsgerät	
⊙	⊙	⊙	Verfügbar mit allen Serien von Robotern und Bildverarbeitungsgeräten.
○	○	○	Verfügbar mit allen Serien von Robotern. Befehlsspezifikationen unterscheiden sich zwischen Robotern mit 4 oder 6 Achsen und dem Bildverarbeitungsgerät.
⊙	V1.2		Verfügbar mit den Robotern mit 4 Achsen oder 6 Achsen ab Version 1.2 oder neuer.

Befehle	Funktionen	4 Achsen	6 Achsen	Bildverarbeitungsgerät	Siehe:
RIGHT$	Extrahiert den rechten Teil einer Zeichenfolge.	⊙	⊙	⊙	15-61
RND	Generiert Zufallszahlen von 0 bis 1.	⊙	⊙	⊙	15-9
ROTATE	Führt eine Rotationsbewegung um eine angegebene Achse aus.	○	○		12-18
ROTATEH	Führt eine Drehbewegung aus, indem ein Anfahrvektor als Achse verwendet wird.	⊙	⊙		12-21
RUN	Führt gleichzeitig ein anderes Programm aus.	⊙	⊙		14-1
RVEC	Extrahiert eine Armstellung.		⊙		15-45
S					
SEC	Wandelt einen in Sekunden ausgedrückten Wert in Millisekunden um.	⊙	⊙	⊙	15-23
SELECT CASE	Führt eine Entscheidung zu einer Mehrfachbedingung aus.	⊙	⊙	⊙	11-19
SET	Stellt einen E/A-Anschluß auf ON ein.	⊙	⊙	⊙	13-5
SETAREA	Wählt den Bereich aus, in dem eine Interferenzprüfung ausgeführt wird.	⊙	⊙		12-61
SetCompControl	Aktiviert eine Funktion zur Nachgiebigkeitssteuerung.		V1.4		22-72
SetCompEralw	Gibt die Werte für die zulässige Abweichung der Position und der Haltung der Tool-Spitze unter der Nachgiebigkeitssteuerung an.		V1.4		22-86
SetCompFControl	Aktiviert eine Funktion zur Nachgiebigkeitssteuerung. Die Verarbeitung zur Kompensierung des Schwerkraft-Offset ist jedoch nicht aktiv.		V1.4		22-73
SetCompJLimit	Legt den aktuellen Grenzwert unter der Nachgiebigkeitssteuerung fest.		V1.4		22-82
SetCompRate	Legt die Nachgiebigkeitsrate unter der Nachgiebigkeitssteuerung fest.		V1.4		22-78
SetCompVMode	Legt den Geschwindigkeits-Steuermodus unter der Nachgiebigkeitssteuerung fest.		V1.4		22-84
SetCurLmt	Legt den Grenzwert für den Motorstrom fest, der auf die angegebene Achse angewendet werden soll.	○	V1.2		22-59
Setcycloid	Bewirkt einen Wechsel in den Zykloid-Betriebsmodus zur Unterdrückung von Überschuß und Restvibration am Ende der Punkt-zu-Punkt-Operation.	⊙	V1.4		22-68
SetDampRate	Legt die Dämpfungsrate unter der Nachgiebigkeitssteuerung fest.		V1.4		22-88
SetEralw	Ändert die zulässige Abweichung für die angegebene Achse.	⊙	V1.2		22-62
SetForce_HC	Eine Strombegrenzungsbibliothek, die den Schub (Einheit: N) der Z-Koordinate mit dem HC-Roboter angibt.	⊙			22-71
SetForce_HM	Eine Strombegrenzungsbibliothek, die den Schub (Einheit: N) der Z-Koordinate mit dem HM/HS-Roboter angibt.	⊙			22-70
SetFrcAssist	Legt die Kraftunterstützung unter der Nachgiebigkeitssteuerung fest.		V1.4		22-80
SetFrcCoord	Wählt ein Koordinatensystem für die Einstellung der Nachgiebigkeitssteuerung aus.		V1.4		22-75
SetFrcLimit	Legt die Rate der Kraftbegrenzung fest.		V1.4		22-76
SetGravity	Kompensiert die auf die einzelnen Gelenke angewendete statische Last (Schwerkraft-Drehmoment) und erzielt eine Balance mit dem Schwerkraft-Drehmoment.		V1.2		22-55

4 Achsen	6 Achsen	Bildverarbei-tungsgerät	
⊙	⊙	⊙	Verfügbar mit allen Serien von Robotern und Bildverarbeitungsgeräten.
○	○	○	Verfügbar mit allen Serien von Robotern. Befehlsspezifikationen unterscheiden sich zwischen Robotern mit 4 oder 6 Achsen und dem Bildverarbeitungsgerät.
⊙	V1.2		Verfügbar mit den Robotern mit 4 Achsen oder 6 Achsen ab Version 1.2 oder neuer.

Befehle	Funktionen	4 Achsen	6 Achsen	Bildverarbei-tungsgerät	Siehe:
SetGrvOffset	Kompensiert das Drehmoment jedes Gelenks, das mit SetGravity für das Schwerkraft-Drehmoment programmiert wurde.		V1.2		22-57
SetVibControl	Legt den Steuermodus für die Reduzierung der Restvibration fest.		V1.4		22-90
SGN	Überprüft ein Vorzeichen.	⊙	⊙	⊙	15-10
SHCIRCLE	Sucht einen Kreis.	⊙	⊙	⊙	21-90
SHCLRMODEL	Löscht ein registriertes Modell.	⊙	⊙	⊙	21-80
SHCOPYMODEL	Kopiert ein registriertes Modell.	⊙	⊙	⊙	21-79
SHCORNER	Sucht eine Ecke.	⊙	⊙	⊙	21-87
SHDEFCIRCLE	Legt die Bedingung zum Suchen eines Kreises fest.	⊙	⊙	⊙	21-89
SHDEFCORNER	Legt die Bedingungen zum Suchen einer Ecke fest.	⊙	⊙	⊙	21-86
SHDEFMODEL	Registriert das Suchmodell.	⊙	⊙	⊙	21-76
SHDISPMODEL	Zeigt ein registriertes Modell auf dem Bildschirm an.	⊙	⊙	⊙	21-81
SHMODEL	Sucht ein Modell.	⊙	⊙	⊙	21-82
SHREFMODEL	Bezieht sich auf die Daten des registrierten Modells.	⊙	⊙	⊙	21-78
SIN	Ermittelt einen Sinus.	⊙	⊙	⊙	15-17
SPEED	Gibt die interne Geschwindigkeit der Hand an.	⊙	⊙		12-40
SPRINTF$	Wandelt einen Ausdruck in ein angegebenes Format um und gibt ihn als Zeichenfolge zurück.	⊙	⊙	⊙	15-53
SQR	Ermittelt die Quadratwurzel.	⊙	⊙	⊙	15-11
STARTLOG	Startet die Aufzeichnung des Servo-Steuerprotokolls.	⊙	⊙		19-4
STATUS	Ermittellt den Programmstatus.	⊙	⊙		14-5
STOP	Beendet die Programmausführung.	⊙	⊙	⊙	11-2
STOPEND	Diese Anweisung stoppt ein kontinuierlich ausgeführtes Programm oder ein Pogramm mit einer Zyklusoption nach einem Zyklus. Wenn ein Zyklus eines Programmes mit dieser Anweisung gestartet wird, hat dies keine Auswirkung auf die Bewegung, auch wenn diese Anweisung ausgeführt wird.	⊙	⊙		11-3
STOPLOG	Stoppt die Aufzeichnung des Servo-Steuerprotokolls.	⊙	⊙		19-6
STR$	Wandelt einen Wert in eine Zeichenfolge um.	⊙	⊙	⊙	15-63
STRPOS	Ruft die Position einer Zeichenfolge ab.	⊙	⊙	⊙	15-62
SUSPEND	Unterbricht eine Task.	⊙	⊙		14-3
T					
T2J	Wandelt Daten des Typs homogene Transformation in Daten des Typs Gelenk um.	⊙	⊙		15-32
T2P	Wandelt Daten vom Typ homogene Transformation in Daten des Typs Position um.	⊙	⊙		15-33
TAKEARM	Ruft die Priorität der Robotersteuerung ab, gibt eine Werkzeugkoordinate und eine Arbeitskoordinate an die Ausgangsposition zurück und stellt die interne Geschwindigkeit, Beschleunigung und Verzögerung auf 100 ein.	⊙	⊙		14-14
TAKESEM	Ruft einen Semaphor mit einer angegebenen Semaphor-ID ab.	⊙	⊙		14-13
TAKEVIS	Ruft die Priorität der visuellen Verarbeitung ab.	⊙	⊙		14-18
TAN	Ermittelt einen Tangens.	⊙	⊙	⊙	15-18
TIME$	Ruft die aktuelle Uhrzeit ab.	⊙	⊙		17-2
TIMER	Ruft die abgelaufene Zeit ab.	⊙	⊙		17-3

4 Achsen	6 Achsen	Bildverarbei-tungsgerät	
⊙	⊙	⊙	Verfügbar mit allen Serien von Robotern und Bildverarbeitungsgeräten.
○	○	○	Verfügbar mit allen Serien von Robotern. Befehlsspezifikationen unterscheiden sich zwischen Robotern mit 4 oder 6 Achsen und dem Bildverarbeitungsgerät.
⊙	V1.2		Verfügbar mit den Robotern mit 4 Achsen oder 6 Achsen ab Version 1.2 oder neuer.

Befehle	Funktionen	4 Achsen	6 Achsen	Bildverarbei-tungsgerät	Siehe:
TINV	Berechnet eine inverse Matrix von Daten des Typs homogene Transformation.	⊙	⊙		15-34
tolChange	Wechsel des Werkzeugs	⊙	⊙		22-44
tolInit1	Werkzeugänderung Initialisierung Schablone 1	⊙	⊙		22-45
tolInitialize	Werkzeugänderung Initialisierung	⊙	⊙		22-46
tolKernel	Werkzeugänderung Bewegung (Kernel)	⊙	⊙		22-47
tolMain1	Werkzeugänderung Initialisierung Schablone 1	⊙	⊙		22-48
TOOL	Deklariert ein Werkzeugkoordinatensystem.	○	○		9-6
TOOLPOS	Gibt ein Werkzeugkoordinatensystem als Positionstyp zurück.	⊙	⊙		15-48
TRUE	Setzt einen Booleschen Wert auf wahr (1).	⊙	⊙	⊙	16-5
V					
VAL	Wandelt eine Zeichenfolge in einen numerischen Wert um.	⊙	⊙	⊙	15-64
VER$	Ruft die Version jedes Moduls ab.	⊙	⊙	⊙	19-3
VISBINA	Binärisiert die Anzeige.	⊙	⊙	⊙	21-47
VISBINAR	Zeigt einen binärisierten Bildschirm an.	⊙	⊙	⊙	21-49
VISBRIGHT	Gibt einen Helligkeitswert für die Zeichnung an.	⊙	⊙	⊙	21-26
VISCAMOUT	Zeigt ein Bild von der Kamera auf dem Monitor an.	⊙	⊙	⊙	21-7
VISCIRCLE	Zeichnet einen Kreis auf dem Bildschirm.	⊙	⊙	⊙	21-33
VISCLS	Füllt (bereinigt) eine angegebene Anzeige, die in einem Modus mit einer angegebenen Helligkeit definiert wurde.	⊙	⊙	⊙	21-27
VISCOPY	Kopiert die Anzeige.	⊙	⊙	⊙	21-54
VISCROSS	Zeichnet ein Kreuzsymbol auf dem Bildschirm.	⊙	⊙	⊙	21-36
VISDEFCHAR	Gibt die Größe der Zeichen und die Anzeigemethode an.	⊙	⊙	⊙	21-39
VISDEFTABLE	Liest Abbilder von der Kamera ein und stellt die Nachschlage-Tabellendaten für die Abbildausgabe ein.	⊙	⊙	⊙	21-11
VISEDGE	Mißt die Kante in einem Fenster.	⊙	⊙	⊙	21-60
VISELLIPSE	Zeichnet eine Ellipse auf dem Bildschirm.	⊙	⊙	⊙	21-34
VISFILTER	Führt eine Filterung auf der Anzeige aus.	⊙	⊙	⊙	21-50
VISGETNUM	Ruft ein Ergebnis einer Abbildverarbeitung aus dem Speicher ab.	⊙	⊙	⊙	21-93
VISGETP	Ruft eine angegebene Koordinatenhelligkeit aus dem Speicher ab (Prozeßfenster).	⊙	⊙	⊙	21-42
VISGETSTR	Ruft das Ergebnis der Codeerkennung ab.	⊙	⊙	⊙	21-94
VISHIST	Ruft das Histogramm (die Helligkeitsverteilung) des Bildschirms ab.	⊙	⊙	⊙	21-43
VISLEVEL	Ruft eine Binärisierungsstufe entsprechend dem Ergebnis des Histogrammes ab.	⊙	⊙	⊙	21-45
VISLINE	Zeichnet eine Linie auf dem Bildschirm.	⊙	⊙	⊙	21-30
VISLOC	Kennzeichnet die Anzeigeposition der Zeichen.	⊙	⊙	⊙	21-37
VISMASK	Führt Berechnungen zwischen Abbildern aus.	⊙	⊙	⊙	21-52
VISMEASURE	Mißt Merkmale in dem Fenster (Bereich, Schwerpunkt, Winkel der Hauptachse).	⊙	⊙	⊙	21-55
VISOVERLAY	Zeigt Zeichenbildschirm-Informationen auf dem Monitor an.	⊙	⊙	⊙	21-9
VISPLNOUT	Zeigt ein Abbild im Speicher des Monitors an.	⊙	⊙	⊙	21-8
VISPOSX	Ruft ein Ergebnis einer Abbildverarbeitung (Koordinate X) aus dem Speicher ab.	⊙	⊙	⊙	21-95
VISPOSY	Ruft ein Ergebnis einer Abbildverarbeitung (Koordinate Y) aus dem Speicher ab.	⊙	⊙	⊙	21-96
VISPRINT	Zeigt Zeichen und Abbildungen auf dem Bildschirm an.	⊙	⊙	⊙	21-40

4 Achsen	6 Achsen	Bildverarbei-tungsgerät	
⊙	⊙	⊙	Verfügbar mit allen Serien von Robotern und Bildverarbeitungsgeräten.
○	○	○	Verfügbar mit allen Serien von Robotern. Befehlsspezifikationen unterscheiden sich zwischen Robotern mit 4 oder 6 Achsen und dem Bildverarbeitungsgerät.
⊙	V1.2		Verfügbar mit den Robotern mit 4 Achsen oder 6 Achsen ab Version 1.2 oder neuer.

Befehle	Funktionen	4 Achsen	6 Achsen	Bildverarbei-tungsgerät	Siehe:
VISPROJ	Mißt die projizierten Daten in dem Fenster.	⊙	⊙	⊙	21-58
VISPTP	Zeichnet eine Linie als Verbindung zwischen zwei Punkten auf dem Bildschirm.	⊙	⊙	⊙	21-31
VISPUTP	Zeichnet einen Punkt auf dem Bildschirm.	⊙	⊙	⊙	21-29
VISREADQR	Liest den QR-Code.	⊙	⊙	⊙	21-64
VISRECT	Zeichnet ein Rechteck auf dem Bildschirm.	⊙	⊙	⊙	21-32
VISREFCAL	Ruft Kalibrierungsdaten (Koordinatenumwandlung Bildverarbeitung-Roboter) ab.	⊙	⊙	⊙	21-98
VISREFHIST	Liest Ergebnisse des Histogrammes.	⊙	⊙	⊙	21-44
VISREFTABLE	Verweist auf Daten in der Nachschlagetabelle.	⊙	⊙	⊙	21-13
VISSCREEN	Gibt einen Zeichnungsbildschirm an.	⊙	⊙	⊙	21-24
VISSECT	Zeichnet einen Sektor auf dem Bildschirm.	⊙	⊙	⊙	21-35
VISSTATUS	Überwacht das Verarbeitungsergebnis jeder Anweisung.	⊙	⊙	⊙	21-97
VISWORKPLN	Gibt den zu verarbeitenden Speicher (Prozeßfenster) an.	⊙	⊙	⊙	21-41
viTran6	Wandelt die Bildverarbeitungskoordinaten in Roboterkoordinaten um (für 6 Achsen).	⊙	⊙		22-92
W					
WAIT	Stoppt die Programmverarbeitung abhängig von einer Bedingung.	⊙	⊙		12-56
WHILE-WEND	Führt eine Anfangs-Entscheidungsiteration durch.	⊙	⊙	⊙	11-16
WINDCLR	Löscht eine eingestellte Fensterinformation.	⊙	⊙	⊙	21-19
WINDCOPY	Kopiert Fensterdaten.	⊙	⊙	⊙	21-20
WINDDISP	Zeichnet ein angegebenes Fenster.	⊙	⊙	⊙	21-23
WINDMAKE	Zeichnet einen Bereich für die Abbildverarbeitung.	⊙	⊙	⊙	21-14
WINDREF	Ruft Fensterinformationen ab.	⊙	⊙	⊙	21-22
WORK	Deklariert ein Benutzerkoordinatensystem.	○	○		9-7
WORKPOS	Gibt das Benutzerkoordinatensystem als Positionstyp zurück.	⊙	⊙		15-49
WRITE	Gibt Daten an einen RS232C-Anschluß aus.	⊙	⊙	⊙	13-11

Befehle nach Funktionen geordnet

4 Achsen	6 Achsen	Bildverarbeitungsgerät	
⊙	⊙	⊙	Verfügbar mit allen Serien von Robotern und Bildverarbeitungsgeräten.
○	○	○	Verfügbar mit allen Serien von Robotern. Befehlsspezifikationen unterscheiden sich zwischen Robotern mit 4 oder 6 Achsen und dem Bildverarbeitungsgerät.
⊙	V1.2		Verfügbar mit den Robotern mit 4 Achsen oder 6 Achsen ab Version 1.2 oder neuer.

Nach Funktionen klassifiziert	Befehle	Funktionen	4 Achsen	6 Achsen	Bildverarbeitungsgerät	Siehe:
Deklarationsanweisungen						
Programmname	PROGRAM	Deklariert einen Programmnamen.	⊙	⊙	⊙	9-1
Interferenzbereichskoordinaten	AREA	Deklariert den Bereich, in dem eine Interferenzprüfung ausgeführt wird.	○	○		9-2
Benutzerfunktion	DEF FN	Deklariert eine benutzerdefinierte Funktion.	⊙	⊙	⊙	9-4
Ausgangskoordinaten	HOME	Deklariert zufällige Koordinaten als Ausgangsposition.	○	○		9-5
Werkzeugkoordinaten	TOOL	Deklariert ein Werkzeugkoordinatensystem.	○	○		9-6
Arbeitskoordinaten	WORK	Deklariert ein Benutzerkoordinatensystem.	○	○		9-7
Lokale Variablen	DEFINT	Deklariert eine ganzzahlige Variable. Der gültige Bereich für die Ganzzahl reicht von -2147483648 bis 2147483647.	⊙	⊙	⊙	9-8
Ganze Zahl						
Gleitkomma	DEFSNG	Deklariert eine Variable vom Typ Real mit einfacher Genauigkeit. Der gültige Bereich für Real-Variablen mit einfacher Genauigkeit reicht von -3,402823E+38 bis 3,402823E+38.	⊙	⊙	⊙	9-9
Doppelte Genauigkeit	DEFDBL	Deklariert eine Variable vom Typ Real mit doppelter Genauigkeit. Der gültige Bereich für Real-Variablen mit doppelter Genauigkeit reicht von -1,79769313486231D + 308 bis 1,79769313486231D + 308.	⊙	⊙	⊙	9-10
Zeichenfolge	DEFSTR	Deklariert eine Variable vom Typ Zeichenfolge. Sie können maximal 247 Zeichen als Zeichenfolge eingeben.	⊙	⊙	⊙	9-11
Vektor	DEFVEC	Deklariert eine Variable vom Typ Vektor.	⊙	⊙		9-12
Position	DEFPOS	Deklariert eine Variable vom Typ Position.	○	○		9-13
Gelenk	DEFJNT	Deklariert eine Variable vom Typ Gelenk.	○	○		9-14
Homogene Transformationsmatrix	DEFTRN	Deklariert eine Variable mit einem homogenen Transformationstyp.	⊙	⊙		9-15
E/A	DEFIO	Deklariert eine E/A-Variable ensprechend dem Ein-/Ausgabeanschluß.	⊙	⊙	⊙	9-16
Array	DIM	Deklariert einen Array.	⊙	⊙	⊙	9-17
Zuweisungsanweisung						
Variablen	LET	Weist einer Variablen einen Wert zu.	○	○	○	10-1
Vektor	LETA	Weist einem Anfahrvektor des homogenen Transformationstyps einen Wert zu.		⊙		10-2

4 Achsen	6 Achsen	Bildverarbeitungsgerät	
⊙	⊙	⊙	Verfügbar mit allen Serien von Robotern und Bildverarbeitungsgeräten.
○	○	○	Verfügbar mit allen Serien von Robotern. Befehlsspezifikationen unterscheiden sich zwischen Robotern mit 4 oder 6 Achsen und dem Bildverarbeitungsgerät.
⊙	V1.2		Verfügbar mit den Robotern mit 4 Achsen oder 6 Achsen ab Version 1.2 oder neuer.

Nach Funktionen klassifiziert	Befehle	Funktionen	4 Achsen	6 Achsen	Bildverarbeitungsgerät	Siehe:
	LETO	Weist einem Orientierungsvektor des homogenen Transformationstyps einen Wert zu.	⊙	⊙		10-3
	LETP	Weist einem Positionsvektor des Typs Position oder homogene Transformation einen Wert zu.	⊙	⊙		10-4
Armstellung	LETF	Weist einer Armstellungskomponente des Typs Position oder homogene Transformation einen Wert zu.	⊙	⊙		10-5
Verbindungswinkel	LETJ	Weist einem angegebenen Verbindungswinkel des Typs Gelenk einen Wert zu.	⊙	⊙		10-6
Haltung	LETR	Weist drei Rotationskomponenten des Typs Position einen Wert zu.		⊙		10-7
Rotationskomponente	LETRX	Weist der X-Achsen-Rotationskomponente des Typs Position einen Wert zu.		⊙		10-8
	LETRY	Weist der Y-Achsen-Rotationskomponente des Typs Position einen Wert zu.		⊙		10-9
	LETRZ	Weist der Z-Achsen-Rotationskomponente des Typs Position einen Wert zu.		⊙		10-10
	LETT	Weist der T-Achsenkomponente des Typs Position einen Wert zu.	⊙			10-11
Achsenkomponente	LETX	Weist der X-Achsenkomponente des Typs Vektor/Position/homogene Transformation einen Wert zu.	⊙	⊙		10-12
	LETY	Weist der Y-Achsenkomponente des Typs Vektor/Position/homogene Transformation einen Wert zu.	⊙	⊙		10-13
	LETZ	Weist der Z-Achsenkomponente des Typs Vektor/Position/homogene Transformation einen Wert zu.	⊙	⊙		10-14
Flußsteuerungsanweisungen						
Programm-Stop	END	Deklariert das Bewegungsende über ein Programm.	⊙	⊙	⊙	11-1
	STOP	Beendet die Programmausführung.	⊙	⊙	⊙	11-2
	STOPEND	Diese Anweisung stoppt ein kontinuierlich ausgeführtes Programm oder ein Pogramm mit einer Zyklusoption nach einem Zyklus. Wenn ein Zyklus eines Programmes mit dieser Anweisung gestartet wird, hat dies keine Auswirkung auf die Bewegung, auch wenn diese Anweisung ausgeführt wird.	⊙	⊙		11-3

4 Achsen	6 Achsen	Bildverarbei-tungsgerät	
⊙	⊙	⊙	Verfügbar mit allen Serien von Robotern und Bildverarbeitungsgeräten.
○	○	○	Verfügbar mit allen Serien von Robotern. Befehlsspezifikationen unterscheiden sich zwischen Robotern mit 4 oder 6 Achsen und dem Bildverarbeitungsgerät.
⊙	V1.2		Verfügbar mit den Robotern mit 4 Achsen oder 6 Achsen ab Version 1.2 oder neuer.

Nach Funktionen klassifiziert	Befehle	Funktionen	4 Achsen	6 Achsen	Bildverarbei-tungsgerät	Siehe:
Aufruf	CALL	Ruft ein Programm auf und führt es aus.	⊙	⊙	⊙	11-4
	GOSUB	Ruft eine Unterroutine auf.	⊙	⊙	⊙	11-6
	ON-GOSUB	Ruft bei einem bestimmten Wert eines Ausdrucks eine entsprechende Unterroutine auf.	⊙	⊙	⊙	11-7
	RETURN	Kehrt aus einer Unterroutine zurück.	⊙	⊙	⊙	11-8
Wiederholung	DO-LOOP	Führt eine Entscheidungsiteration (Wiederholung) durch.	⊙	⊙	⊙	11-9
	EXIT DO	Erzwingt das Beenden der DO-SCHLEIFE.	⊙	⊙	⊙	11-11
	FOR-NEXT	Führt wiederholt eine Reihe von Anweisungen zwischen den Abschnitten FOR-NEXT aus.	⊙	⊙	⊙	11-12
	EXIT FOR	Erzwingt das Beenden des FOR-NEXT-Zweigs.	⊙	⊙	⊙	11-14
	REPEAT-UNTIL	Führt eine Endentscheidungsiteration (Wiederholung) durch.	⊙	⊙	⊙	11-15
	WHILE-WEND	Führt eine Anfangsentscheidungsiteration durch.	⊙	⊙	⊙	11-16
Bedingte Verzweigung	IF-END IF	Führt eine bedingte Entscheidung eines bedingten Ausdrucks zwischen IF-END IF aus.	⊙	⊙	⊙	11-17
	IF-THEN-ELSE	Führt eine bedingte Entscheidung eines logischen Ausdrucks aus.	⊙	⊙	⊙	11-18
	SELECT CASE	Führt eine Entscheidung zu einer Mehrfachbedingung aus.	⊙	⊙	⊙	11-19
Unbedingte Verzweigung	GOTO	Führt eine unbedingte Programmverzweigung aus.	⊙	⊙	⊙	11-21
	ON-GOTO	Führt bei einem bestimmten Wert eines Ausdrucks eine unbedingte Verzweigung aus.	⊙	⊙	⊙	11-22
Kommentar	REM	Gibt einen Kommentar an.	⊙	⊙	⊙	11-23
Robotersteuerungsanweisungen						
Bewegungssteuerung	APPROACH	Führt die im Werkzeugkoordinatensystem angegebene absolute Bewegung aus.	○	○		12-1
	DEPART	Führt die relative Bewegung in dem Werkzeugkoordinatensystem aus.	○	○		12-4
	DRAW	Führt die im Arbeitskoordinatensystem angegebene relative Bewegung aus.	⊙	⊙		12-7
	DRIVE	Führt die relative Bewegung der einzelnen Achsen aus.	⊙	⊙		12-9
	DRIVEA	Führt die absolute Bewegung der einzelnen Achsen aus.	⊙	⊙		12-11
	GOHOME	Bewegt den Roboter an die mit der HOME-Anweisung definierte Position (die Ausgangsposition).	⊙	⊙		12-13

4 Achsen	6 Achsen	Bildverarbei-tungsgerät	
⊙	⊙	⊙	Verfügbar mit allen Serien von Robotern und Bildverarbeitungsgeräten.
○	○	○	Verfügbar mit allen Serien von Robotern. Befehlsspezifikationen unterscheiden sich zwischen Robotern mit 4 oder 6 Achsen und dem Bildverarbeitungsgerät.
⊙	V1.2		Verfügbar mit den Robotern mit 4 Achsen oder 6 Achsen ab Version 1.2 oder neuer.

Nach Funktionen klassifiziert	Befehle	Funktionen	4 Achsen	6 Achsen	Bildverarbei- tungsgerät	Siehe:
	MOVE	Bewegt den Roboter an die angegebene Koordinate.	○	○		12-14
	ROTATE	Führt eine Rotationsbewegung um eine angegebene Achse aus.	○	○		12-18
	ROTATEH	Führt eine Drehbewegung aus, indem ein Anfahrvektor als Achse verwendet wird.	⊙	⊙		12-21
	CURJNT	Ruft den aktuellen Winkel des Roboters mit Typ J ab.	○	○		12-23
	CURPOS	Ruft die aktuelle Position in dem Werkzeugkoordinatensystem mit Typ P ab.	○	○		12-24
	CURTRN	Ruft die aktuelle Position in dem Werkzeugkoordinatensystem mit Typ T ab.	⊙	⊙		12-25
	DESTJNT	Ermittelt die Zielposition der aktuellen Bewegungsanweisung mit Typ J. Die aktuelle Position (Wert der Anweisung) wird abgerufen, wenn der Roboter stoppt.	○	○		12-26
	DESTPOS	Ermittelt die Zielposition der aktuellen Bewegungsanweisung mit Typ P. Wenn der Roboter stoppt, wird der aktuelle Wert (Wert der Anweisung) abgerufen.	○	○		12-27
	DESTTRN	Ermittelt die Zielposition der aktuellen Bewegungsanweisung mit Typ T. Wenn der Roboter stoppt, wird die aktuelle Position (Wert der Anweisung) abgerufen.	⊙	⊙		12-28
	ARRIVE	Definiert das Bewegungsverhältnis relativ zu der programmierten vollständigen Distanz bis zum Zielpunkt, um das Programm zur Ausführung des nächsten Schritts bereit zu machen, bis der Roboter das definierte Bewegungsverhältnis erzielt.	⊙	V1.2		12-29
Steuerung der Armstellung	CURFIG	Ruft den aktuellen Wert der Roboterarmstellung ab.	⊙	⊙		12-31
	FIGAPRL	Berechnet Armstellungen in einer Anfahrposition und in einer Standardposition, die für die Linearbewegung verfügbar ist.	○	○		12-33
	FIGAPRP	Berechnet eine Anfahrposition, bei der die Punkt-zu-Punkt-Bewegung verfügbar ist, und eine Armstellung einer Referenzposition.	○	○		12-35
Steuerung für Stop	HOLD	Hält die Programmverarbeitung für eine bestimmte Zeit an.	⊙	⊙		12-36
	HALT	Stoppt die Ausführung eines Programmes.	⊙	⊙		12-37

4 Achsen	6 Achsen	Bildverarbei-tungsgerät	
⊙	⊙	⊙	Verfügbar mit allen Serien von Robotern und Bildverarbeitungsgeräten.
○	○	○	Verfügbar mit allen Serien von Robotern. Befehlsspezifikationen unterscheiden sich zwischen Robotern mit 4 oder 6 Achsen und dem Bildverarbeitungsgerät.
⊙	V1.2		Verfügbar mit den Robotern mit 4 Achsen oder 6 Achsen ab Version 1.2 oder neuer.

Nach Funktionen klassifiziert	Befehle	Funktionen	4 Achsen	6 Achsen	Bildverarbei- tungsgerät	Siehe:
	INTERRUPT ON/OFF	Unterbricht eine Roboterbewegung.	⊙	⊙		12-38
Steuerung der Geschwindigkeit	SPEED	Gibt die interne Geschwindigkeit der Hand an.	⊙	⊙		12-40
	JSPEED	Gibt die Bewegungsgeschwindigkeit der Roboterachse an.	⊙	⊙		12-41
	ACCEL	Gibt die interne Beschleunigung und die interne Verzögerung an.	⊙	⊙		12-42
	JACCEL	Gibt die interne Achsenbeschleunigung und die interne Achsenverzögerung an.	⊙	⊙		12-43
	DECEL	Gibt die interne Verzögerung an.	⊙	⊙		12-44
	JDECEL	Gibt die interne Achsenverzögerung an.	⊙	⊙		12-45
	CURACC	Ruft den aktuellen internen Beschleunigungswert ab.	⊙	⊙		12-46
	CURJACC	Ruft den aktuellen internen Achsenbeschleunigungswert ab.	⊙	⊙		12-47
	CURDEC	Ruft den aktuellen internen Verzögerungswert ab.	⊙	⊙		12-48
	CURJDEC	Ruft den aktuellen internen Achsenverzögerungswert ab.	⊙	⊙		12-49
	CURJSPD	Ruft die interne Bewegungsgeschwindigkeit der momentan eingestellten Achse ab.	⊙	⊙		12-50
	CURSPD	Ruft den aktuellen Wert der internen Bewegungsgeschwindigkeit an der Hand ab.	⊙	⊙		12-51
	CUREXTACC	Ruft den aktuellen externen Beschleunigungswert ab.	V1.4	V1.4		12-52
	CUREXTDEC	Ruft den aktuellen externen Verzögerungswert ab.	V1.4	V1.4		12-53
	CUREXTSPD	Ruft den aktuellen externen Geschwindigkeitswert ab.	V1.4	V1.4		12-54
Steuerung der Zeit	DELAY	Unterbricht die Programmverarbeitung für die angegebene Dauer.	⊙	⊙	⊙	12-55
	WAIT	Stoppt die Programmverarbeitung abhängig von einer Bedingung.	⊙	⊙		12-56
Koordinaten-umwandlung	CHANGETOOL	Ändert das Werkzeugkoordinatensystem.	⊙	⊙		12-57
	CHANGEWORK	Ändert das Benutzerkoordinatensystem.	⊙	⊙		12-58
	CURTOOL	Ruft die Nummer des momentan angegebenen Werkzeugs ab.	V1.4	V1.4		12-59
	CURWORK	Ruft die Nummer der momentan angegebenen Arbeit ab.	V1.4	V1.4		12-60
Interferenzprüfung	SETAREA	Wählt den Bereich aus, in dem eine Interferenzprüfung ausgeführt wird.	⊙	⊙		12-61
	RESETAREA	Initialisiert eine Interferenzprüfung.	⊙	⊙		12-62

4 Achsen	6 Achsen	Bildverarbeitungsgerät	
⊙	⊙	⊙	Verfügbar mit allen Serien von Robotern und Bildverarbeitungsgeräten.
○	○	○	Verfügbar mit allen Serien von Robotern. Befehlsspezifikationen unterscheiden sich zwischen Robotern mit 4 oder 6 Achsen und dem Bildverarbeitungsgerät.
⊙	V1.2		Verfügbar mit den Robotern mit 4 Achsen oder 6 Achsen ab Version 1.2 oder neuer.

Nach Funktionen klassifiziert	Befehle	Funktionen	4 Achsen	6 Achsen	Bildverarbeitungsgerät	Siehe:
Steuerungsanweisungen für die Ein-/Ausgabe						
E/A-Anschluß	IN	Liest Daten aus dem E/A-Anschluß, der mit einer E/A-Variablen angegeben wurde.	⊙	⊙	⊙	13-1
	OUT	Gibt Daten an den mit einer E/A-Variablen angegebenen E/A-Anschluß aus.	⊙	⊙	⊙	13-2
	IOBLOCK ON/OFF	Führt während der Ausführung einer Bewegungsanweisung gleichzeitig eine Nicht-Bewegungsanweisung wie beispielsweise eine E/A- oder Berechnungsanweisung durch.	⊙	⊙		13-3
	SET	Stellt einen E/A-Anschluß auf ON ein.	⊙	⊙	⊙	13-5
	RESET	Stellt einen E/A-Anschluß auf OFF ein.	⊙	⊙	⊙	13-7
RS232C-Anschluß	INPUT	Ruft Daten von dem RS232C-Anschluß ab.	⊙	⊙	⊙	13-8
	LINEINPUT	Liest Daten über einen RS232C-Anschluß in einem Begrenzer ein und weist sie einer Variablen des Typs Zeichenfolge zu.	⊙	⊙	⊙	13-9
	PRINT	Gibt Daten an einen RS232C-Anschluß aus.	⊙	⊙	⊙	13-10
	WRITE	Gibt Daten an einen RS232C-Anschluß aus.	⊙	⊙	⊙	13-11
	FLUSH	Löscht den Eingabepuffer.	⊙	⊙	⊙	13-12
Programmiergerät	PRINTMSG	Zeigt eine Meldung mit einer Beschriftung und einem Symbol auf der Farb-LCD des Programmiergeräts an.	⊙	⊙		13-13
	PRINTDBG	Gibt Daten an das Debug-Fenster aus.	⊙	⊙		13-14
	BUZZER	Gibt einen Buzzer-Ton aus.	⊙	⊙		13-15
	PRINTLBL	Definiert eine Kennung (Beschriftung) für eine benutzerdefinierte Schaltfläche.	⊙	⊙		13-16
Multitasking-Steuerungsanweisungen						
Task-Steuerung	RUN	Führt gleichzeitig ein anderes Programm aus.	⊙	⊙		14-1
	KILL	Erzwingt das Beenden einer Task.	⊙	⊙		14-2
	SUSPEND	Unterbricht eine Task.	⊙	⊙		14-3
	DEFEND	Verteidigt eine Task.	⊙	⊙		14-4
	STATUS	Ruft den Programmstatus ab.	⊙	⊙		14-5
Semaphor	CREATESEM	Erstellt einen Semaphor.	⊙	⊙		14-7
	DELETESEM	Löscht einen Semaphor.	⊙	⊙		14-10
	FLUSHSEM	Gibt die auf einen Semaphor wartenden Tasks frei.	⊙	⊙		14-11
	GIVESEM	Gibt eine auf einen Semaphor wartende Task frei.	⊙	⊙		14-12
	TAKESEM	Ruft einen Semaphor mit einer angegebenen Semaphor-ID ab.	⊙	⊙		14-13

4 Achsen	6 Achsen	Bildverarbei-tungsgerät	
⊙	⊙	⊙	Verfügbar mit allen Serien von Robotern und Bildverarbeitungsgeräten.
○	○	○	Verfügbar mit allen Serien von Robotern. Befehlsspezifikationen unterscheiden sich zwischen Robotern mit 4 oder 6 Achsen und dem Bildverarbeitungsgerät.
⊙	V1.2		Verfügbar mit den Robotern mit 4 Achsen oder 6 Achsen ab Version 1.2 oder neuer.

Nach Funktionen klassifiziert	Befehle	Funktionen	4 Achsen	6 Achsen	Bildverarbei-tungsgerät	Siehe:
Arm-Semaphor	TAKEARM	Ruft die Priorität der Robotersteuerung ab, gibt eine Werkzeugkoordinate und eine Arbeitskoordinate an die Ausgangsposition zurück und stellt die interne Geschwindigkeit, Beschleunigung und Verzögerung auf 100 ein.	⊙	⊙		14-14
	GIVEARM	Gibt die Priorität der Robotersteuerung frei.	⊙	⊙		14-17
	TAKEVIS	Ruft die Priorität der visuellen Verarbeitung ab.	⊙	⊙		14-18
	GIVEVIS	Gibt die Priorität der visuellen Verarbeitung frei.	⊙	⊙		14-19
Funktionen						
Arithmetische Funktion	ABS	Ruft den absoluten Wert eines Ausdruckwertes ab.	⊙	⊙	⊙	15-1
	EXP	Ruft eine Exponentialfunktion mit einem natürlichen Logarithmus als Basis auf.	⊙	⊙	⊙	15-2
	INT	Ruft den maximalen Wert aus einem angegebenen Wert ab.	⊙	⊙	⊙	15-3
	LOG	Ermittelt einen natürlichen Loga-rithmus.	⊙	⊙	⊙	15-4
	LOG10	Ermittelt einen Zehnerlogarithmus.	⊙	⊙	⊙	15-5
	POW	Ermittelt einen Exponenten.	⊙	⊙	⊙	15-6
	MAX	Extrahiert den maximalen Wert.	⊙	⊙	⊙	15-7
	MIN	Extrahiert den minimalen Wert.	⊙	⊙	⊙	15-8
	RND	Generiert Zufallszahlen von 0 bis 1.	⊙	⊙	⊙	15-9
	SGN	Überprüft ein Vorzeichen.	⊙	⊙	⊙	15-10
	SQR	Ermittelt die Quadratwurzel.	⊙	⊙	⊙	15-11
Trigonometrische Funktion	ACOS	Ermittelt einen Arkuskosinus.	⊙	⊙	⊙	15-12
	ASIN	Ermittelt einen Arkussinus.	⊙	⊙	⊙	15-13
	ATN	Ermittelt einen Arkustangens.	⊙	⊙	⊙	15-14
	ATN2	Ermittelt den Arkustangens von Ausdruck 1, geteilt durch Ausdruck 2.	⊙	⊙	⊙	15-15
	COS	Ermittelt einen Kosinus.	⊙	⊙	⊙	15-16
	SIN	Ermittelt einen Sinus.	⊙	⊙	⊙	15-17
	TAN	Ermittelt einen Tangens.	⊙	⊙	⊙	15-18
Winkelumwandlung	DEGRAD	Wandelt die Einheit in Radianten um.	⊙	⊙	⊙	15-19
	RAD	Wandelt einen in Radianten angegebenen Wert in Grad um.	⊙	⊙	⊙	15-20
	RADDEG	Wandelt die Einheit in Grad um.	⊙	⊙	⊙	15-21
Geschwindigkeitsum-wandlung	MPS	Wandelt einen Ausdruck für die Geschwindigkeit um.	⊙	⊙		15-22
Zeitfunktion	SEC	Wandelt einen in Sekunden ausgedrückten Wert in Millisekunden um.	⊙	⊙	⊙	15-23
Vektor	AVEC	Extrahiert einen Anfahrvektor.	⊙	⊙		15-24
	OVEC	Extrahiert einen Orientierungsvektor.	⊙	⊙		15-25

4 Achsen	6 Achsen	Bildverarbeitungsgerät	
⊙	⊙	⊙	Verfügbar mit allen Serien von Robotern und Bildverarbeitungsgeräten.
○	○	○	Verfügbar mit allen Serien von Robotern. Befehlsspezifikationen unterscheiden sich zwischen Robotern mit 4 oder 6 Achsen und dem Bildverarbeitungsgerät.
⊙	V1.2		Verfügbar mit den Robotern mit 4 Achsen oder 6 Achsen ab Version 1.2 oder neuer.

Nach Funktionen klassifiziert	Befehle	Funktionen	4 Achsen	6 Achsen	Bildverarbeitungsgerät	Siehe:
	PVEC	Extrahiert einen Positionsvektor.	○	○		15-26
	MAGNITUDE	Ruft die Vektorgröße ab.	⊙	⊙		15-27
Transformation von Stellungsdaten	J2P	Wandelt Daten vom Typ Gelenk in Daten des Typs Position um.	○	○		15-28
	J2T	Wandelt Daten vom Typ Gelenk in Daten des Typs Homogene Transformation um.	○	○		15-29
	P2J	Wandelt Daten vom Typ Position in Daten des Typs Gelenk um.	○	○		15-30
	P2T	Wandelt Daten vom Typ Position in Daten des Typs Homogene Transformation um.	○	○		15-31
	T2J	Wandelt Daten vom Typ homogene Transformation in Daten vom Typ Gelenk um.	⊙	⊙		15-32
	T2P	Wandelt Daten vom Typ homogene Transformation in Daten vom Typ Position um.	⊙	⊙		15-33
	TINV	Berechnet eine inverse Matrix von Daten des Typs homogene Transformation.	⊙	⊙		15-34
Distanzextraktion	DIST	Gibt die Distanz zwischen zwei Punkten zurück.	○	○		15-35
Armstellungskomponente	FIG	Extrahiert eine Armstellung.	○	○		15-36
Winkelkomponente	JOINT	Extrahiert einen Winkel aus Koordinaten des Typs Gelenk.	○	○		15-37
Achsenkomponente	POSX	Extrahiert die X-Komponente.	○	○		15-38
	POSY	Extrahiert die Y-Komponente.	○	○		15-39
	POSZ	Extrahiert die Z-Komponente.	○	○		15-40
Rotationskomponente	POSRX	Extrahiert die X-Achsen-Rotationskomponente.		⊙		15-41
	POSRY	Extrahiert die Y-Achsen-Rotationskomponente.		⊙		15-42
	POSRZ	Extrahiert die Z-Achsen-Rotationskomponente.		⊙		15-43
	POST	Extrahiert die T-Achsen-Rotationskomponente.	⊙			15-44
Armstellungskomponente	RVEC	Extrahiert eine Armstellung.		⊙		15-45
Positionsfunktion	AREAPOS	Gibt die Mittelpunktposition und die Richtung eines Quaders mit dem Positionstyp für einen Bereich zurück, für den eine Interferenzprüfung durchgeführt wird.	⊙	⊙		15-46
	AREASIZE	Gibt die Größe (jede Seitenlänge) eines Quaders zurück, der den Bereich für die Interferenzprüfung mit dem Vektortyp definiert.	⊙	⊙		15-47
	TOOLPOS	Gibt ein Werkzeugkoordinatensystem als Positionstyp zurück.	⊙	⊙		15-48
	WORKPOS	Gibt das Benutzerkoordinatensystem als Positionstyp zurück.	⊙	⊙		15-49

4 Achsen	6 Achsen	Bildverarbei-tungsgerät	
⊙	⊙	⊙	Verfügbar mit allen Serien von Robotern und Bildverarbeitungsgeräten.
○	○	○	Verfügbar mit allen Serien von Robotern. Befehlsspezifikationen unterscheiden sich zwischen Robotern mit 4 oder 6 Achsen und dem Bildverarbeitungsgerät.
⊙	V1.2		Verfügbar mit den Robotern mit 4 Achsen oder 6 Achsen ab Version 1.2 oder neuer.

Nach Funktionen klassifiziert	Befehle	Funktionen	4 Achsen	6 Achsen	Bildverarbei-tungsgerät	Siehe:
Zeichenfolgefunktion	ASC	Führt eine Transformation in einen Zeichencode durch.	⊙	⊙	⊙	15-50
	BIN$	Wandelt den Wert eines Ausdrucks in eine binäre Zeichenfolge um.	⊙	⊙	⊙	15-51
	CHR$	Wandelt einen ASCII-Code in ein Zeichen um.	⊙	⊙	⊙	15-52
	SPRINTF$	Wandelt einen Ausdruck in ein angegebenes Format um und gibt ihn als Zeichenfolge zurück.	⊙	⊙	⊙	15-53
	HEX$	Ruft einen Wert, der von einer dezimalen in eine hexadezimale Zahl umgewandelt wurde, als Zeichenfolge ab.	⊙	⊙	⊙	15-56
	LEFT$	Extrahiert den linken Teil einer Zeichenfolge.	⊙	⊙	⊙	15-57
	LEN	Ruft die Länge einer Zeichenfolge in Byte ab.	⊙	⊙	⊙	15-58
	MID$	Extrahiert eine Zeichenfolge für die angegebene Anzahl Zeichen aus einer Zeichenfolge.	⊙	⊙	⊙	15-59
	ORD	Führt eine Transformation in einen Zeichencode durch.	⊙	⊙	⊙	15-60
	RIGHT$	Extrahiert den rechten Teil einer Zeichenfolge.	⊙	⊙	⊙	15-61
	STRPOS	Ruft die Position einer Zeichenfolge ab.	⊙	⊙	⊙	15-62
	STR$	Wandelt einen Wert in eine Zeichenfolge um.	⊙	⊙	⊙	15-63
	VAL	Wandelt eine Zeichenfolge in einen numerischen Wert um.	⊙	⊙	⊙	15-64
Konstanten						
Integrierte Konstanten	OFF	Stellt den Wert OFF (0) ein.	⊙	⊙	⊙	16-1
	ON	Stellt den Wert ON (1) ein.	⊙	⊙	⊙	16-2
	PI	Stellt einen Wert π ein.	⊙	⊙	⊙	16-3
	FALSE	Setzt einen Booleschen Wert auf falsch (0).	⊙	⊙	⊙	16-4
	TRUE	Setzt einen Booleschen Wert auf wahr (1).	⊙	⊙	⊙	16-5
Steuerung von Zeit/Datum						
Zeit/Datum	DATE$	Ruft das aktuelle Datum ab.	⊙	⊙		17-1
	TIME$	Ruft die aktuelle Uhrzeit ab.	⊙	⊙		17-2
	TIMER	Ruft die abgelaufene Zeit ab.	⊙	⊙		17-3
Fehlersteuerung						
Fehlerinformationen	ERL	Ruft die Zeilennummer ab, in der ein Fehler auftrat.	⊙	⊙		18-1
	ERR	Ruft die Nummer eines aufgetretenen Fehlers ab.	⊙	⊙		18-2
	ERRMSG$	Legt eine Fehlermeldung fest.	⊙	⊙	⊙	18-3
Fehlerunterbrechung	ON ERROR GOTO	Unterbricht die Verarbeitung, wenn ein Fehler auftritt.	⊙	⊙		18-4

4 Achsen	6 Achsen	Bildverarbei-tungsgerät	
⊙	⊙	⊙	Verfügbar mit allen Serien von Robotern und Bildverarbeitungsgeräten.
○	○	○	Verfügbar mit allen Serien von Robotern. Befehlsspezifikationen unterscheiden sich zwischen Robotern mit 4 oder 6 Achsen und dem Bildverarbeitungsgerät.
⊙	V1.2		Verfügbar mit den Robotern mit 4 Achsen oder 6 Achsen ab Version 1.2 oder neuer.

Nach Funktionen klassifiziert	Befehle	Funktionen	4 Achsen	6 Achsen	Bildverarbei-tungsgerät	Siehe:
	RESUME	Kehrt von einer Prozessunterbrechungsroutine zurück.	⊙	⊙		18-5
Systeminformationen						
System	GETENV	Ruft die Werte für die Umgebungseinstellung des Systems ab.	⊙	⊙	⊙	19-1
	LETENV	Definiert die Werte für die Umgebungseinstellung des Systems.	⊙	⊙	⊙	19-2
	VER$	Ruft die Version jedes Moduls ab.	⊙	⊙	⊙	19-3
Protokoll	STARTLOG	Startet die Aufzeichnung des Servo-Steuerprotokolls.	⊙	⊙		19-4
	CLEARLOG	Initialisiert die Aufzeichnung des Servo-Steuerprotokolls.	⊙	⊙		19-5
	STOPLOG	Stoppt die Aufzeichnung des Servo-Steuerprotokolls.	⊙	⊙		19-6
Präprozessor						
Symbolkonstanten · Makrodefinitionen	#define	Ersetzt eine angegebene Konstante oder einen Makronamen in dem Programm durch eine angegebene Zeichenfolge.	⊙	⊙	⊙	20-1
	#undef	Macht eine mit #define oder einer Makrodefinition definierte Symbolkonstante ungültig.	⊙	⊙	⊙	20-2
	#error	Erzwingt das Generieren eines Kompilierungsfehlers bei Ausführung des Befehls #error.	⊙	⊙	⊙	20-3
Dateieinbindung	#include	Bindet das Präprozessorprogramm ein.	⊙	⊙	⊙	20-4
Optimierung	#pragma optimize	Gibt die auszuführende Optimierung für jedes Programm an.	⊙	⊙	⊙	20-5
Bildverarbeitungssteuerung (Option)						
Bildeingabe und -ausgabe	CAMIN	Speichert ein Abbild von der Kamera im Bildspeicher (Prozeßfenster).	⊙	⊙	⊙	21-3
	CAMMODE	Stellt die Funktion zum Speichern eines Kameraabbildes ein.	⊙	⊙	⊙	21-4
	CAMLEVEL	Stellt die Eingangsstufe für Kamerabilder ein.	⊙	⊙	⊙	21-6
	VISCAMOUT	Zeigt ein Bild von der Kamera auf dem Monitor an.	⊙	⊙	⊙	21-7
	VISPLNOUT	Zeigt ein Abbild im Speicher des Monitors an.	⊙	⊙	⊙	21-8

4 Achsen	6 Achsen	Bildverarbeitungsgerät	
⊙	⊙	⊙	Verfügbar mit allen Serien von Robotern und Bildverarbeitungsgeräten.
○	○	○	Verfügbar mit allen Serien von Robotern. Befehlsspezifikationen unterscheiden sich zwischen Robotern mit 4 oder 6 Achsen und dem Bildverarbeitungsgerät.
⊙	V1.2		Verfügbar mit den Robotern mit 4 Achsen oder 6 Achsen ab Version 1.2 oder neuer.

Nach Funktionen klassifiziert	Befehle	Funktionen	4 Achsen	6 Achsen	Bildverarbeitungsgerät	Siehe:
	VISOVERLAY	Zeigt Zeichenbildschirm-Informationen auf dem Monitor an.	⊙	⊙	⊙	21-9
	VISDEFTABLE	Liest Abbilder von der Kamera ein und stellt die Nachschlage-Tabellendaten für die Abbildausgabe ein.	⊙	⊙	⊙	21-11
	VISREFTABLE	Verweist auf Daten in der Nachschlagetabelle.	⊙	⊙	⊙	21-13
Fenstereinstellung	WINDMAKE	Zeichnet einen Bereich für die Abbildverarbeitung.	⊙	⊙	⊙	21-14
	WINDCLR	Löscht eine eingestellte Fensterinformation.	⊙	⊙	⊙	21-19
	WINDCOPY	Kopiert Fensterdaten.	⊙	⊙	⊙	21-20
	WINDREF	Ruft Fensterinformationen ab.	⊙	⊙	⊙	21-22
	WINDDISP	Zeichnet ein angegebenes Fenster.	⊙	⊙	⊙	21-23
Zeichnen	VISSCREEN	Gibt einen Zeichnungsbildschirm an.	⊙	⊙	⊙	21-24
	VISBRIGHT	Gibt einen Helligkeitswert für die Zeichnung an.	⊙	⊙	⊙	21-26
	VISCLS	Füllt (bereinigt) eine angegebene Anzeige, die in einem Modus mit einer angegebenen Helligkeit definiert wurde.	⊙	⊙	⊙	21-27
	VISPUTP	Zeichnet einen Punkt auf dem Bildschirm.	⊙	⊙	⊙	21-29
	VISLINE	Zeichnet eine Linie auf dem Bildschirm.	⊙	⊙	⊙	21-30
	VISPTP	Zeichnet eine Linie als Verbindung zwischen zwei Punkten auf dem Bildschirm.	⊙	⊙	⊙	21-31
	VISRECT	Zeichnet ein Rechteck auf dem Bildschirm.	⊙	⊙	⊙	21-32
	VISCIRCLE	Zeichnet einen Kreis auf dem Bildschirm.	⊙	⊙	⊙	21-33
	VISELLIPSE	Zeichnet eine Ellipse auf dem Bildschirm.	⊙	⊙	⊙	21-34
	VISSECT	Zeichnet einen Sektor auf dem Bildschirm.	⊙	⊙	⊙	21-35
	VISCROSS	Zeichnet ein Kreuzsymbol auf dem Bildschirm.	⊙	⊙	⊙	21-36
	VISLOC	Kennzeichnet die Anzeigeposition der Zeichen.	⊙	⊙	⊙	21-37
	VISDEFCHAR	Gibt die Größe der Zeichen und die Anzeigemethode an.	⊙	⊙	⊙	21-39
	VISPRINT	Zeigt Zeichen und Abbildungen auf dem Bildschirm an.	⊙	⊙	⊙	21-40
Bildverarbeitung	VISWORKPLN	Gibt den zu verarbeitenden Speicher (Prozeßfenster) an.	⊙	⊙	⊙	21-41
	VISGETP	Ruft eine angegebene Koordinatenhelligkeit aus dem Speicher ab (Prozeßfenster).	⊙	⊙	⊙	21-42

4 Achsen	6 Achsen	Bildverarbeitungsgerät	
⊙	⊙	⊙	Verfügbar mit allen Serien von Robotern und Bildverarbeitungsgeräten.
○	○	○	Verfügbar mit allen Serien von Robotern. Befehlsspezifikationen unterscheiden sich zwischen Robotern mit 4 oder 6 Achsen und dem Bildverarbeitungsgerät.
⊙	V1.2		Verfügbar mit den Robotern mit 4 Achsen oder 6 Achsen ab Version 1.2 oder neuer.

Nach Funktionen klassifiziert	Befehle	Funktionen	4 Achsen	6 Achsen	Bildverarbeitungsgerät	Siehe:
	VISHIST	Ruft das Histogramm (die Helligkeitsverteilung) des Bildschirms ab.	⊙	⊙	⊙	21-43
	VISREFHIST	Liest Ergebnisse des Histogrammes.	⊙	⊙	⊙	21-44
	VISLEVEL	Ruft eine Binärisierungsstufe entsprechend dem Ergebnis des Histogrammes ab.	⊙	⊙	⊙	21-45
	VISBINA	Binärisiert die Anzeige.	⊙	⊙	⊙	21-47
	VISBINAR	Zeigt einen binärisierten Bildschirm an.	⊙	⊙	⊙	21-49
	VISFILTER	Führt eine Filterung auf der Anzeige aus.	⊙	⊙	⊙	21-50
	VISMASK	Führt Berechnungen zwischen Abbildern aus.	⊙	⊙	⊙	21-52
	VISCOPY	Kopiert die Anzeige.	⊙	⊙	⊙	21-54
	VISMEASURE	Mißt Merkmale in dem Fenster (Bereich, Schwerpunkt, Winkel der Hauptachse).	⊙	⊙	⊙	21-55
	VISPROJ	Mißt die projizierten Daten in dem Fenster.	⊙	⊙	⊙	21-58
	VISEDGE	Mißt die Kante in einem Fenster.	⊙	⊙	⊙	21-60
Codeerkennung	VISREADQR	Liest den QR-Code.	⊙	⊙	⊙	21-64
Bezeichnung	BLOB	Kennzeichnung ausführen.	⊙	⊙	⊙	21-67
	BLOBMEASURE	Führt eine Funktionsmessung der Objektkennungsnummer aus.	⊙	⊙	⊙	21-70
	BLOBLABEL	Ruft die Kennungsnummer für angegebene Koordinaten ab.	⊙	⊙	⊙	21-72
	BLOBCOPY	Kopiert eine Objektkennungsnummer.	⊙	⊙	⊙	21-74
Suchfunktion	SHDEFMODEL	Registriert das Suchmodell.	⊙	⊙	⊙	21-76
	SHREFMODEL	Bezieht sich auf die Daten des registrierten Modelles.	⊙	⊙	⊙	21-78
	SHCOPYMODEL	Kopiert ein registriertes Modell.	⊙	⊙	⊙	21-79
	SHCLRMODEL	Löscht ein registriertes Modell.	⊙	⊙	⊙	21-80
	SHDISPMODEL	Zeigt ein registriertes Modell auf dem Bildschirm an.	⊙	⊙	⊙	21-81
	SHMODEL	Sucht ein Modell.	⊙	⊙	⊙	21-82
	SHDEFCORNER	Legt die Bedingungen zum Suchen einer Ecke fest.	⊙	⊙	⊙	21-85
	SHCORNER	Sucht eine Ecke.	⊙	⊙	⊙	21-86
	SHDEFCIRCLE	Legt die Bedingung zum Suchen eines Kreises fest.	⊙	⊙	⊙	21-88
	SHCIRCLE	Sucht einen Kreis.	⊙	⊙	⊙	21-89
Abrufen von Ergebnissen	VISGETNUM	Ruft ein Ergebnis einer Abbildverarbeitung aus dem Speicher ab.	⊙	⊙	⊙	21-92
	VISGETSTR	Ruft das Ergebnis der Codeerkennung ab.	⊙	⊙	⊙	21-93
	VISPOSX	Ruft ein Ergebnis einer Abbildverarbeitung (Koordinate X) aus dem Speicher ab.	⊙	⊙	⊙	21-94

4 Achsen	6 Achsen	Bildverarbei-tungsgerät	
⊙	⊙	⊙	Verfügbar mit allen Serien von Robotern und Bildverarbeitungsgeräten.
○	○	○	Verfügbar mit allen Serien von Robotern. Befehlsspezifikationen unterscheiden sich zwischen Robotern mit 4 oder 6 Achsen und dem Bildverarbeitungsgerät.
⊙	V1.2		Verfügbar mit den Robotern mit 4 Achsen oder 6 Achsen ab Version 1.2 oder neuer.

Nach Funktionen klassifiziert	Befehle	Funktionen	4 Achsen	6 Achsen	Bildverarbei-tungsgerät	Siehe:
	VISPOSY	Ruft ein Ergebnis einer Abbildverarbeitung (Koordinate Y) aus dem Speicher ab.	⊙	⊙	⊙	21-95
	VISSTATUS	Überwacht das Verarbeitungsergebnis jeder Anweisung.	⊙	⊙	⊙	21-96
	VISREFCAL	Ruft Kalibrierungsdaten (Koordinatenumwandlung Bildverarbeitung-Roboter) ab.	⊙	⊙	⊙	21-97
PAC-Bibliothek						
Konventionelle Sprache	aspACLD	Ändert die internen Lastbedingungswerte. Zu den Lastbedingungswerten gehören die Masse der Nutzlast in Gramm (g) und der Nutzlastschwerpunkt (Angabe in mm). Geben Sie beide Werte an.	⊙	⊙		22-2
	aspChange	Wählt den internen Modus für die richtige Steuereinstellung der Bewegungsoptimierung aus.	⊙	⊙		22-4
	ndInb	Wandelt die Eingabe des angegebenen Anschlusses in eine Dezimalzahl um und behandelt sie wie eine Binärzahl.	⊙	⊙		22-5
	ndJf	Eine bedingte Verzweigung nach dem Erhalten von OK/NG von einem externen Gerät (RS232C Eingabe/Ausgabe).	⊙	⊙		22-6
	ndOnb	Wandelt eine Dezimalzahl in eine Binärzahl um und gibt sie an einen angegebenen Anschluß aus.	⊙	⊙		22-7
	ndOnbI	Wandelt eine Dezimalzahl in eine Binärzahl um und gibt sie an einen angegebenen Anschluß aus.	⊙	⊙		22-8
	ndVdt	Ruft eine Variable ab, die von einem externen Gerät übertragen wurde (RS232C Eingabe/Ausgabe).	⊙	⊙		22-9
	ndVis	Überträgt eine angegebene ganze Zahl mit 2 Ziffern an ein externes Gerät (RS232C Eingabe/Ausgabe).	⊙	⊙		22-10
	ndVput	Überträgt die Positionshaltung an ein externes Gerät (RS232C Eingabe/Ausgabe).	⊙	⊙		22-11
	ndVrst	Initialisiert ein externes Gerät (RS232C Eingabe/Ausgabe).	⊙	⊙		22-12
	ndVset	Empfängt Daten von einem externen Gerät (RS232C-Eingabe/Ausgabe).	⊙	⊙		22-13
	ndVType	Legt das Protokoll für die Kommunikation mit einem externen Gerät fest (RS232C-Ausgabe).	⊙	⊙		22-14
	ndVcom	Kommunikation mit einem externen Gerät (Kernel) (RS232C Eingabe/Ausgabe)	⊙	⊙		22-15

4 Achsen	6 Achsen	Bildverarbei-tungsgerät	
⊙	⊙	⊙	Verfügbar mit allen Serien von Robotern und Bildverarbeitungsgeräten.
○	○	○	Verfügbar mit allen Serien von Robotern. Befehlsspezifikationen unterscheiden sich zwischen Robotern mit 4 oder 6 Achsen und dem Bildverarbeitungsgerät.
⊙	V1.2		Verfügbar mit den Robotern mit 4 Achsen oder 6 Achsen ab Version 1.2 oder neuer.

Nach Funktionen klassifiziert	Befehle	Funktionen	4 Achsen	6 Achsen	Bildverarbei-tungsgerät	Siehe:
	ndApra	Führt eine absolute Operation mit dem angegebenen Werkzeugkoordinatensystem aus (ausschließlich für Roboter mit 4 Achsen).	⊙			22-16
	ndDepa	Führt eine absolute Operation mit dem angegebenen Werkzeugkoordinatensystem aus (ausschließlich für Roboter mit 4 Achsen).	⊙			22-17
	ndTC	Stellt die WM-Zeit ein.	⊙	V1.2		4-23
	ndTS	Legt die Dauer der WM-Zeit fest und die langsame Geschwindigkeit.	⊙	V1.2		4-23
Palettierung	pltDecCnt	Verringert die Zahl im Palettierungsgesamtzähler.	⊙	⊙		22-18
	pltGetCnt	Ruft den Inhalt des Palettierungsgesamtzählers ab.	⊙	⊙		22-19
	pltGetK	Ruft den Wert K des Palettierungssatzes ab.	⊙	⊙		22-20
	pltGetK1	Ruft den Inhalt des Palettierungszählers K1 ab.	⊙	⊙		22-21
	pltGetM	Ruft den Wert M des Palettierungssatzes ab.	⊙	⊙		22-22
	pltGetM1	Ruft den Inhalt des Palettierungszählers M1 ab.	⊙	⊙		22-23
	pltGetN	Ruft den Wert N des Palettierungssatzes ab.	⊙	⊙		22-24
	pltGetN1	Ruft den Inhalt des Palettierungszählers N1 ab.	⊙	⊙		22-25
	pltGetNextPos	Ruft die nächste Position ab.	⊙	⊙		22-26
	pltGetPLT1END	Ruft eine Markierung Palettierung 1. Zeile ab.	⊙	⊙		22-27
	pltGetPLTEND	Ruft eine Markierung Palettierung alle Zeilen ab.	⊙	⊙		22-28
	pltIncCnt	Erhöht den Inhalt des Palettierungsgesamtzählers.	⊙	⊙		22-29
	pltInit1	Palettierungsinitialisierung Schablone 1.	⊙	⊙		22-30
	pltInitialize	Palettierungsinitialisierung	⊙	⊙		22-31
	pltKernel	Palettierungsbewegung (Kernel)	⊙	⊙		22-32
	pltLetCnt	Definiert den Inhalt des Palettierungsgesamtzählers.	⊙	⊙		22-33
	pltLetK1	Definiert den Inhalt des Palettierungszählers K1.	⊙	⊙		22-34
	pltLetM1	Definiert den Inhalt des Palettierungszählers M1.	⊙	⊙		22-35
	pltLetN1	Definiert den Inhalt des Palettierungszählers N1.	⊙	⊙		22-36
	pltMain1	Palettierung Schablone 1	⊙	⊙		22-37
	pltMain2	Palettierung Schablone 2	⊙	⊙		22-38
	pltMove	Standardpalettierung Schablone 1	⊙	⊙		22-39

4 Achsen	6 Achsen	Bildverarbeitungsgerät	
⊙	⊙	⊙	Verfügbar mit allen Serien von Robotern und Bildverarbeitungsgeräten.
○	○	○	Verfügbar mit allen Serien von Robotern. Befehlsspezifikationen unterscheiden sich zwischen Robotern mit 4 oder 6 Achsen und dem Bildverarbeitungsgerät.
⊙	V1.2		Verfügbar mit den Robotern mit 4 Achsen oder 6 Achsen ab Version 1.2 oder neuer.

Nach Funktionen klassifiziert	Befehle	Funktionen	4 Achsen	6 Achsen	Bildverarbeitungsgerät	Siehe:
	pltMove0	Standardpalettierung Bewegung 1	⊙	⊙		22-40
	pltResetAll	Setzt alle Palettierungszähler zurück.	⊙	⊙		22-41
	pltResetPLT1END	Setzt eine Markierung Palettierung 1. Zeile zurück.	⊙	⊙		22-42
	pltResetPLTEND1	Setzt eine Markierung Palettierung alle Zeilen zurück.	⊙	⊙		22-43
Werkzeugbedienung	tolChange	Wechsel des Werkzeugs	⊙	⊙		22-44
	tolInit1	Werkzeugänderung Initialisierung Schablone 1	⊙	⊙		22-45
	tolInitialize	Werkzeugänderung Initialisierung	⊙	⊙		22-46
	tolKernel	Werkzeugänderung Bewegung (Kernel)	⊙	⊙		22-47
	tolMain1	Werkzeugänderung Initialisierung Schablone 1	⊙	⊙		22-48
Ein-/Ausgabe	dioSync	Führt eine Synchronisation mit einem an DIO angeschlossenen externen Gerät (z. B. einem Sequenzierer) durch.	⊙	⊙		22-49
Armbewegung	mvResetPulseWidth	Stellt die Impulswerte des Standardcodierers für die zulässige Positionsabweichung wieder her.	⊙	⊙		22-50
	mvResetTimeOut	Stellt den Zeitüberschreitungswert für das Ende der Standardbewegung wieder her.	⊙	⊙		22-51
	mvReverseFlip	Umkehrung der Armstellung bei 4 Achsen	⊙	⊙		22-52
	mvSetPulseWidth	Definiert die zulässige Breite des Stopimpulses.	⊙	⊙		22-53
	mvSetTimeOut	Definiert den Zeitüberschreitungswert für das Ende der Bewegung.	⊙	⊙		22-54
	SetGravity	Kompensiert die auf die einzelnen Gelenke angewendete statische Last (Schwerkraft-Drehmoment) und erzielt eine Balance mit dem Schwerkraft-Drehmoment.		V1.2		22-55
	ResetGravity	Deaktiviert die Balanceeinstellung zwischen dem begrenzten Motordrehmoment und dem Schwerkraftdrehmoment, die mit SetGravity definiert wurde.		V1.2		22-56
	SetGrvOffset	Kompensiert das Drehmoment jedes Gelenkes, das mit SetGravity für das Schwerkraftdrehmoment programmiert wurde.		V1.2		22-57
	ResetGrvOffset	Deaktiviert die Schwerkraft-Offsetfunktion.		V1.2		22-58
	SetCurLmt	Legt den Grenzwert für den Motorstrom fest, der auf die angegebene Achse angewendet werden soll.	○	V1.2		22-59

4 Achsen	6 Achsen	Bildverarbei-tungsgerät	
⊙	⊙	⊙	Verfügbar mit allen Serien von Robotern und Bildverarbeitungsgeräten.
○	○	○	Verfügbar mit allen Serien von Robotern. Befehlsspezifikationen unterscheiden sich zwischen Robotern mit 4 oder 6 Achsen und dem Bildverarbeitungsgerät.
⊙	V1.2		Verfügbar mit den Robotern mit 4 Achsen oder 6 Achsen ab Version 1.2 oder neuer.

Nach Funktionen klassifiziert	Befehle	Funktionen	4 Achsen	6 Achsen	Bildverarbei-tungsgerät	Siehe:
	ResetCurLmt	Setzt den Motorstromgrenzwert für die angegebene Achse zurück.	⊙	V1.2		22-61
	SetEralw	Ändert die zulässige Abweichung für die angegebene Achse.	⊙	V1.2		22-62
	ResetEralw	Setzt den Wert für die zulässige Abweichung der angegebenen Achse auf den Anfangswert zurück.	⊙	V1.2		22-63
	OnSrvLock	Führt eine Servo-Sperre einer angegebenen Achse aus (ausschließlich für Roboter mit 4 Achsen).	⊙			22-64
	OffSrvLock	Gibt die Servo-Sperre für die angegebene Achse frei. (Ausschließlich für Roboter mit 4 Achsen)	⊙			22-65
	OnPWM	Führt eine PWM-Umschaltung an einer angegebenen Achse aus (ausschließlich für Roboter mit 4 Achsen).	⊙			22-66
	OffPWM	Gibt die PWM-Umschaltung an der angegebenen Achse frei (ausschließlich für Roboter mit 4 Achsen).	⊙			22-67
	Setcycloid	Bewirkt einen Wechsel in den Zykloid-Betriebsmodus zur Unterdrückung von Überschuß und Restvibration am Ende der Punkt-zu-Punkt-Operation.	⊙	V1.4		22-68
	Resetcycloid	Bewirkt einen Übergang vom Zykloidmodus in den normalen Betriebsmodus.	⊙	V1.4		22-69
	SetForce_HM	Eine Strombegrenzungsbibliothek, die den Schub (Einheit: N) der Z-Koordinate mit dem HM/HS-Roboter angibt.	⊙			22-70
	SetForce_HC	Eine Strombegrenzungsbibliothek, die den Schub (Einheit: N) der Z-Koordinate mit dem HC-Roboter angibt.	⊙			22-71
	SetCompControl	Aktiviert eine Funktion zur Nachgiebigkeitssteuerung.		V1.4		22-72
	SetCompFControl	Aktiviert eine Funktion zur Nachgiebigkeitssteuerung. Die Verarbeitung zur Kompensierung des Schwerkraft-Offset ist jedoch nicht aktiv.		V1.4		22-73
	ResetCompControl	Deaktiviert eine Funktion zur Nachgiebigkeitssteuerung.		V1.4		22-74
	SetFrcCoord	Wählt ein Koordinatensystem für die Einstellung der Kraftbegrenzung aus.		V1.4		22-75
	SetFrcLimit	Legt die Rate der Kraftbegrenzung fest.		V1.4		22-76
	ResetFrcLimit	Initialisiert die Rate der Kraftbegrenzung.		V1.4		22-77

4 Achsen	6 Achsen	Bildverarbei-tungsgerät	
⊙	⊙	⊙	Verfügbar mit allen Serien von Robotern und Bildverarbeitungsgeräten.
○	○	○	Verfügbar mit allen Serien von Robotern. Befehlsspezifikationen unterscheiden sich zwischen Robotern mit 4 oder 6 Achsen und dem Bildverarbeitungsgerät.
⊙	V1.2		Verfügbar mit den Robotern mit 4 Achsen oder 6 Achsen ab Version 1.2 oder neuer.

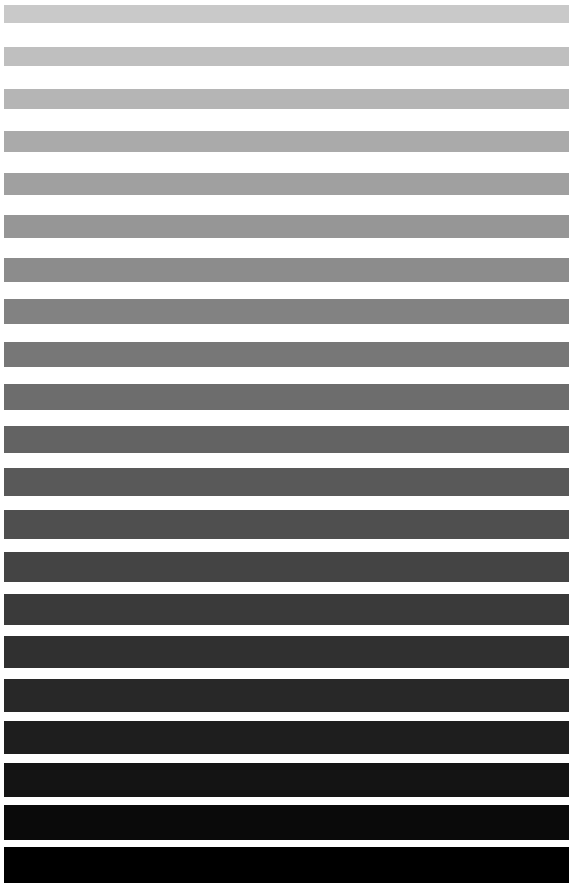
Nach Funktionen klassifiziert	Befehle	Funktionen	4 Achsen	6 Achsen	Bildverarbei-tungsgerät	Siehe:
	SetCompRate	Legt die Nachgiebigkeitsrate unter der Nachgiebigkeitssteuerung fest.		V1.4		22-78
	ResetCompRate	Initialisiert die Nachgiebigkeitsrate unter der Nachgiebigkeitssteuerung.		V1.4		22-79
	SetFrcAssist	Legt die Kraftunterstützung unter der Nachgiebigkeitssteuerung fest.		V1.4		22-80
	ResetFrcAssist	Initialisiert die Kraftunterstützung unter der Nachgiebigkeitssteuerung.		V1.4		22-81
	SetCompJLimit	Legt den aktuellen Grenzwert unter der Nachgiebigkeitssteuerung fest.		V1.4		22-82
	ResetCompJLimit	Initialisiert den aktuellen Grenzwert unter der Nachgiebigkeitssteuerung.		V1.4		22-83
	SetCompVMode	Legt den Geschwindigkeits-Steuermodus unter der Nachgiebigkeitssteuerung fest.		V1.4		22-84
	ResetCompVMode	Deaktiviert den Geschwindigkeits-Steuermodus unter der Nachgiebigkeitssteuerung.		V1.4		22-85
	SetCompEralw	Gibt die Werte für die zulässige Abweichung der Position und der Haltung der Tool-Spitze unter der Nachgiebigkeitssteuerung an.		V1.4		22-86
	ResetCompEralw	Initialisiert die Werte für die zulässige Abweichung der Position und die Haltung der Tool-Spitze unter der Nachgiebigkeitssteuerung.		V1.4		22-87
	SetDampRate	Legt die Dämpfungsrate unter der Nachgiebigkeitssteuerung fest.		V1.4		22-88
	ResetDampRate	Initialisiert die Dämpfungsrate unter der Nachgiebigkeitssteuerung.		V1.4		22-89
	SetVibControl	Legt den Steuermodus für die Reduzierung der Restvibration fest.		V1.4		22-90
	ResetVibControl	Kehrt vom Steuermodus zur Reduzierung der Restvibration zurück in den normalen Steuermodus.		V1.4		22-91
Bildverarbeitung	viTran6	Wandelt die Bildverarbeitungskoordinaten in Roboterkoordinaten um (für 6 Achsen).	⊙	⊙		22-92

TEIL 1

PROGRAMMKONZEPTION

Kapitel 1

Musterprogramm



Diese Kapitel verdeutlicht die Verwendung der einzelnen Befehle anhand eines einfachen Musterprogramms.

1.1 Modellfall-Anwendung

Dieser Abschnitt beschreibt ein Musterprogramm durch Verwendung einer Anwendung als Modellfall, wie in Abb. 1-1 dargestellt.

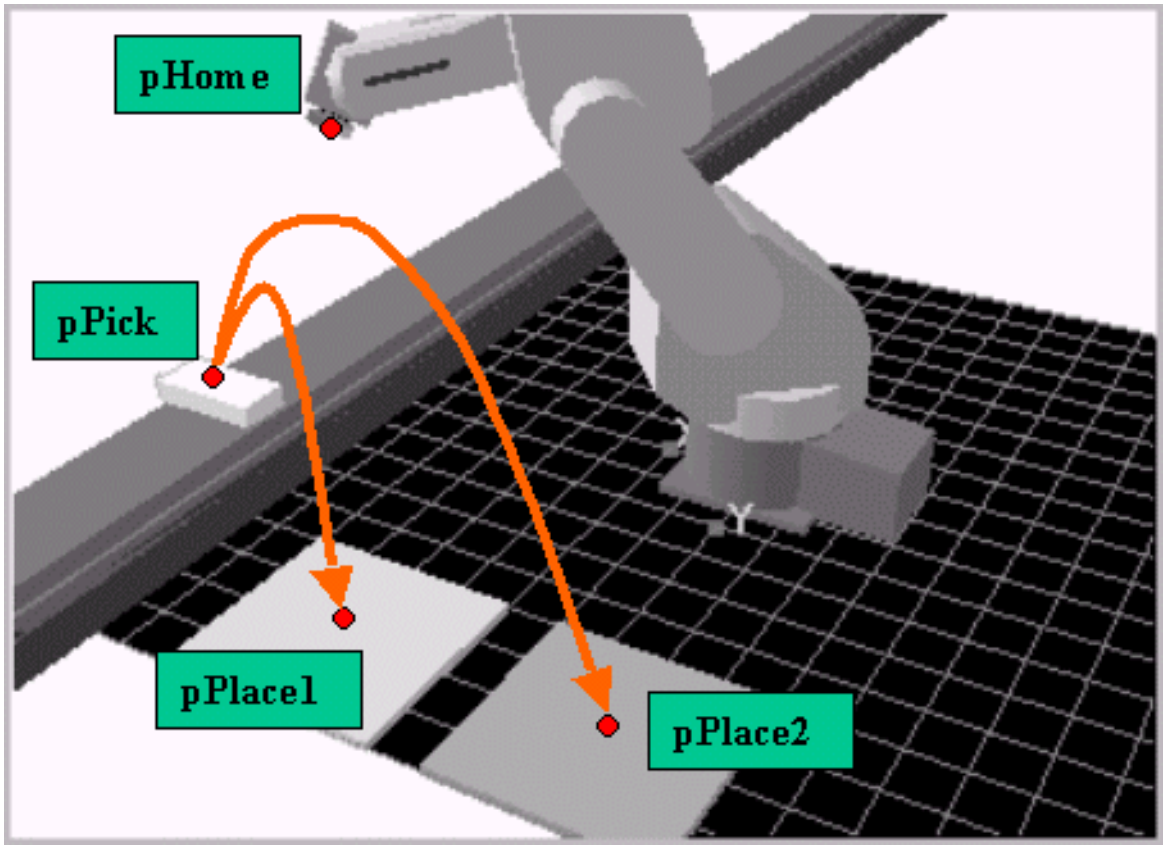


Abb. 1-1 Modellfall-Anwendung

In diesem Modellfall liest der Roboter den QR-Code in dem Objekt an pPick. Der Roboter ermittelt anhand des QR-Codes, an welche der Positionen pPlace1 und pPlace2 das Objekt transportiert werden soll. Anschließend greift er das Objekt auf und transportiert es an jede der vordefinierten Positionen und legt es dort ab. pHome ist die Ausgangsposition oder die Bezugspunktposition.

Anmerkung:	**Anleitungen zum Erstellen eines Programmes finden Sie in Teil 2, “Erstellen eines Programmes auf einem PC mit WINCAPSII” im EINFÜHRUNGSHANDBUCH.** **Das Kapitel bietet eine Erläuterung der Prozedur zum Erstellen eines Programmes mit der gleichen Modellfall-Anwendung.**

1.2 Programmablauf

Der Programmablauf des Modellfalls ist in Abb. 1-2 dargestellt.

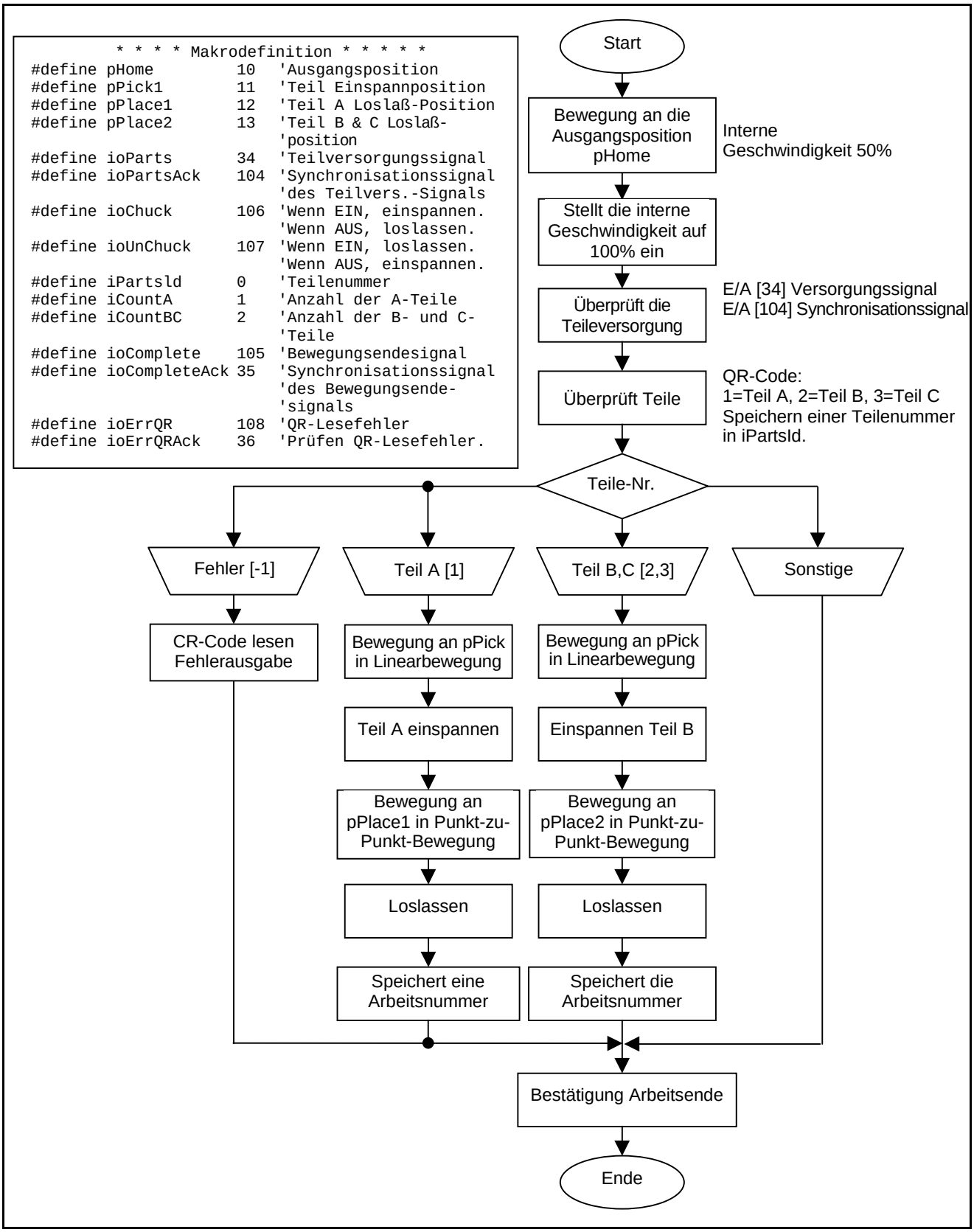


Abb. 1-2 Modellfall-Programmablauf

1.3 Programmliste

Nachfolgend finden Sie eine Programmliste für den Modellfall.
Es gibt vier Programme: “PRO1,” “PRO2,” “dioSetAndWait” und “dioWaitAndSet.”
“PRO1” ist das Hauptprogramm.
“PRO2” ist das Unterprogramm zum Lesen des QR-Codes.
“dioSetAndWait” und “dioWaitAndSet” sind Unterprogramme für die E/A-Operationen zur Prüfung der Teileversorgung. Diese beiden Programme sind in der Programmbank des PAC Program Manager gespeichert.

Programmcodeliste “PRO1”

```
'!TITLE “Aufnehmen & Plazieren”
#INCLUDE  “dio_tab.h”                                  'Liest eine DIO-Makrodefinitionsdatei ein.
#INCLUDE  “var_tab.h”                                  'Liest eine variable Makrodefinitionsdatei.
#DEFINE   appLen      100                              'Definiert eine Annäherungslänge.
          appLen und 100 sind funktional gleichwertig.

PROGRAM pro1  Deklariert einen Programmnamen.

   TAKEARM    Anweisung zur Ausführung des Roboters.   'Ermittelt einen Arm-Semaphor.

   MOVE P, P[pHome], S=50                              'Bewegung an die Ausgangsposition von 50% der
              Entspricht Makro P[10] oder P10.         'internen Geschwindigkeit bei der Punkt-zu-
                                                       'Punkt-Bewegung.

   SPEED 100                                           'Stellt die interne Geschwindigkeit auf 100% ein.
   CALL dioWaitAndSet(ioParts, ioPartsAck)             'Überprüft die Teileversorgung.
        Ruft ein Programm auf.

   CALL pro2                                           'Liest den QR-Code.
   SELECT CASE I[iPartsId]  Anweisung, die einen Anweisungsblock bedingt ausführt in
                  Makro     Abhängigkeit von I[○○] und die Steuerung übergibt.
     CASE -1 Wenn I[○○] -1 ist, wird folgendes ausgeführt.   'Gegenmessung zu Lesefehlern
                                                             'beim QR-Code.
       CALL dioSetAndWait(ioErrQR, ioErrQRAck)           'Gibt einen Fehler aus.
     CASE 1
       GOSUB *PlacePartsA                                'Verarbeitet Teil A.
       Ruft ein Unterprogramm auf.

     CASE 2, 3
       GOSUB *PlacePartsBC                               'Verarbeitet Teile B und C.
   END SELECT
   CALL dioSetAndWait(ioComplete,ioCompleteAck)          'Gibt das Signal für das Bewegungsende aus.
   GIVEARM Gibt den Arm-Semaphor frei, der mit TAKEARM   'Gibt den Arm-Semaphor frei.
           deklariert wurde.
END
```

Fortsetzung folgt auf der nächsten Seite.

Programmcodeliste “PRO1”(Fortsetzung)

```
' ===== Teil einspannen =====
*ChuckItem: *Gibt eine Kennung an mit ○○:. Wird später aufgerufen von CALL*ChuckItem.
    SET IO[ioChuck] Entspricht IO[106].
            Makro
      Schaltet IO ein.
    RESET IO[ioUnChuck] Entspricht IO[106].
                Makro
      Schaltet IO aus.

    RETURN Anweisung, die ein Unterprogramm mit *ChuckItem strukturiert.


 ' ===== Teile ausspannen =====
 *UnchuckItem: Kennung
    RESET IO[ioChuck]
    SET IO[ioUnChuck]
    RETURN Anweisung

 ' ===== Verarbeitung Teil A =====
 *PlacePartsA: Kennungsname (deklariert ein Unterprogramm)
    APPROACH P, P[pPick], appLen                    'Bewegung an eine Position 100 mm entfernt vom
    Bewegt den Arm  Entspricht   Entspricht dem in der 4. Zeile   'Einspannpunkt in Linearbewegung.
    an einen Punkt  Makro P[11]. definierten Makro 100. Durch
    direkt über                  Definition auf diese Weise kann die
    P[pPick1].                   Annäherungslänge durch Änderung
                                 eines einzigen Wertes geändert
    MOVE L, P[pPick], S=80 werden                   'Bewegung an einen Einspannpunkt bei 80% der
      Anweisung.                                    'internen Geschwindigkeit bei Linearbewegung.

    GOSUB *ChuckItem                                'Spannt ein Teil ein.
    DEPART L, appLen                                'Bewegung an eine Position 100 mm entfernt von der
    Anweisung.   Entspricht dem in der 4. Zeile definierten Makro   'aktuellen Position.
                 100. Durch Definition auf diese Weise kann die
                 Annäherungslänge durch Änderung eines
                 einzigen Wertes geändert werden.

    APPROACH P, P[pPlace1], appLen                  'Bewegung an eine Position 100 mm entfernt von der
    Bewegt den Arm an einen   Entspricht dem in der 4. Zeile   'Loslaß-Position bei Linearbewegung.
    Punkt direkt über P[pPick1]. definierten Makro 100. Durch
                                 Definition auf diese Weise kann
                                 die Annäherungslänge durch
                                 Änderung eines einzigen Wertes
                                 geändert werden.
    MOVE L, P[pPlace1], S=50                        'Bewegung an den Loslaß-Punkt bei 50% der
      Anweisung.                                    'internen Geschwindigkeit bei Linearbewegung.

    GOSUB *UnchuckItem                              'Läßt das Teil los.
    DEPART L, appLen                                'Bewegung an eine Position 100 mm entfernt von der
    Anweisung.   Entspricht dem in der 4. Zeile definierten Makro   'aktuellen Position bei Linearbewegung.
                 100. Durch Definition auf diese Weise kann die
                 Annäherungslänge durch Änderung eines
                 einzigen Wertes geändert werden.

    I[iCountA] = I[iCountA] + 1                     'Zählt die Anzahl der Teile A.
         Erhöht I[iCountA] um eins.
 RETURN
```

Fortsetzung folgt auf der nächsten Seite.

Programmcodeliste “PRO1”(Fortsetzung)

```
' ===== Verarbeitung Teile B und C =====
*PlacePartsBC: Kennungsname (deklariert ein Unterprogramm)
   APPROACH P, P[pPick], appLen                'Bewegung an eine Position 100 mm entfernt vom
   Bewegt den Arm an einen Punkt direkt über P[pPick1].   Entspricht dem in der 4. Zeile definierten Makro 100. Durch Definition auf diese Weise kann die Annäherungslänge durch Änderung eines einzigen Wertes geändert werden.
                                               'Einspannpunkt in Linearbewegung.
   MOVE L, P[pPick], S=90                      'Bewegung an einen Einspannpunkt bei 90% der
                                               'internen Geschwindigkeit in Linearbewegung.
   GOSUB *ChuckItem                            'Spannt ein Teil ein.
   DEPART L, appLen                            'Bewegung an eine Position 100 mm entfernt von
   Anweisung.   Entspricht dem in der 4. Zeile definierten Makro 100. Durch Definition auf diese Weise kann die Annäherungslänge durch Änderung eines einzigen Wertes geändert werden.
                                               'der aktuellen Position.
   APPROACH P, P[pPlace2], appLen              'Bewegung an eine Position 100 mm entfernt von der
   Bewegt den Arm an einen Punkt direkt über P[pPick1].   Entspricht dem in der 4. Zeile definierten Makro 100. Durch Definition auf diese Weise kann die Annäherungslänge durch Änderung eines einzigen Wertes geändert werden
                                               'Loslaß-Position bei Linearbewegung.
   MOVE L, P[pPlace2], S=80                    'Bewegung an den Loslaß-Punkt bei 80% der
        Anweisung.                             'interne Geschwindigkeit bei Linearbewegung.
   GOSUB *UnchuckItem                          'Läßt das Teil los.
   DEPART L, appLen                            'Bewegung an eine Position 100 mm entfernt von
   Anweisung.   Entspricht dem in der 4. Zeile definierten Makro 100. Durch Definition auf diese Weise kann die Annäherungslänge durch Änderung eines einzigen Wertes geändert werden.
                                               'der aktuellen Position bei Linearbewegung.
   I[iCountBC] = I[iCountBC] + 1               'Zählt die Anzahl der Teile B und C.
RETURN
```

Fortsetzung folgt auf der nächsten Seite.

Programmcodeliste “PRO2”

```
'!TITLE “QR-Code lesen”
#INCLUDE “var_tab.h”                                  'Liest die variable Makrodefinitionsdatei.

'Speichert eine Teilenummer in [iPartsId].
PROGRAMM pro2
   DIM len1 AS INTEGER
                Macht eine Deklaration einer lokalen Variablen "lenl" als lokale ganzzahlige Variable verwendbar.

   TAKEVIS Deklariert die Steuerung am Bildverarbeitungsgerät.   'Ruft den visuellen Semaphor ab.
           Zur Verwendung des Bildverarbeitungsgeräts
           erforderlich.
   VISSCREEN 1,0,1                                    'Gibt den Zeichnungsbildschirm 0 für die Zeichnung
                                                      'an Ziel.
   VISCLS 0                                           'Löscht den Zeichnungsbildschirm 0.

   VISOVERLAY 1                                       'Zeigt den Zeichnungsbildschirm 0.

   CAMIN 1,0,0                                        'Ruft das Abbild auf der Kamera in den
                                                      'Prozeßbildschirm 0 ab.
   VISPLNOUT 0,1                                      'Zeigt den Prozeßbildschirm 0 an.

   VISREADQR 1,WINDREF(1, 2),WINDREF                  'Liest den QR-Code.
    (1, 3),0,0,1

   WINDDISP 1                                         'Zeigt das Fenster an.

   VISLOC 10, 10                                      'Gibt die anzuzeigende Position an.

   IF VISSTATUS(0) = 0 THEN                           'Wenn das Messungsergebnis OK ist.
     len1 = VISGETNUM(0, 0)                           'Ruft die Anzahl der gemessenen Zeichen ab.
     VISPRINT VISGETSTR(1, len1)                      'Zeigt das Messungsergebnis an.
     I[iPartsId] = VAL(VISGETSTR(1, len1))            'Speichert die Teilenummer.

   ELSE
     VISPRINT “Kann einen Code nicht erkennen.”       'Zeigt einen Fehler an.
     I[iPartsId] = -1                                 'Speichert – 1 wenn ein Fehler auftritt.

   ENDIF
   Bedingte Verzweigung.
   GIVEVIS Gibt die Steuerung des Bildverarbeitungsgeräts frei.   'Gibt den visuellen Semaphor frei.
END
```

Anmerkung: Prozeßfenster 1 zum Lesen des QR-Codes muß bereits definiert worden sein.

Programmcodeliste "WAIT → SET" (Bibliothek)

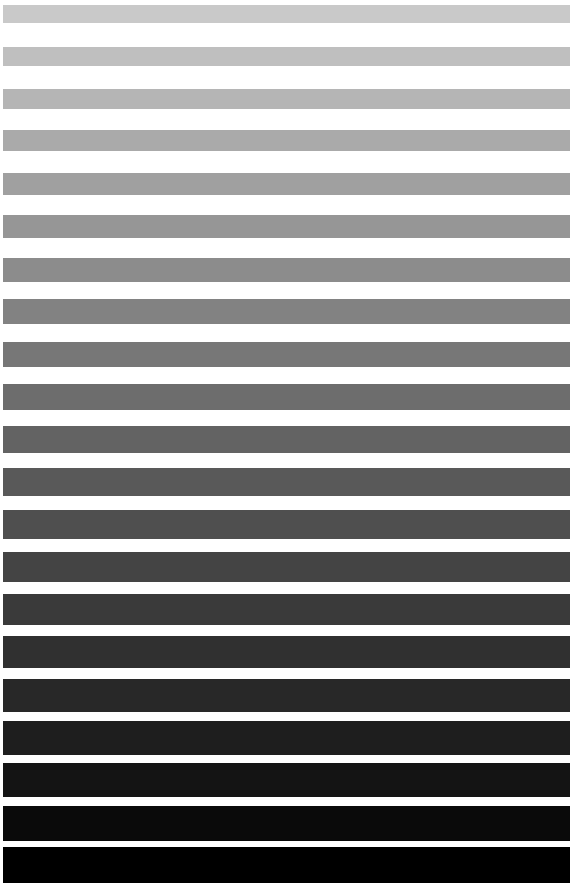
```
'!TITLE "WAIT → SET"
PROGRAM dioWaitAndSet(waitIndex%, ackIndex%)
                      Deklaration mit Argument.
   RESET IO[ackIndex] Anweisung.                    'Synchronisationssignal AUS

   WAIT IO[waitIndex] = ON Anweisung, die auf IO□   'Wartet auf Synchronisationssignal EIN.
                           wartet.

   SET IO[ackIndex]                                 'Synchronisationssignal EIN

END
```

Programmcodeliste "SET → WAIT" (Bibliothek)

```
'!TITLE "SET → WAIT"
PROGRAM dioSetAndWait(ackIndex%, waitIndex%)
                      Deklaration mit Argument.

   SET IO[ackIndex]                                 'Synchronisationssignal EIN

   WAIT IO[waitIndex] = ON Anweisung, die auf IO□   'Wartet auf Synchronisationssignal EIN.
                           wartet.
   RESET IO[ackIndex] Anweisung                     'Synchronisationssignal AUS

END
```

Kapitel 2

Programmablauf



Dieses Kapitel beschreibt die Ablaufregulierung zum Erstellen von Programmen mit der PAC-Sprache.

2.1 Aufrufen eines Programmes und eines Unterprogrammes

Ein Programmabschnitt, der eine bestimmte Bewegung ausführt, kann aus dem Programm herausgenommen und, falls erforderlich, aufgerufen werden.
Das Anlegen dieses Abschnittes im gleichen Programm wird als Unterprogramm (Unterroutine) bezeichnet. Wird dieser Abschnitt gesondert in eine separate Datei als anderes Programm eingefügt und aufgerufen, wird dies als Aufrufen eines Programmes bezeichnet.
Eine Unterroutine (Unterprogramm) muß in der gleichen Datei vorhanden sein wie das aufrufende Programm.
Das Programm in einer gesonderten Datei kann von verschiedenen Programmen aus aufgerufen und allgemein verwendet werden.
Wenn eine Arbeitsabfolge als Einheit einer Unterroutine (Unterprogramm) oder eines anderen Programmes angesehen wird, muß der gleiche Inhalt nicht noch einmal beschrieben werden. Dies erweist sich als wirkungsvoll beim Korrigieren von Beschreibungen, beim Verringern der Bearbeitungszeit sowie wenn es darum geht, das Lesen eines Programmes zu verbessern und zu erleichtern.

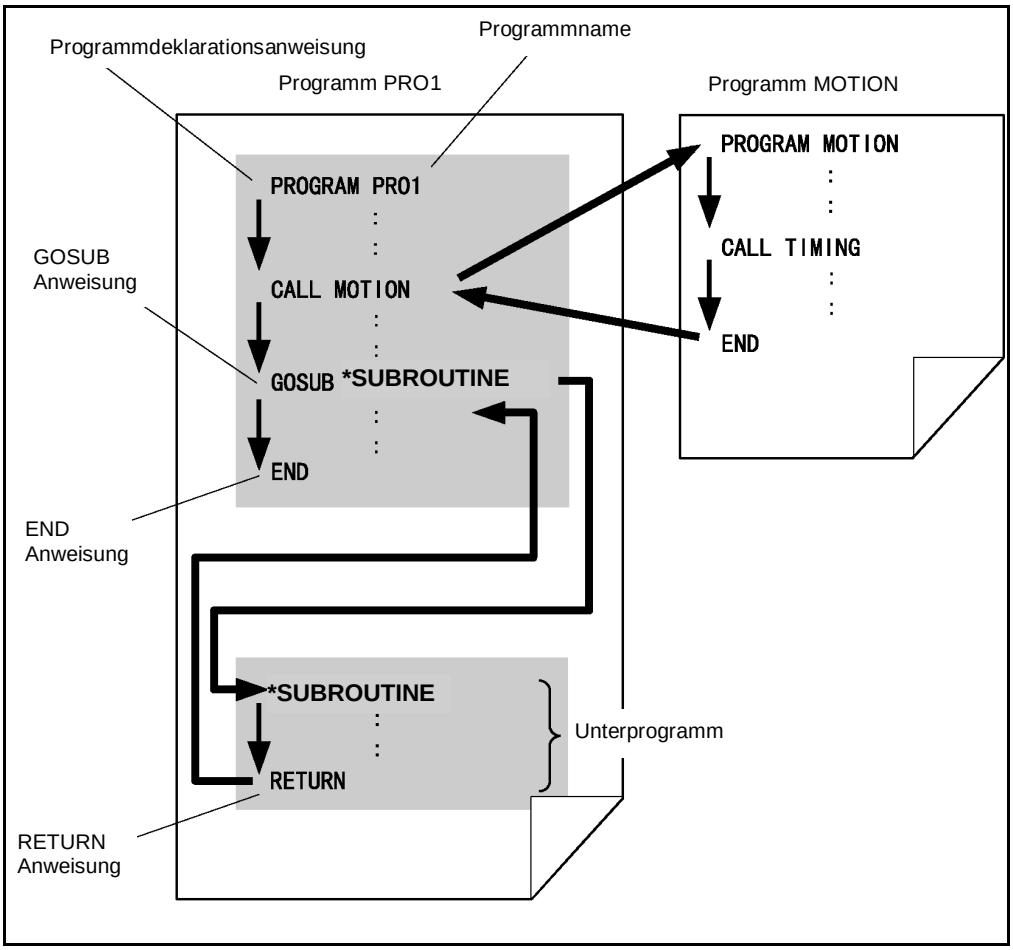


Abb. 2-1 Unterschied zwischen dem Aufrufen eines Programmes und dem Aufrufen einer Unterroutine (Unterprogramm)

2.1.1 Aufrufen einer Unterroutine

Um das gleiche Programm in verschiedenen Positionen innerhalb eines Programmes zu verwenden, beschreiben Sie den Prozeß als Unterroutine (Unterprogramm). Durch Aufrufen aus verschiedenen Positionen kann die Unterroutine verwendet werden.
Eine Unterroutine (Unterprogramm) muß in der gleichen Datei beschrieben sein, wie das aufrufende Programm.
Wenn eine Unterroutine (Unterprogramm) mit der GOSUB-Anweisung aufgerufen wird, bewegt sich die Steuerung zur Unterroutine. Wenn die Steuerung in der letzten Zeile der Unterroutine (Unterprogramm) die RETURN-Anweisung ausführt, kehrt sie zur nächsten Zeile des Programmes zurück, das die Unterroutine (Unterprogramm) aufgerufen hat.
Eine Unterroutine (Unterprogramm) kann auch von einer anderen Unterroutine aus aufgerufen werden.
Da die Unterroutine (Unterprogramm) sich in der gleichen Datei befindet wie das aufrufende Programm, können lokale Variablen von beiden Programmen gemeinsam verwendet werden. Deshalb müssen Variablen nicht übertragen werden, wenn die Unterroutine (Unterprogramm) aufgerufen wird. Sie kann jedoch nicht von anderen Dateien aus oder direkt vom Programmier- oder einem anderen externen Gerät aufgerufen werden.

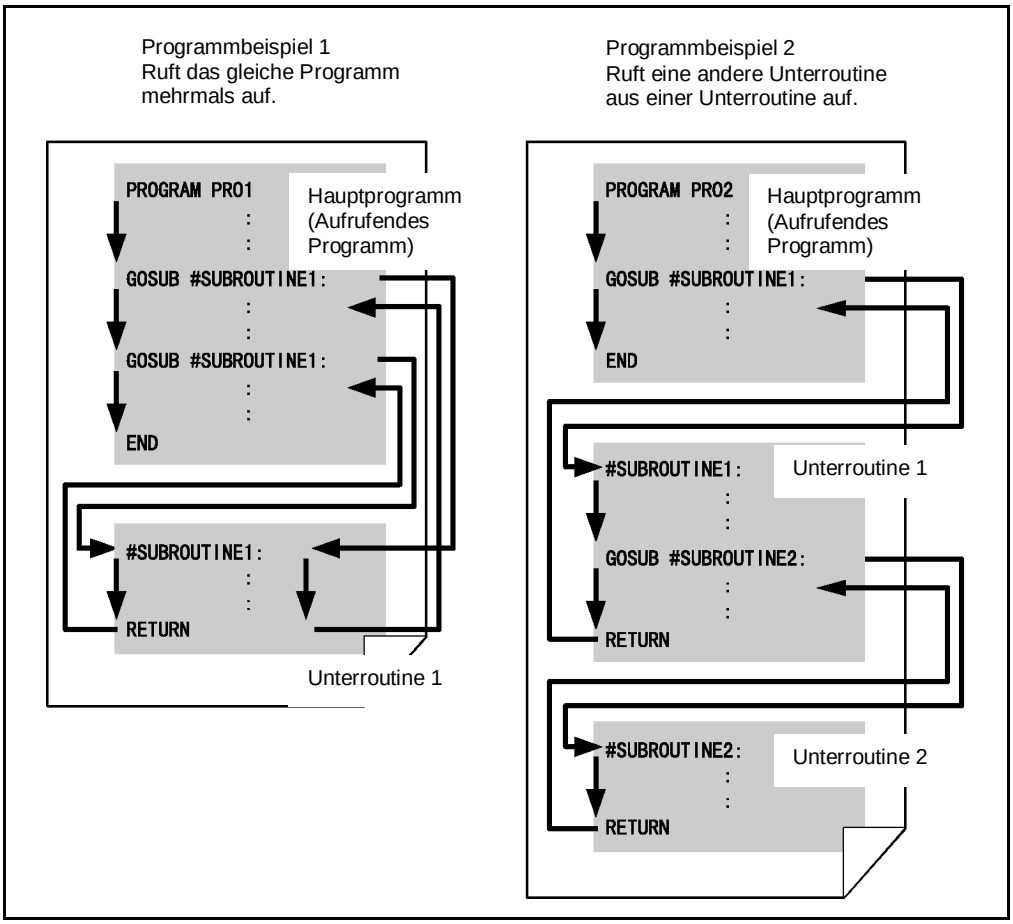


Abb. 2-2 Aufrufen einer Unterroutine

2.1.2 Aufrufen eines Programmes

Wenn ein Programm getrennt vom Hauptprogramm erstellt wird, kann es verwendet werden, indem es wie eine Unterroutine (Unterprogramm) aufgerufen wird.
Beim Aufrufen des Programmes geben Sie den Programmnamen mit der CALL-Anweisung an.
Wenn die CALL-Anweisung ein Programm aufruft, bewegt sich die Steuerung zum aufgerufenen Programm. Wenn die Steuerung die END-Anweisung in der letzten Zeile des aufgerufenen Programmes ausführt, kehrt sie zur nächsten Zeile im aufgerufenen Programm zurück.
Das aufgerufene Programm kann auch ein weiteres Programm aufrufen. Es kann jedoch nicht das aufrufende Programm aufrufen.
Da ein Programm mehrere Programme aufrufen kann, kann die Effektivität durch Erstellen eines Universalprogrammes erhöht werden.
Nur globale Variable können üblicherweise im aufrufenden Programm verwendet werden. Übergeben Sie lokale Variablen, falls erforderlich, als Argumente, da sie üblicherweise nicht verwendet werden. Siehe Abschnitt 8.16 "Aufrufen mit einem Wert und einem Bezug".

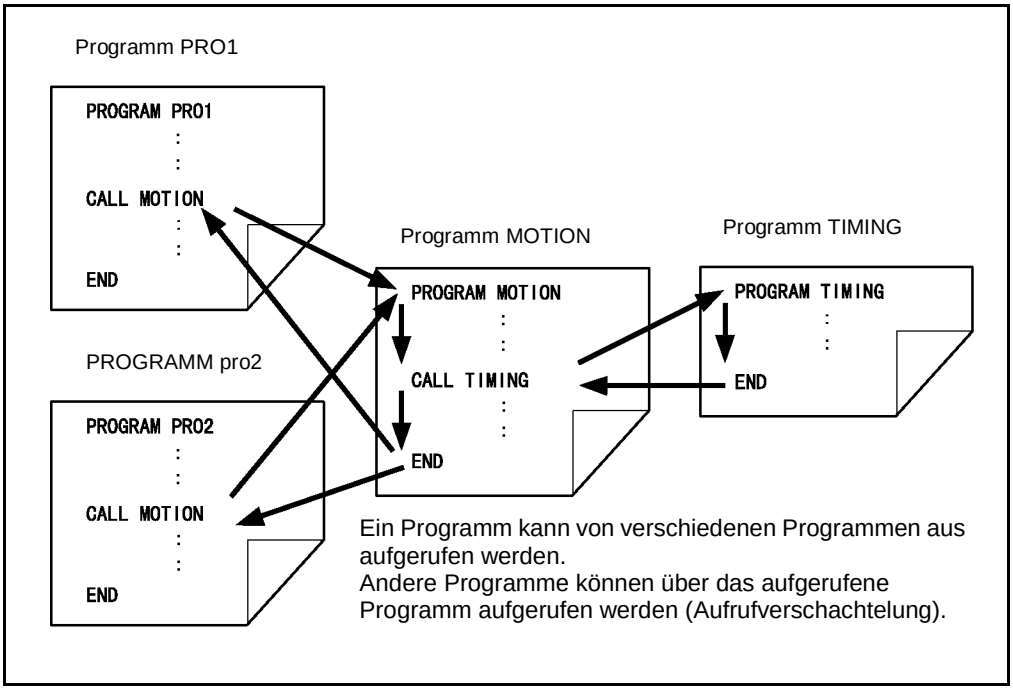


Abb. 2-3 Diagramm eines Programmaufrufs

2.1.3 Rekursiver Programmaufruf

Sobald ein Programm aufgerufen wird, kann das aufrufende Programm selbst mit der CALL-Anweisung als Programmname angegeben werden. Dies wird als rekursiver Aufruf bezeichnet.

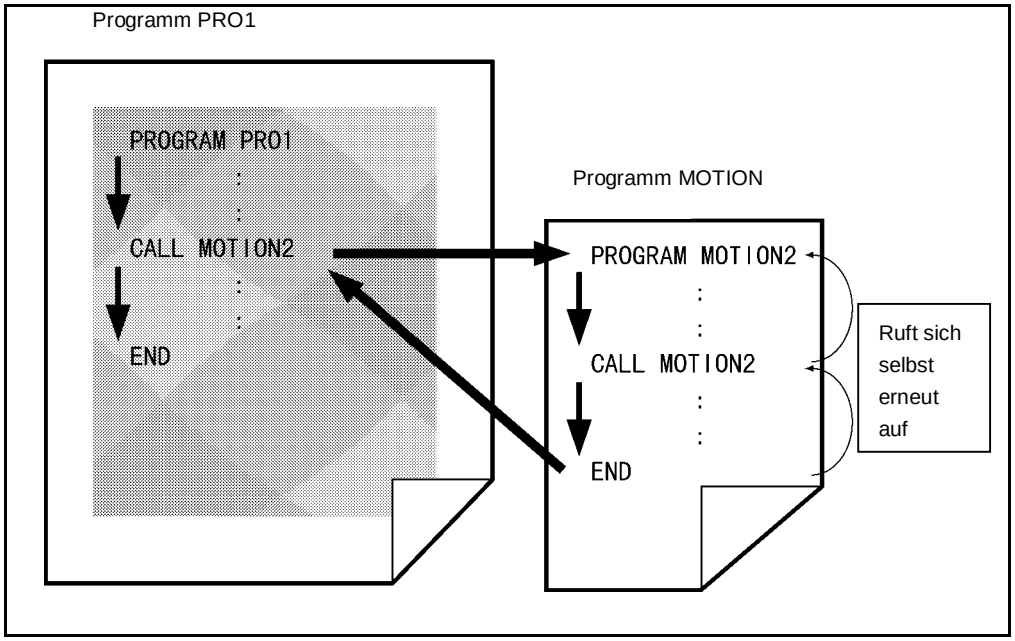


Abb. 2-4 Diagramm eines rekursiven Programmaufrufs

Anmerkung: **Die Variablen im PAC-Programm werden im statischen Speicher jedes Programmes gespeichert. Beachten Sie deshalb, daß bei einem rekursiven Aufruf Variablen verwendet werden müssen. Wenn der Anfangswert mit einer lokalen Variablen festgelegt wird, kehrt der Wert jedesmal beim Aufrufen des Programmes zu seinem Anfangswert zurück.**

2.2 Ausführen eines Programmes

Es gibt folgende 3 Möglichkeiten, ein Programm auszuführen:

- Über eine Programmiergerätoperation.
- Durch die Verwendung des Bedienfelds.
- Durch das Verwenden der E/A von einem externen Gerät.

Wie im folgenden beschrieben, gibt es je nach Ausführungsmethode Beschränkungen beim Ausführen eines Programmes.
Hinweise zu der Methode, mit der das gestartete Programm ein anderes Programm aufruft, finden Sie unter Abschnitt 2.1 "Aufrufen eines Programmes und eines Unterprogrammes".

2.2.1 Ausführen über das Programmiergerät

Um ein Programm über das Programmiergerät auszuführen, verwenden Sie den Programmierprüfmodus oder einen internen, automatischen Modus.
Es können nur Programme ausgeführt werden, die keine Argumente verwenden.
Programme mit einem beliebigen Programmnamen können ausgeführt werden.

2.2.2 Ausführen über das Bedienfeld

Um ein Programm über das Bedienfeld auszuführen, verwenden Sie den internen automatischen Modus.
Es können nur die Programme gestartet werden, die einen Namen haben, der dem Format “PRO< Nummer >” entspricht.

Anmerkung (1):	**Der Programmierprüfmodus kann vom Bedienfeld aus nicht gewählt werden.**
Anmerkung (2):	**Ein Programm mit einem Programmnamenformat “PRO< Nummer >” kann keine Argumente enthalten.**

2.2.3 Ausführen über ein externes Gerät

Im externen automatischen Modus kann ein Programm durch Eingabe einer externen E/A ausgeführt werden.
Es können nur die Programme gestartet werden, die einen Namen haben, der dem Format “PRO< Nummer >” entspricht.
Über externe Geräte können nur die Programme PR00 bis PR032767 im Standardmodus und die Programme PR00 bis PR0127 im kompatiblen Modus ausgeführt werden.

Anmerkung:	**Ein Programm mit einem Programmnamenformat “PRO< Nummer >” kann keine Argumente enthalten.**

2.3 Multitasking

Ein Programm mit PAC-Sprache kann gleichzeitig die Fortschrittsverwaltung mehrerer Programme ausführen. Jedes Programm hat seinen eigenen Bewegungsprozeß, der als Task bezeichnet wird.
Gleichzeitige Forschrittsverwaltung mehrerer Programme bedeutet also, daß mehrere Tasks vorliegen. Dies wird als Multitasking bezeichnet.

2.3.1 Priorität

Um eine Task mit dem RUN-Befehl auszuführen, geben Sie eine Priorität an. Folglich haben alle Aufgaben (Tasks) ihre eigene Priorität.
Der Roboter der VS-Reihe prüft bei einer Programmausführung die Aufgabenpriorität in festgelegten Zeitintervallen und ändert die Aufgaben, damit jene im Wartestatus mit der höchsten Priorität ausgeführt werden.
Folglich werden Programme in der Abfolge von der höchsten zur niedrigsten Priorität ausgeführt. Bei gleicher Priorität steuert der Roboter die Aufgaben so, daß sich die Reihenfolge der Aufgabenausführung mit jedem festgelegten Zeitintervall ändert. Da die wechselnde Aufgabenausführung nur kurze Zeit zur Steuerung erfordert, hat es den Anschein, als wenn mehrere Programme gleichzeitig ausgeführt werden.

2.3.2 Kommunikation zwischen Tasks

Wird der Roboter im Multitasking-Betrieb gesteuert, müssen Aufgaben manchmal synchronisiert oder exklusiv gesteuert werden, damit diese nicht zur gleichen Zeit ausgeführt werden. Eine Kommunikation zwischen Aufgaben (Tasks) ist unbedingt erforderlich, da Signale ausgetauscht werden müssen.
Zur Kommunikation zwischen Aufgaben werden im allgemeinen Semaphoren verwendet, aber bei Robotern der VS-Reihe kann auch die E/A verwendet werden.

2.3.2.1 Semaphor

Nach Bedarf können bis zu 32 Semaphoren erstellt werden. Vor dem Verwenden von Semaphoren wird eine Semaphor-ID vergeben, falls der Semaphor mit dem CREATESEM-Befehl erstellt wird. Diese Semaphor-ID dient dazu, einen bestimmten Semaphor zu benennen.
Bei jeder Synchronisationssteuerung oder exklusiven Steuerung führt eine Aufgabe, die auf den Befehl einer anderen Aufgabe wartet, einen TAKESEM-Befehl aus und wartet dann darauf, von der anderen Aufgabe einen Befehl zur Ausführung zu erhalten, und zwar in Form eines GIVESEM-Befehls. Wenn die Aufgabe zum Weiterleiten eines Befehls bereit ist, führt sie den GIVESEM-Befehl aus und ermöglicht es der Aufgabe, die auf den Semaphor wartet, ihren Prozeß auszuführen.
Der GIVESEM-Befehl ist nur für eine Aufgabe gültig, die auf einen Semaphor wartet.
Warten mehrere Aufgaben auf einen Semaphor mit der gleichen Semaphor-ID, kann eines von zwei Warteschleifensystemen zur Verarbeitung ausgewählt werden (Reihenfolge nach dem Eingang oder der Prioritätsreihenfolge beginnend mit einer spezifischen Aufgabe). Verwenden Sie den CREATESEM-Befehl, um das Wartesystem anzugeben.

Die beiden folgenden Programme “MOTION1” und “TIMING” sind Beispiele für eine Synchronisationssteuerung, die Semaphoren verwenden.
Beim Starten des Programmes “TIMING1” wird das Programm “MOTION1” gleichzeitig gestartet. Das Timing für “MOTION1” wird von “TIMING1” mit einem Semaphor zur Ausführung des MOVE-Befehls erstellt.

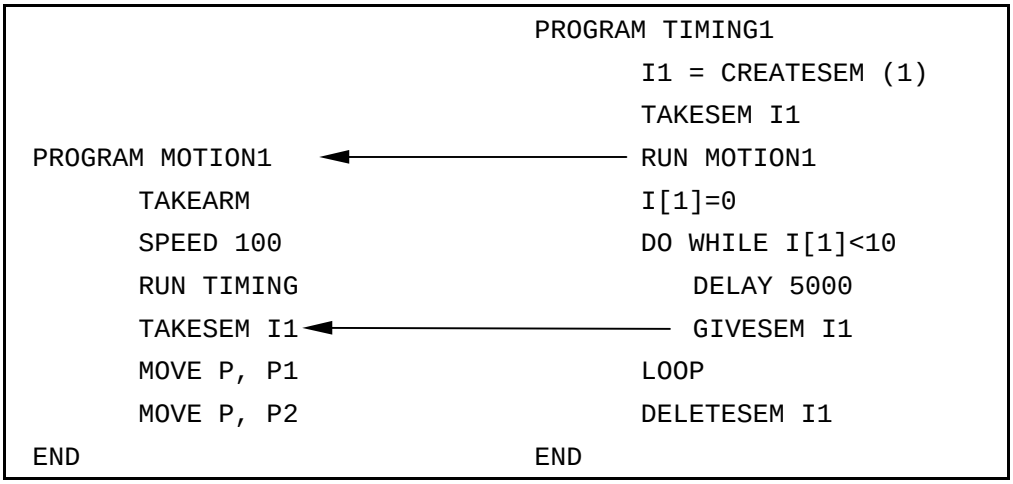


Abb. 2-5 Beispiel für eine Synchronisationssteuerung mit Semaphor

2.3.2.2 E/A

Der Roboter kann über E/A die Kommunikation zwischen Tasks herstellen.
Die beiden folgenden Programme “MOTION2” und “TIMING2” sind Beispiele für ein Synchronisationssteuerung, die E/A verwenden. Die Bewegung ist die gleiche wie in “Beispiel der Synchronisation mit Semaphor” in Abschnitt 2.3.2.1 “Semaphor.”
Wenn die Kommunikation zwischen Tasks über E/A hergestellt wird, kann in der Warteschleife nur die Reihenfolge des Eingangs berücksichtigt werden.

Bei Ausführung des Programmes “TIMING2” wird das Timing von “MOTION2” von “TIMING2” entsprechend dem Status von IO [118] zur Ausführung des MOVE-Befehls erstellt. Diese Operation wiederholt sich, bis die “TIMING2”-Schleife endet.

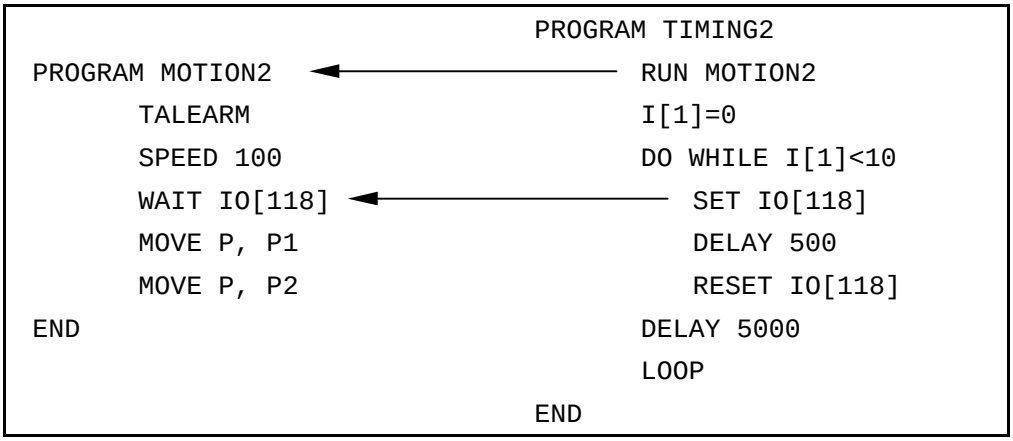


Abb. 2-6 Beispiel für eine Synchronisationssteuerung über E/A

2.4 Serielle Kommunikation

2.4.1 Schaltkreisnummer

Tabelle 2-1 führt die Schaltkreisnummern mit den entsprechenden Kanalnummern der Robotersteuerung auf.

Tabelle 2-1 Schaltkreisnummern der Robotersteuerung

Schaltkreisnummer	Kanalnummer	Verwendung
0	ch1	Anschluß des Programmiergeräts
1	ch2	Universalanschluß
2	ch3	(Reserviert)
3	ch4	(Reserviert)

2.4.2 Kommunikationsbefehl

Die folgenden 4 PAC-Sprachbefehle werden zur Kommunikation über die RS232C-Schnittstelle verwendet.

- PRINT-Anweisung (Ausgabe)
- WRITE- Anweisung (Ausgabe)
- INPUT-Anweisung (Eingabe)
- LINEINPUT-Anweisung (Leitungseingabe)

Die PRINT- und WRITE-Anweisungen unterscheiden sich in der Ausgabeform. Die Ausgabe von Zeichenfolgen in Form einer WRITE-Anweisung kann mit der INPUT-Anweisung korrekt eingegeben werden.

Erläuterungen zu den Befehlen finden Sie in Teil 2 “Befehlsreferenz.”

- PRINT-Anweisung : “PRINT”
- WRITE-Anweisung : “WRITE”
- INPUT-Anweisung : “INPUT”
- LINEINPUT-Anweisung : “LINEINPUT”

2.4.3 Bereinigen des Kommunikationspuffers

Die FLUSH-Anweisung löscht den RS232C-Kommunikationspuffer.
Stellen Sie sicher, daß dies vor Beginn der Kommunikation ausgeführt wird.

- FLUSH-Anweisung : “FLUSH”

2.4.4 Musteranwendung

In diesem Abschnitt wird ein praktisches Programmbeispiel beschrieben, nämlich eine einfache Anwendung, die mit serieller Kommunikation arbeitet. Bei praktischen Anwendungen wird auf dieses Beispiel verwiesen.
Die Robotersteuerung zählt bei dieser Musteranwendung die Zahl der Bewegungen und übermittelt die zugehörigen Daten in festgelegten Zeitintervallen. Der PC empfängt die von der Robotersteuerung gesendeten Daten für die Zahl der Bewegungen. Das Programm in der Robotersteuerung ist in der PAC-Sprache geschrieben. Visual Basic wird für ein Beispiel am PC verwendet.
Wenn die in Abschnitt 2.4.4.1 “Roboter-Controller-Programm” und Abschnitt 2.4.4.2 “PC-Programm” gezeigten Programme in der Robotersteuerung bzw. im PC vorbereitet und anschließend ausgeführt werden, wird die Anzahl der Roboterbewegungen über den RS232C-Schaltkreis übertragen und vom PC empfangen.

2.4.4.1 Roboter-Controller-Programm

Abb. 2-7 und 2-8 zeigen Beispiele für Programme in der Robotersteuerung. Diese beiden Programme führen Multitasking-Operationen aus.
Das Programm “PRO1” initialisiert zunächst den Zähler und die Koordinaten und führt anschließend “PRO2” aus. In “PRO1” führt der Roboter die Reziprokbewegung zwischen “P2” und “P1” aus. Bei dieser Operation sendet “PRO2” die Anzahl der in der Variablen I[1] gespeicherten Roboterbewegungen alle 2 Sekunden (2000 Millisekunden) über den RS232C-Anschluß (ch2).

```
'!TITLE ”Roboterbewegung”
PROGRAM PRO1
     I1=0                          ' Löscht die Zählung.
     P1=(370,400,750,-90,90,0,5)   ' Legt die Koordinate fest.
     P2=(510,400,750,-90,90,0,5)   ' Legt die Koordinate fest.
     RUN PRO2                      ' Führt Tasks gleichzeitig aus.
     TAKEARM                       ' Ruft die Priorität für die
                                   ' Robotersteuerung ab.
     DO
         MOVE P,P2                 ' Bewegung zu Punkt P2.
         DELAY 1000                ' Wartet 1 Sekunde.
         MOVE P, P1                ' Bewegung zu Punkt P1.
         I1=I1+1                   ' Zählt die Anzahl
                                   ' der Bewegungen.
     LOOP                          ' Führt die Schleife aus.
END
```

Abb. 2-7 Programm “PRO1” “Roboterbewegung” der Robotersteuerung

```
'!TITLE ”Senden”
PROGRAM PRO2
   DO
        PRINT #2,I1
        DELAY 2000
   LOOP
END
```

Abb. 2-8 Programm “PRO2” “Senden” der Robotersteuerung

2.4.4.2 PC-Programm

Unten finden Sie das Programm, wie es im Personal Computer verwendet wird. Der einleitende Abschnitt der Programmliste beschreibt ein Programmodul zur Kommunikation. Im letzten Abschnitt des Programmes finden Sie Programme zum Empfangen von Daten unter Verwendung jedes Moduls.

Anmerkung: Dieses Programm verwendet Visual Basic von Microsoft Corporation und die Kommunikationssteuerung von Bunka Orient Co., Ltd. Führen Sie daher das Programm auf dem PC aus, auf dem Visual Basic und PDQComm installiert sind.

```
'   Moduldefinitionsanweisung
'
'   Einstellung des Kommunikationsanschlusses und der
'   Kommunikationsgeschwindigkeit
'   Anschlußnr.%  : Gibt Kommunikationsanschluß 1 oder 2 an
'   ComInitString$ : Stellt die Kommunikationsgeschwindigkeit etc.
'   ein:   “9600,N,8,1”
'
Private Sub CommOpen(PortNo%,ComInitString$)
       '---------------------------------------
       ' Einstellung der Kommunikationsgeschwindigkeit
       '---------------------------------------
       PDQComm1.CommPort = PortNo%
       PDQComm1.Settings = ComInitString$      '“9600,N,8,1”
       '---------------------------------------
       ' Ohne Datenaustausch (Handshake)
       ' Einstellung des Eingabezeitüberschreitungswertes
       ' (in Millimeter)
       ' Empfangen der Puffergrößeneinstellung
       ' Senden der Puffergrößeneinstellung
       '---------------------------------------
       PDQComm1.Handshaking = 0
       PDQComm1.InTimeout = 50
       PDQComm1.InBufferSize = 2048
       PDQComm1.OutBufferSize = 2048

       '---------------------------------------
       ' Öffnen eines Anschlusses.
       '--------------------------------------- Fortsetzung folgt
                                                 auf der nächsten
                                                 Seite.
```

```
        PDQComm1.PortOpen = True
End Sub


'
' Schließt den Kommunikationsanschluß.
'
'
Private Sub CommClose()
        '----------------------------------------
        ' Schließt einen Anschluß.
        '----------------------------------------
        PDQComm1. PortOpen = False
End Sub

'
' Umschließt die zum Kommunikationsanschluß angegebene Zeichenfolge
' in doppelten Anführungszeichen. Das System fügt dann einen
' Zeilenumbruch am Ende der Zeile ein und sendet diese.
' SendText$:  Zu sendende Zeichenfolge
'
Private Sub CommWrite(SendText$)
        '----------------------------------------
        ' Wenn Zeichenfolge und Steuerungscode (CR+LF) gesendet sind.
        '----------------------------------------
        PDQComm1.Output = Chr(&H22) &SendText$ & Chr(&H22) & Chr(13) &
        Chr(10)

        '----------------------------------------
        ' Wartet bis der Ausgabepuffer leer ist.
        '----------------------------------------
        Do
               DoEvents
        Loop Until PDQComm1. OutBufferCount = 0
End Sub

'
' Ruft die vom Kommunikationsanschluß empfangene Zeichenfolge ab,
' bis der Zeilenumbruch erfolgt und sendet dann die
' Zeichenfolge.
'
Private Function CommRead()

        '----------------------------------------
        ' Datenleseprozeß
        '----------------------------------------
        InString$ =“”
        Do While 1
               DoEvents
```

Fortsetzung folgt auf der nächsten Seite.

```
            '---------------------------------------
            'Lesen empfangener Daten.
            '---------------------------------------
            If PDQComm1. InBufferCount > 0 Then
                  InString$ = InString$ +PDQComm1.Input
                  If InStr(1, InString$, vbCr,
                  vbBinaryCompare) <> 0 Then Exit Do
            End If
      Loop
      CommRead = InString$

End Function

Using sample
Private Sub mnuCh1_Click()
      ' Verwendet Anschluß 1 und stellt die Baudrate auf 19200 ein.
      CommOpen 1,”19200,N,8,1”
      ' Empfängt die Anzahl der Bewegungen vom Controller.
      ReadBuf$ = CommRead
      ' Schließt den verwendeten Anschluß.
      CommClose
End Sub
```

Abb. 2-9 PC-Programm

2.5 Bibliothek

Die Programmbibliothek dient dazu, Allround-Programme zu sammeln und diese dementsprechend zu verwenden. Da andere Programme von einem Programm aus aufgerufen werden können, können in der PAC-Sprache Programme unter Verwendung von Programmen der Bibliothek oder durch Registrieren eines erstellten Programmes in der Bibliothek effizienter entwickelt werden.

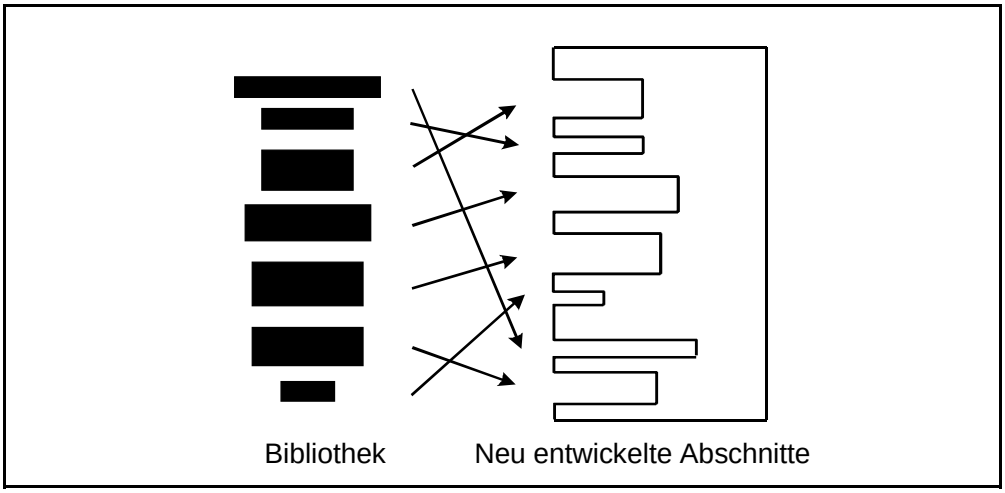


Abb. 2-10 Darstellung der Programmentwicklung mit der Bibliothek

2.5.1 Programmbank

Der PAC-Manager von WINCAPS II verfügt über eine Programmbank zur Verwendung der Bibliothek. Bei der Programmbank handelt es sich um ein Tool, mit dem Programme als Bibliothek registriert oder registrierte Programme für ein Projekt verwendet werden können.
Hinweise zur Verwendung der Programmbank finden Sie im Abschnitt 5.6.2 “Programmbank” in Kapitel 5 des WINCAPSII-HANDBUCHES.

2.5.1.1 Klassifikationen der Programmbibliothek

Die Standardprogrammbibliothek ist in folgende 7 Kategorien unterteilt:

Tabelle 2-2 Kategorien der Standardprogrammbibliothek

	Klassenname	Beschreibung
1	Konventionelle Sprache	Bietet Funktionen ähnlich den konventionellen Sprachbefehlen.
2	Palettierung	Bietet eine Palettierungsfunktion.
3	Werkzeug-bedienung	Bietet Funktionen zur Werkzeugbedienung.
4	Ein-/Ausgabe	Bietet Funktionen für DIO- und RS232C-Eingabe/Ausgabe.
5	Armbewegung	Bietet (mit Ausnahme der oben beschriebenen) Funktionen für die Armbewegung.
6	Bildverarbeitung	Bietet Funktionen zur Bildverarbeitung.
7	Kompatibel mit Ver. 1.2	Bietet die mit Version 1.2 kompatible Bibliothek, die in der Controller Software Version 1.2* oder früher verwendet werden kann. Diese Bibliothek enthält drei Programme: ndVcom, pltMove und pltMove0. Wenn in Version 1.2* oder früher eines dieser Programme verwendet wird, das nicht aus dieser Bibliothek, sondern aus den oben beschriebenen Kategorien 1 bis 6 stammt, führt dies zu einem Kompilierungsfehler. Verwenden Sie Bibliotheken in den oben beschriebenen Kategorien 1 bis 6 mit Ausnahme dieser drei Programme.

2.5.2 Palettierungsbibliothek

Es gibt keine spezielle Palettierungsanweisung in der PAC-Sprache. Palettierung kann unter Verwendung eines als Bibliothek vorbereiteten Programmes ausgeführt werden.
Unter Verwendung der Programmbank kann eine Bibliothek zur Palettierung für ein Projekt herangezogen werden. Wenn das System “Neues Systemprojekt” ausführt, wenn [1-Palettierung] in der Spalte [Einrichtungstyp] des Dialogfensters [Neues Projekt] ausgewählt ist, wird im Programmprojekt vom Anfang der Bibliothek automatisch ein Bibliotheksprogramm registriert.

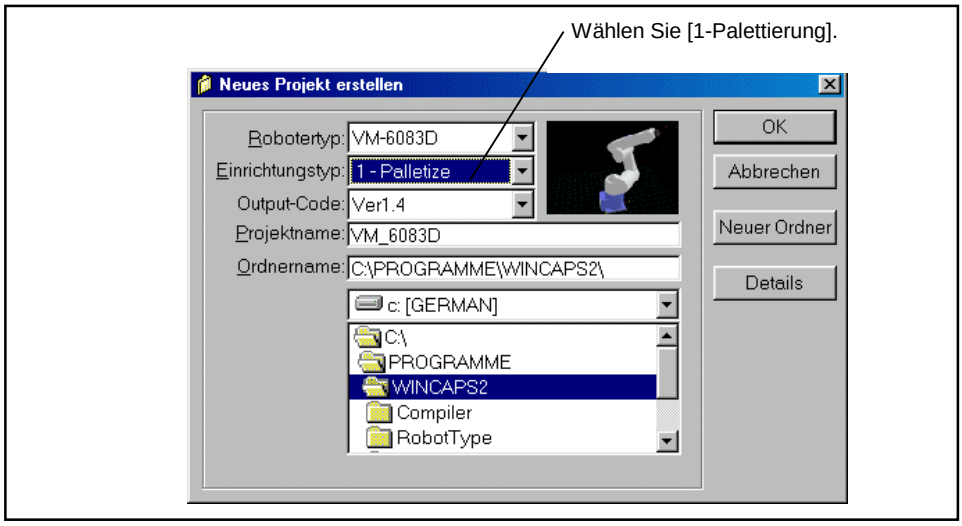


Abb. 2-11 Dialogfenster [Neues Projekt]

2.5.2.1 Palettierung

Die Palettierung dient dazu, Teile auf einer Palette (wie in Abb. 2-12 dargestellt) anzuordnen oder in einer bestimmten Reihenfolge von einer aufgeteilten Palette zu nehmen.
Ein Bibliotheksprogramm zur Palettierung wird erstellt, damit der Roboter die Palettierung nur ausführt, wenn die Zahl der Unterteilungen und Positionen der 4 Ecken programmiert sind.
Das Palettierungsprogramm ändert die Entnahmeposition auf der Palette jedesmal wenn das Programm aufgerufen wird.

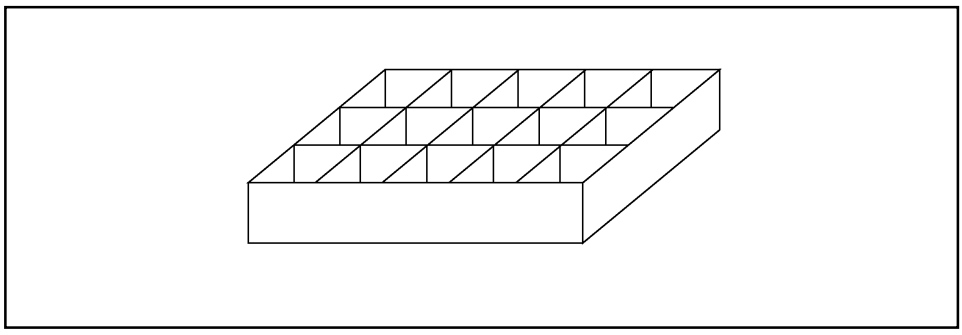


Abb. 2-12 Palette mit Unterteilungen

[1] Palettierungs-Parameter

Die für die Palettierung erforderlichen Parameter sind in den Abbildungen 2-13 bis 2-15 sowie in Tabelle 2-3 dargestellt.

Die PAC-Sprache speichert diese Parameter als Werte von Variablen.

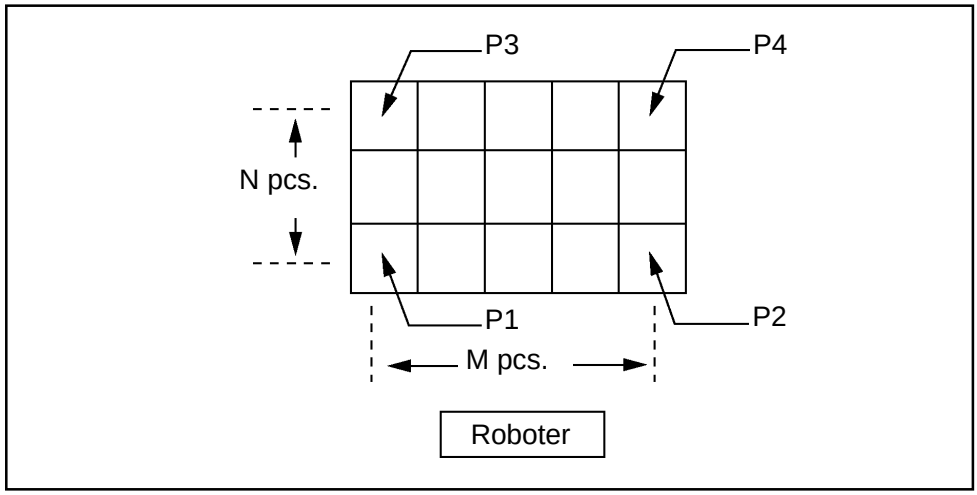


Abb. 2-13 Draufsicht auf eine Palette

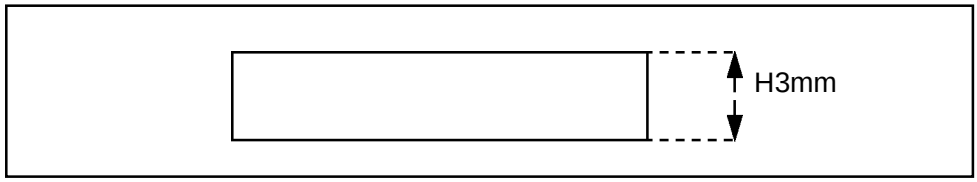


Abb. 2-14 Seitenansicht einer Palette

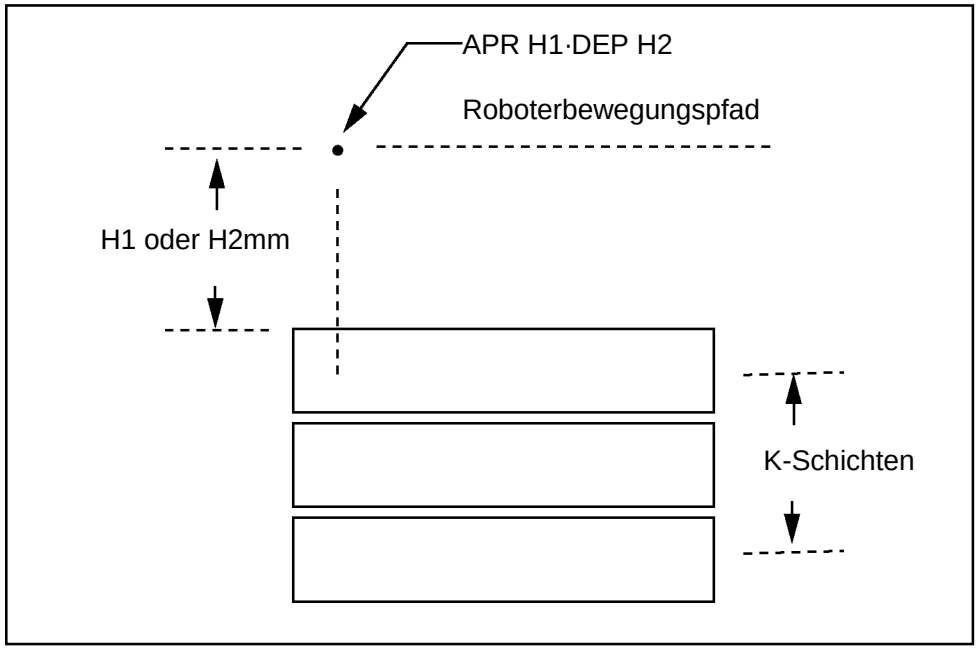


Abb. 2-15 Gestapelte Paletten

Tabelle 2-3 Für die Palettierung erforderliche Parameter

Symbol	Name	Bedeutung	Einheit
	Palettierungsnummer	Palettierungs-Indexnummer	Keine (ganze Zahl)
N	Zahl der Seitenunterteilungen	Zahl der Unterteilungen von P1 zu P3	Pc (ganze Zahl)
M	Zahl der Längsunterteilungen	Zahl der Unterteilungen von P1 zu P2	Pc (ganze Zahl)
K	Zahl der Schichten	Anzahl der Palettenschichten	Pc (ganze Zahl)
H1	Annäherungslänge	Annäherungslänge, wenn der Roboter auf eine Palette zugreift.	mm (Real, einfache Genauigkeit)
H2	Abweichungslänge	Abweichungslänge, wenn der Roboter sich von der Palette fortbewegt.	mm (Real, einfache Genauigkeit)
H3	Palettenhöhe	Höhe einer Palette	mm (Real, einfache Genauigkeit)
	H1 und H2 müssen jedoch folgende Bedingungen erfüllen: H1 > [H3 × (K-1)] + 5 H2 > [H3 × (K-1)] + 5		
P1 P2 P3 P4	Die relative Position der 4 Eckpunkte der in Abb. 2-13 dargestellten Palette können nicht ausgetauscht werden. Die für Position P1 programmierte Roboterarmstellung wird an allen Punkten beibehalten.		

N Zahl der Seitenunterteilungen
Gibt die Zahl der Unterteilungen in Seitenrichtung der Palette an.
Abb. 2-13 zeigt 3 Unterteilungen.

M Zahl der Längsunterteilungen
Gibt die Zahl der Unterteilungen in Längsrichtung der Palette an.
Abb. 2-14 zeigt 5 Unterteilungen.

K Zahl der gestapelten Paletten
Gibt die Zahl der Paletten in einem Palettenstapel an.
Abb. 2-15 zeigt 3 gestapelte Paletten.

H1 Annäherungslänge
Gibt die Annäherungslänge an, wenn der Roboter auf eine Palette zugreift.
Jedesmal wenn das Palettierungsprogramm aufgerufen wird, wird die gleiche Annäherungslänge verwendet.

H2 Abweichungslänge
Gibt die Abweichungslänge an, wenn der Roboter sich von der Palette fortbewegt.
Jedesmal wenn das Palettierungsprogramm aufgerufen wird, wird die gleiche Abweichungslänge verwendet.

H3 Palettenhöhe
Gibt die Höhe einer Palette an.
Erhöht sich die Zahl der gestapelten Paletten, geben Sie einen positiven Wert ein.
Nimmt die Zahl der gestapelten Palette aufgrund entfernter Paletten ab, geben Sie einen negativen Wert ein.
Ändert sich die Zahl der gestapelten Paletten nicht, geben Sie 0 ein.

Anmerkung: H1 und H2 müssen folgende Bedingungen erfüllen.

H1 > {H3 × (K-1)} + 5

H2 > {H3 × (K-1)} + 5

Sind diese Bedingungen nicht erfüllt, tritt ein Fehler bei der Initialisierung auf. Sie müssen erfüllt sein, damit der Roboter nicht mit einer Palette kollidiert. Mit diesen Bedingungen werden Annäherungs- und Abweichungspunkte festgelegt, die 5 mm über der maximalen Zahl an gestapelten Paletten liegen. Selbst wenn die Zahl der gestapelten Paletten zu- oder abnimmt, bleiben Annäherungs- und Abweichungspunkte so wie in Abb. 2-16 dargestellt unverändert.

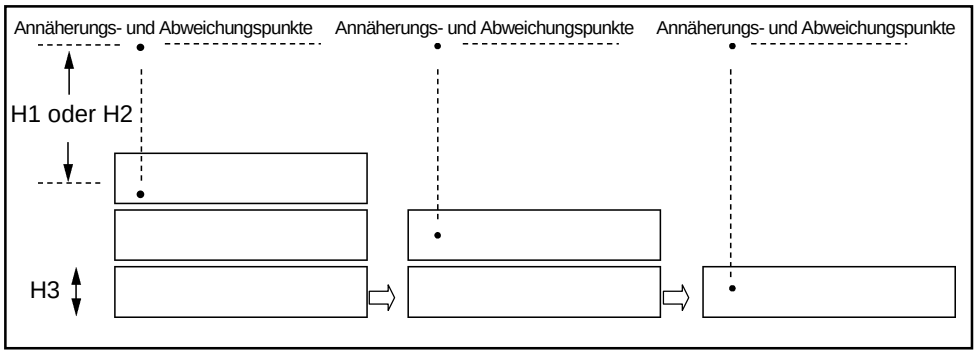


Abb. 2-16 Änderung der Zahl gestapelter Paletten und der Annäherungs- bzw. Abweichungspunkte.

Die Punkte P1, P2, P3 und P4 an den 4 Ecken der Palette geben die Positionen der einzelnen Teile an. Abb. 2-17 zeigt jeden Punkt und die jeweilige Ausführungsfolge an.

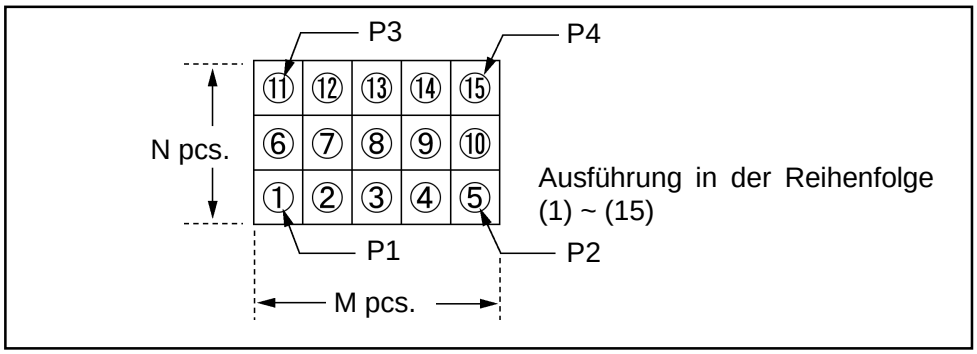


Abb. 2-17 Palettierungsabfolge

[2] Einstellen des Parameterwertes

Stellen Sie die Parameterwerte wie die Anzahl der Unterteilungen in Seiten- und Längsrichtung und die Anzahl der gestapelten Paletten ein, indem Sie “pltInitialize” in der Programmbibliothek aufrufen.
Wenn Sie [1-Palettierung] in [Einrichtungstyp] im Dialogfenster [Neues Projekt] auswählen, wenn “Neues Systemprojekt” ausgeführt wird, wird das Programm “PRO2” mit dem Titel “Palettierungsinitialisierung Schablone 1” automatisch in der Bibliothek registriert. Der Name dieses Programmes lautet “pltInitialize”; ändern Sie also die Werte der Argumente in der CALL-Anweisung auf die gewünschten Werte, bevor Sie das Programm verwenden.

```
'!TITLE ”Palettierungsinitialisierung Schablone 1”
#DEFINE pltIndex    0

PROGRAM PRO2
  CALL pltInitialize(pltIndex, 4,3,1,50,50,50,52,53,54,55)     'Initialisiert die
                                                               'Palettierung 0.
END
```

Abb. 2-18 Bibliotheksprogramm “Palettierungsinitialisierung Schablone 1”

```
CALL pltInitialize(pltIndex, 3, 5, 3, 50, 50, 10, 52, 53, 54, 55) bei der Parameterbeschreibung
   1. . . Palettierungsprogrammnummer (pltIndex)
   2. . . Zahl der Seitenunterteilungen (3)
   3. . . Zahl der Längsunterteilungen (5)
   4. . . Zahl der gestapelten Paletten (3)
   5. . . Annäherungslänge (50mm)
   6. . . Abweichungslänge (50mm)
   7. . . Palettenhöhe (10mm)
   8. . . Position P1 (P52)
   9. . . Position P2 (P53)  }
  10. . . Position P3 (P54)  }  Geben Sie nur eine Zahl für Variable des Typs P an.
  11. . . Position P4 (P55)  }
```

Abb. 2-19 Beschreibung des Parameters “pltInitialize”

Die Bedeutung dieses Parameters kann mit dem Werkzeug “Befehlseditor” im PAC Manager von WINCAPSII angezeigt werden.

[3] Palettierungszähler

Beim Palettieren zählt der Roboter die Zahl der Unterteilungen und speichert die gezählten Werte als Variable.
Es gibt 4 Arten von Zählern, nämlich Seitenrichtung (N), Längsrichtung (M), Höhe (K) und Gesamtzahl (cnt).
Diese Zähler sind in “pltKernl” definiert, dem Kernprogramm zum Steuern der Palettierungsoperation.
Das Bibliotheksprogramm “pltMove” fügt dem Gesamtzähler jedesmal den Wert Eins hinzu, wenn eine Operation abgeschlossen ist; außerdem werden die Werte der anderen Zähler angepaßt.
Das Bibliotheksporogramm “pltDecCnt” subtrahiert bei jedem Aufruf den Wert 1 von dem Gesamtzähler; die einzelnen Zählerwerte werden durch Subtraktion angepaßt.
Bis zu 30 Palettierungsprogramme können bei der Anfangseinstellung erstellt werden.
Deshalb gibt es 31 Gruppen von Palettierungszählern. Das Ändern der Anzahl der Palettierungsprogramme ist in Abschnitt 2.5.2.2.1 “Anpassen des Palettierungsprogramms” beschrieben.

Zählregel

Der Palettierungszähler fügt dem Gesamtzähler bei jeder Ausführung von “pltMove” den Wert Eins hinzu; außerdem werden die Werte der anderen Zähler angepaßt. Dadurch ist die nächste Palettenposition sichergestellt.
Wird 1 zum Gesamtzähler hinzuaddiert, bewegt sich der Längsrichtungszähler (M) zur nächsten Position. Erreicht der Längsrichtungszähler (M) das Ende bzw. seinen Maximalwert, dann bewegt sich der Seitenrichtungszähler (N) zur nächsten Position und der Längsrichtungszähler wird zu seinem Minimalwert. Erreicht der Breitenrichtungszähler (N) das Ende bzw. den Maximalwert, dann bewegt sich der Vertikalrichtungszähler (K) zur nächsten Position und der Breitenrichtungszähler (N) wird zum festgelegten Minimalwert.
Wird ein Palettierungsprogramm mitten in der Bearbeitung unterbrochen und dann wieder neu gestartet, bewegt sich der Roboter zur nächsten Position, weil vom Wert der Zählervariable weitergezählt wird.
Der Wert des Palettierungszählers bleibt gespeichert, selbst wenn die Stromzufuhr abgeschaltet wird. Wenn das System nach dem erneuten Systemstart nicht initialisiert ist, fährt der Roboter mit der Palettierung anhand des letzten Zählerwertes fort.

Anmerkung:	**Wird ein Ausführungsprogramm nach dem Kompilieren ausgeführt, werden die Werte der Variablen initialisiert.**

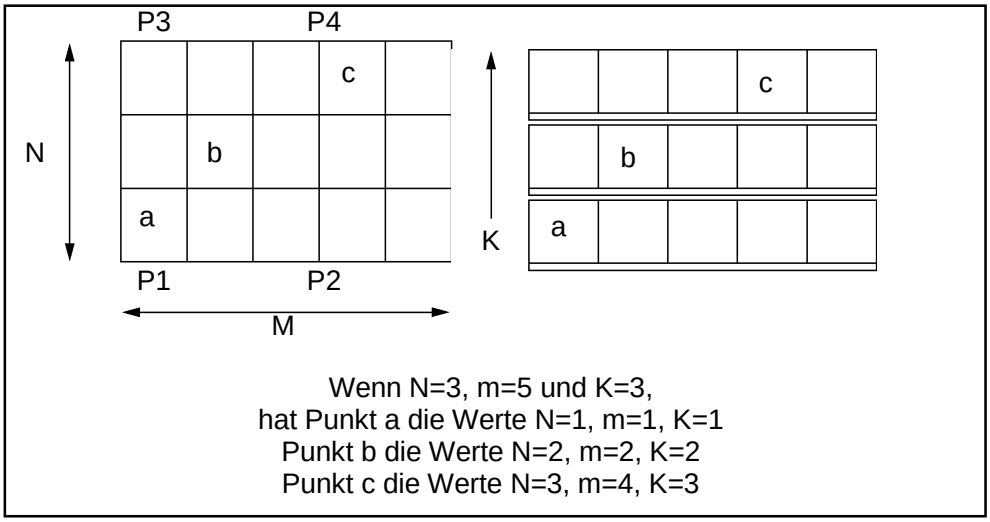


Abb. 2-20 Verhältnis zwischen Palettierungsposition und Zähler

Zählerinitialisierung

Sobald Paletten ersetzt oder nicht alle Unterteilungen verwendet werden, müssen die Zähler initialisiert werden.
Bei der Initialisierung der Zähler werden alle Palettierungszähler auf “1” eingestellt.
Alle Palettierungszähler können auf einmal initialisiert werden durch Verwendung von “pltResetAll” des Bibliotheksprogrammes.
Wenn beispielsweise alle Zähler der Palette Nummer 1 initialisiert werden, sieht die Beschreibung wie folgt aus:

```
CALL    pltResetAll(1)
```

Wenn jeder Palettierungszähler einzeln initialisiert wird, verwenden Sie “pltLetN1,” “pltLetM1,” “pltLetK1” und “pltLetCnt.”
Wenn beispielsweise der Zähler N der Palettennummer 1 initialisiert wird, sieht die Beschreibung wie folgt aus:

```
CALL    pltLetN1(1, 1)
```

Anmerkung: Variablen werden nach dem Kompilieren initialisiert, indem das Ausführungsprogramm geladen wird.

Endesignal des Palettierungsprogrammes

Das Palettierungsprogramm setzt je eine entsprechende Markierung, wenn ein oder alle Palettenstapel fertig bearbeitet sind.
Verwenden Sie zum Abrufen des Status “getapelte Palette” das Bibliotheksprogramm “pltGetPLT1END.” Verwenden Sie zum Abrufen des Status “gesamter Palettenstapel fertig” das Bibliotheksprogramm “pltGetPLTEND.”
VerwendenSie zum Zurücksetzen (0) der Markierung “gestapelte Palette fertig” das Bibliotheksprogramm “pltResetPLT1END.” Verwenden Sie zum Zurücksetzen (0) der Markierung “gesamter Palettenstapel fertig" das Bibliotheksprogramm “pltResetPLTEND.”

2.5.2.2 Palettierungsprogramm

Ein praxisnahes Palettierungsprogramm variiert je nach Anwendung für bestimmte Situationen. Die Standardprozedur zum Zusammensetzen des Programmes durch Verwendung der Bibliothek wird jedoch im Programm “PRO1” mit dem Titel “Palettierung Schablone 2” vorbereitet.
Verwenden Sie dieses Programm “Palettierung Schablone 2” als Modell, und fügen Sie die erforderlichen Elemente für die Anwendung hinzu, um die Programmentwicklung voll auszuschöpfen.
Das Programm “PRO1” mit dem Titel “Palettierung Schablone 2” ist in Abb. 2-21 dargestellt.
Dieses Beispiel beruht auf folgenden Annahmen:

- Die Punkte der Paletierung sind P50-P55.
- In dem vorbereiteten Programm bewegt sich der Roboter an die feste Position P50 und führt das Paletierungsprogramm 0 aus. Anschließend bewegt er sich an die Assemblierungsposition P51 zum Loslassen. Er prüft das Stapelende und ob das Endesignal ausgegeben wird, und wechselt die Palette. Wenn kein Arbeitsmaterial mehr vorhanden ist, beendet er die Bewegung.

“Palettierung Schablone 2” mit dem Programmnamen “PRO1” und “Palettierungsinitialisierung Schablone” mit dem Programmnamen “PRO2” werden automatisch registriert, wenn “Palettierung” unter “Einrichtungstyp” des Dialogfensters “Neues Projekt” ausgewählt ist, wenn ein neues Projekt mit WINCAPSII erstellt wird. Hinweise hierzu finden Sie unter Abchnitt 2.5.2 “Palettierungsbibliothek”

```
' !TITLE "Palettierung Schablone 2"
' !AUTHOR "Denso Roboter Engineering-Abtlg.
#DEFINE pltIndex     0               ' Palettierungsprogrammnummer.
                                     ' Jeder numerische Wert von 0 bis 30 steht für die
#DEFINE ChuckNG      40              ' Anzahl der Entnahmeprüfungen E/A zur Verfügung.
                                     ' Jeder numerische Wert steht zur Auswahl zur Verfügung.
PROGRAM PRO1                         ' Eingabe eines geeigneten Namens.
   MOVE P, P50                       ' Bewegung zur festgelegten Position P50.
                                     ' Bewegung zur palettierten P50.
   IF IO[ChuckNG] = ON THEN          ' Entnahmeprüfung für den vorangegangenen Ablauf.
   CALL pltDecCnt(pltIndex)          ' Vom Gesamtzähler 1 subtrahieren.
   END IF
   CALL pltMove(pltIndex)            ' Ausführung Palettierungsnummer 0.
   MOVE P, P51                       ' Bewegung zur Assemblierposition P51.
                                     ' Bewegung zur palettierten P51.
   '<--- Einfügen, z. B. einer Loslaß-Bewegung --->
   CALL pltGetPLT1END(pltIndex,0)    ' Abrufen eines Endesignals für die 1. Schicht in I[0].
                                     ' Das zweite Argument "0" ist eine Array-Nummer von "I",
                                     ' die in der nächsten Zeile festgelegt wird.
   IF I[0] THEN                      ' Wenn dies eingeschaltet ist (z. B. wenn <>0)
   '<--- Einfügen einer Palettenwechselbewegung --->
   CALL pltResetPLT1END(pltIndex)    ' Löschen des Endsignals für den ersten Stapel.
   END IF
END
```

Abb. 2-21 Programmname “PRO1” und “Palettierung Schablone 2”

[1] Anpassen des Palettierungsprogrammes

Frage: Wie ändert man ein auszuführendes Palettierungsprogramm auf den Wert 1 ab?
Antwort: Ändern Sie den Wert von “pltIndex” in “PRO1” und “PRO2” auf “1.”

Frage: Wie ändert man die Zahl der Palettierungsunterteilungen auf 5 für die Seiten- und 3 für die Längsrichtung?
Antwort: Ändern Sie “3” und “5” im zweiten und dritten Parameter von “pltInitialize” auf “5” bzw. “3”.
* CALL pltInitialize (pltIndex, 5, 3, 3, 50, 50, 10, 52, 53, 54, 55)

Frage: Wie ändert man die Zahl der verfügbaren Palettierungsprogramme?
Antwort: Ändern Sie “31” von “#DEFINE mcPaltMax 31” im Programm “pltKernel” auf eine beliebige Zahl, damit die Zahl der Palettierungsprogramme festgelegt werden kann.

Frage: Wie legt man einen Ausweichpunkt für die Palettierung fest?
Antwort: Wenn Sie der Annäherungsrichtung einen Hilfspunkt X = 10, Y = 10, Z = -10 hinzufügen, fügenen Sie einen Befehl hinzu zum Bewegen eines Punkts mit einer parallelen Abweichung “(10, 10, -10) H”, zusätzlich zu einem Punkt P[mcNextPos] in der Datei “paltmove1. pac”.

```
* MOVE P, P[mcNextPos] + (10, 10, -10)H     'Bewegt sich zum
                                            'Ausweichpunkt von
                                            'X=10, Y=10, Z=-10.
```

Frage: Wie erhält man N1, M1 und K1?
Antwort: Fügen Sie die Programmbibliotheken “pltGetK1”, “pltGetM1” und “pltGetN1” hinzu.

```
* call pltGetK1(Index, iValue)
* call pltGetM1(Index, iValue)
* call pltGetN1(Index, iValue)
'Index . . . Palettierungsprogrammnr.
'iValue. . . Nummer der Variablen des Typs I, an die die Werte
             K1, M1 und N1 zurückgegeben werden.
```

```
Beispiele) #define pltIndex    0     'Palettierungsprogrammnummer.
                  Program-Exmp
                  DefInt Index, iValue
                  Index = pltIndex  'Fügt eine Palettierungsprogrammnummer
                                    'in ein Indexregister ein.
                  call pltGetK1(Index, iValue)       'Gibt den Wert von
                                                     'K1 der Variable vom
                                                     'Typ I zurück,
                                                     'die mit iValue
                                                     'angegeben ist.
                  call pltGetM1(Index, iValue)       'Gibt den Wert
                                                     'M1 an die Variable
                                                     'vom Typ I zurück,
                                                     'die mit iValue
                                                     'angegeben ist.
                  call pltGetN1(Index, iValue)       'Gibt den Wert
                                                     'N1 an die Variable
                                                     'vom Typ I zurück,
                                                     'die mit iValue
                                                     'angegeben ist.
                  End
```

Frage: Wie ändert man N1, M1 und K1?

Antwort: Fügen Sie die Programmbibliotheken “pltLetK1”, “pltLetM1”, “pltLetN1” hinzu.

```
* call pltLetK1(Index, iValue)
* call pltLetM1(Index, iValue)
* call pltLetN1(Index, iValue)
         'Index . . . Palettierungsprogrammnr.
         'iValue. . . Geben Sie diesen Wert in K1, M1 und N1 ein.

Beispiele) #define pltIndex    0          'Palettierungsprogrammnummer

                   Program Exmp
                   DefInt Index, iValue
                   Index = pltIndex       'Fügt die Palettierungs-
                                          'programmnummer
                                          'in das Indexregister ein.
                   iValue = 10            'Gibt die Werte in K1, M1 und
                                          'N1 ein.
                   call pltLetK1(Index, iValue)
                                          'Gibt den Wert von iValue
                                          'in K1 ein.
                   call pltLetM1(Index, iValue)
                                          'Gibt den Wert von iValue
                                          'in M1 ein.
                   call pltLetN1(Index, iValue)
                                          'Gibt den Wert von iValue
                                          'in N1 ein.
                   End
```

Frage: Wie erhält man den Gesamtzählwert?

Antwort: Fügen Sie die Programmbibliothek “pltGetCnt” hinzu.

```
*call pltGetCnt(Index, iValue)
'Index ....Palettierungsprogrammnummer
'iValue ...Nummer der Variablen vom Typ I, der der Wert des Gesamtzählers
           zurückgegeben wird.

Beispiele) #define   pltIndex    0            'Palettierungsprogrammnummer.

                   Program Exmp
                   DefInt Index, iValue
                   Index = pltIndex       'Fügt die Palettierungs-
                                          'programmnummer in das
                                          'Indexregister ein.
                   call pltGetCnt(Index, iValue)
                                          'Gibt den Wert des
                                          'Gesamtzählers an die
                                          'Variable mit dem
                                          'Typ I zurück, die mit dem
                                          'iValue angegeben ist.
                   End
```

Frage: Wie ändert man den Gesamtzähler?

Antwort: Fügen Sie die Programmbibliothek “pltIncCnt” hinzu.

```
* callpltLetCnt(Index, iValue)
         'Index . . . Palettierungsprogrammnr.
         'iValue. . . Dieser Wert wird im Gesamtzähler eingegeben.

Beispiele) #define  pltIndex    0       'Palettierungsprogrammnummer.

                    Program Exmp
                    DefInt Index, iValue
                    Index = pltIndex  'Fügt die Palettierungs-
                                      'programmnummer
                                      'in das Indexregister ein.
                    iValue = 0        'Gibt einen Wert im
                                      'Gesamtzähler ein.
                    call pltLetCnt(Index, iValue)
                                      'Der Wert von iValue wird
                                      'im Gesamtzähler
                                      'eingegeben.
                    End
```

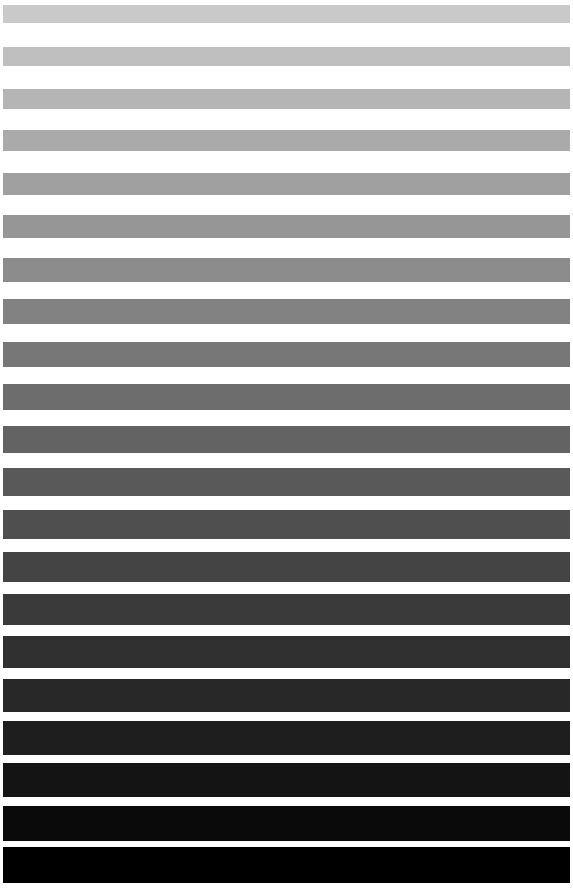
Frage: In welcher zeitlichen Abfolge wird der Gesamtzählwert erfaßt?

Antwort: Der Gesamtzähler zählt, wenn “pltKernel (Index, -15, mcNextPos, ndErr)” ausgeführt wird.
Wenn daher die Programmbibliotheken “pltGetNextPos,” “pltMove” und “pltMove0” ausgeführt werden, wird der Gesamtzählwert hochgezählt.

```
        * call  pltMove(Index)
        * call  pltMove0
        * call  pltGetNextPos(Index, NextPos)
                'Index . . . .    Palettierungsprogrammnr.
                'NextPos . . .    Variablennummer vom Typ P, wobei
                                  die Position der nächsten
                                  Position eingegeben wird.
```

Kapitel 3

Roboterbewegung



Es gibt verschiedene Roboterbewegungen entsprechend den Bezugspunkteinstellungen oder der Festlegung, ob der Roboter seine Zielposition erreicht. Dieses Kapitel beschreibt diese Roboterbewegungen.

3.1 Absolute Bewegung und relative Bewegung

3.1.1 Absolute Bewegung

Eine absolute Bewegung dient dazu, eine programmierte Position zu erreichen. Eine absolute Bewegung führt immer zu einer programmierten Position, ohne daß sie von der vorhergehenden Bewegung beeinflußt wird.
Die Befehle zur Ausführung einer absoluten Bewegung lauten wie folgt:

APPROACH, MOVE, GOHOME, DRIVEA

3.1.2 Relative Bewegung

Eine relative Bewegung dient dazu, sich um eine einprogrammierte Distanz von der aktuellen Position wegzubewegen.
Da eine relative Bewegung ihren Bezug zur aktuellen Position nach dem Ergebnis des Ausführens des vorhergehenden Bewegungsbefehls festlegt, beeinflußt der vorhergehende Bewegungsbefehl die relative Bewegung.
Die Befehle zur Ausführung einer relativen Bewegung lauten wie folgt:

DEPART, DRAW, DRIVE, ROTATE, ROTATEH

3.1.3 Beispiele für die absolute Bewegung und die relative Bewegung

Hier sind zwei Beispielprogramme, um den Roboter von der aktuellen Position P1 über den Punkt P2 zum Punkt P3 zu bewegen.

“MOVEMENT1” wird als absolute Bewegung ausgedrückt.
“MOVEMENT2” wird als absolute Bewegung und als relative Bewegung ausgedrückt. Bei Ausführung führen beide Programme, wie in Abb. 3-3 dargestellt, die gleiche Bewegung durch.

```
PROGRAM MOVEMENT1
        TAKEARM
        MOVE L, P2
        MOVE L, P3
END
```

Abb. 3-1 Beispiel für absolute Bewegung

```
PROGRAM MOVEMENT2
      TAKEARM
      MOVE L, P2
      DRAW L, V3    'V3 ist die relative Distanz zwischen P2 und P3.
END
```

Abb. 3-2 Beispiel für relative Bewegung

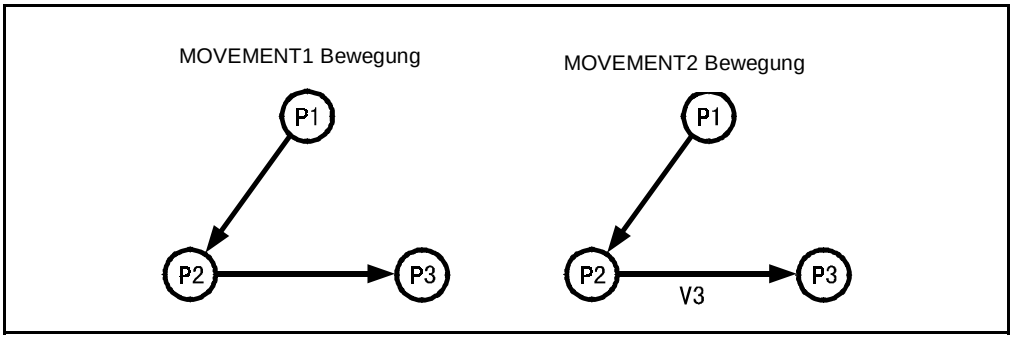


Abb. 3-3 Bewegungsbeispiele zweier Programme

Wenn Sie die erste Bewegungsanweisung “MOVE P, P1” aus “MOVEMENT1” und “MOVEMENT2” löschen, ändern sich die Bewegungen, wie in Abb. 3-4 dargestellt.
Mit “MOVEMENT1” bewegt sich der Roboter mit einer absoluten Bewegung zu Punkt P3, aber “MOVEMENT2” bewegt den Roboter um V3 mit einer relativen Bewegung von der aktuellen Position.

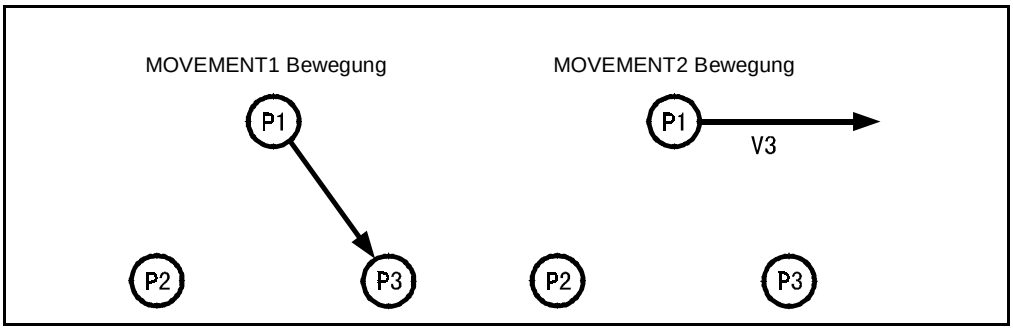


Abb. 3-4 Bewegungsbeispiele aus zwei Programmen (nach dem Löschen)

Anmerkung: Eine relative Bewegung führt eine Bewegung an eine angegebene relative Distanz von der aktuellen Position aus durch. Wenn Sie daher eine relative Bewegung ausführen, kann sich gleich nach dem Überspringen der Bewegungsanweisung mit der Anweisung INTERRUPT ON/OFF (siehe Abschnitt 12.3 “INTERRUPT ON/OFF” in Kapitel 12 dieses Handbuches) die Position der Endposition ändern, je nach dem Timing des ON-Unterbrechungssignals. Um die Position des Bewegungsendes festzulegen, verwenden Sie eine absolute Bewegung.

3.2 Bestätigen der erreichten Position

Es gibt 3 Methoden zur Festlegung des ersten Bewegungsendes, wenn der Roboterarm von einer Bewegung zur anderen wechselt. Sie werden als Durchlaufbewegung, Endebewegung und Codiererwert-Prüfbewegung bezeichnet. Im folgenden wird jede Bewegung erklärt.

3.2.1 Durchlaufbewegung

Bei einer Durchlaufbewegung wird der unmittelbare Bereich einer programmierten Bewegungsposition oder relativen Position durchlaufen.
Wenn Sie die Durchlaufstartverschiebungsdistanz auf “@P” oder “@Numerischer Wert (Numerischer Wert > 0)” für alle Bewegungsbefehle angeben, können Sie eine Durchlaufbewegung ausführen.

3.2.2 Endebewegung

Unter den Bewegungen zum Erreichen einer programmierten Bewegungsposition oder einer relativen Position wird mit der Endebewegung bestimmt, ob der Roboter die Zielposition erreicht hat, wenn der Befehlswert des Servosystems die Zielposition erfüllt.
Wenn Sie die Durchlaufstartverschiebungsdistanz für alle Bewegungsbefehle auf “@0” festlegen, können Sie eine Endebewegung ausführen.

3.2.3 Codiererwert-Prüfbewegung

Unter den Bewegungen zum Erreichen einer programmierten Bewegungsposition oder einer relativen Position wird mit der Codiererwert-Prüfbewegung bestimmt, ob der Roboter die Zielposition erreicht hat, wenn der eingegebene Codiererwert innerhalb eines eingegebenen Impulses (Anfangswert 20) von der Zielposition liegt. Sie können im Programm den angegebenen Impuls für jede Achse ändern.
Wenn Sie die Durchlaufverschiebungsdistanz für alle Bewegungsbefehle auf “@E” festlegen, können Sie eine Codiererwert-Prüfbewegung ausführen.

Anmerkung (1):	**Liegt der Codiererwert außerhalb des angegebenen Impulses von der Zielposition entfernt, stellt der Roboter fest, daß er die Zielposition nicht erreicht hat und fährt nicht mit der nächsten Bewegung fort. Sie müssen den Abweichungswert des Servosystems berücksichtigen, wenn Sie den angegebenen Impuls verringern oder wenn die Nutzlast groß ist.**
Anmerkung (2):	**Im Betrieb mit Maschinenverriegelung werden alle mit “@E” gekennzeichneten Bewegungen mit Bewegungen “@0” ausgeführt. Deshalb liegt die angezeigte Programmausführungszeit unter der einer tatsächlichen Bewegung.**
Anmerkung (3):	**Bei der Beschreibung der Positionsprüfmethode wird die Endbewegung “@0” automatisch auf die Durchlaufstartverschiebungsdistanz eingestellt.**

3.2.4 Bewegungsbeispiele zu Durchlauf, Ende und Codiererwertprüfung

Bei diesen Beispielen handelt es sich um 3 Programme, die den Roboter über Punkt P2 von der aktuellen Position P1 zu Punkt P3 bewegen. Es sind jeweils Programmbeispiele für eine Durchlauf- und Endbewegung sowie zum Bestätigen der Codiererwert-Prüfbewegung.

```
PROGRAM PASS_MOVE
     TAKEARM
     MOVE P, @P P2
     MOVE P, @0 P3
END
```

Abb. 3-5 Durchlaufbewegung Programmbeispiel

```
PROGRAM END_MOVE
     TAKEARM
     MOVE L, @0 P2
     MOVE L, @0 P3
END
```

Abb. 3-6 Endbewegung Programmbeispiel

```
PROGRAM ENCODER_MOVE
     TAKEARM
     MOVE L, @E P2
     MOVE L, @0 P3
END
```

Abb. 3-7 Codiererwert-Prüfbewegung Programmbeispiel

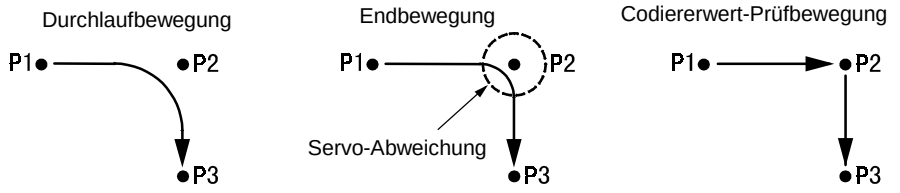


Abb. 3-8 Beispiele für Durchlauf-, End- und Codiererwert-Prüfbewegung

3.2.5 Unterschiede der Ausführungszeit bei Durchlauf-, End- und Codiererwert-Prüfbewegung

Die Durchlaufbewegung hat von allen 3 Bewegungen die schnellste Ausführungszeit, dann folgt die Endebewegung und zum Schluß die Codiererwert-Prüfbewegung.
Die Durchlaufbewegung beginnt mit der Bewegung “Nächste Beschleunigung” während der Verzögerungszeit, bevor P2 erreicht wird, wie in Abb. 3-9 dargestellt. Die Ausführungszeit ist daher kürzer als bei der Endbewegung, die die Verzögerung und Beschleunigung individuell ausführt.
Bei der Codiererwert-Prüfbewegung ist mehr Zeit erforderlich um die Servoabweichung auszuschalten, als bei der Endbewegung, weil der Roboter das Erreichen der Zielposition exakt mit dem Codiererwert prüft.

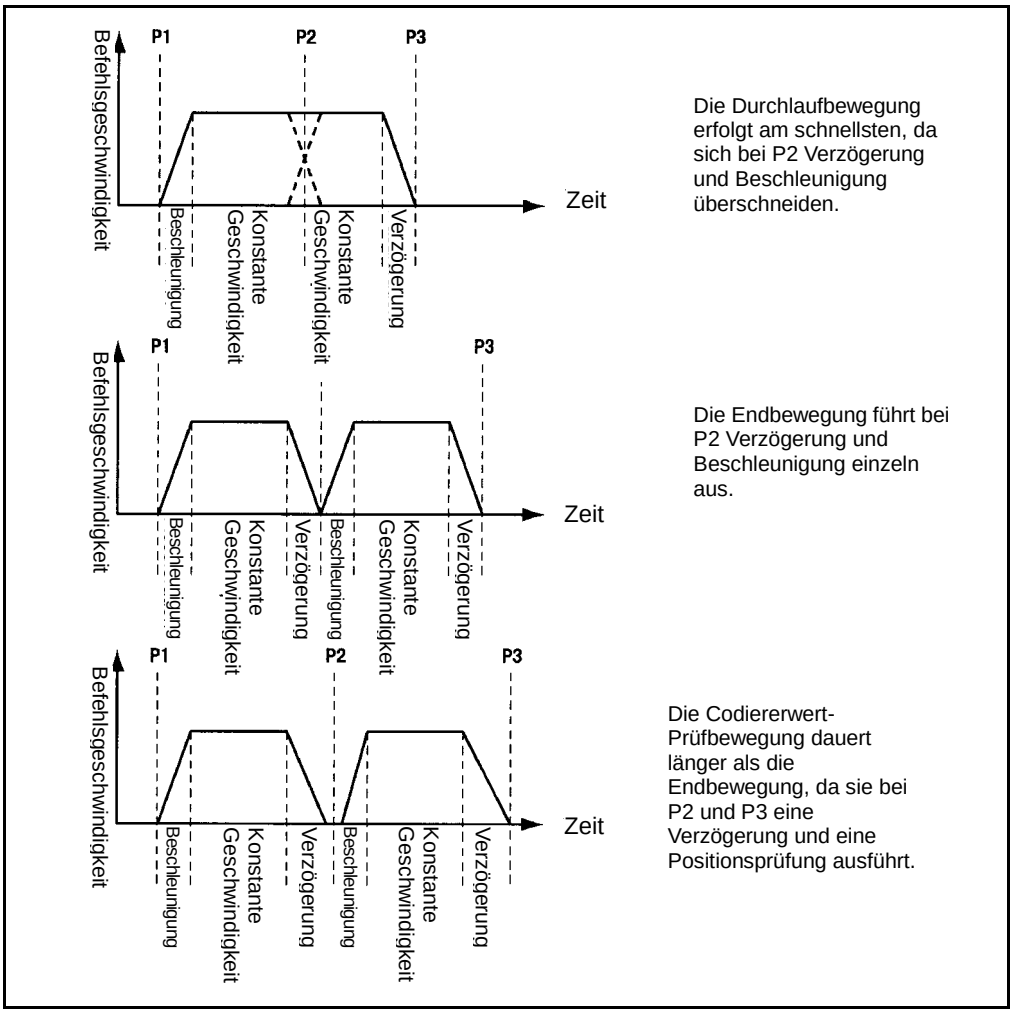


Abb. 3-9 Ausführungszeit von Durchlauf-, End- und Codiererwert-Prüfbewegung

3.2.6 Wenn Durchlaufbewegung nicht ausgeführt wird

In folgenden Fällen bewegt sich der Roboter mit einer Endbewegung, selbst wenn eine Durchlaufbewegung angegeben ist.

3.2.6.1 Wenn der Befehl für die Durchlaufbewegung am Ende des Hauptprogrammes steht

Der am Ende des Hauptprogrammes auszuführende Durchlaufbewegungsbefehl wird wie ein Endbewegungsbefehl ausgeführt. Wie beispielsweise in Abb. 3-10 dargestellt, erreicht der Roboter durch eine Endbewegung die letzte Position P3.

```
PROGRAM PRO10
    TAKEARM
    MOVE L, @P P1
    MOVE L, @P P2
    MOVE L, @P P3
END
```

Abb. 3-10 Programmbeispiel eines in der letzten Zeile des Hauptprogrammes vorliegenden Befehles zur Durchlaufbewegung.

Wenn Sie jedoch fortlaufend das Hauptprogramm ausführen, wird aus P3 eine Durchlaufbewegung wie in Abb. 3-11 dargestellt.

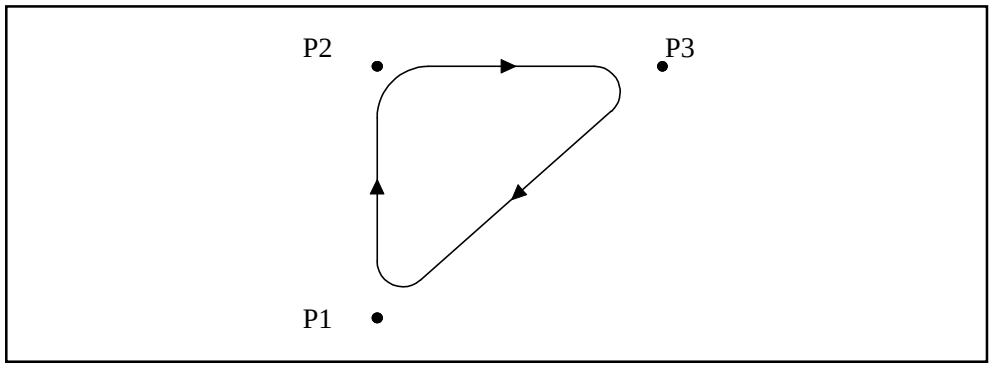


Abb. 3-11 Hauptprogramm wird fortlaufend ausgeführt.

3.2.6.2 Wenn GIVEARM nach einem Befehl zur Durchlaufbewegung vorhanden ist

Wenn Sie GIVEARM ausführen (siehe “GIVEARM”), wartet der Roboter, bis die Bewegung vollständig beendet ist. Wenn Sie GIVEARM daher kurz nach dem Befehl zur Durchlaufbewegung ausführen, führt der Roboter die Durchlaufbewegung nicht aus.

3.2.7 Wenn der Effekt der Durchlaufbewegung reduziert ist

3.2.7.1 Wenn ein Nicht-Bewegungsbefehl nach der Durchlaufbewegung vorhanden ist

Wenn ein Nicht-Bewegungsbefehl zwischen Durchlaufbewegungsbefehl und nächstem Bewegungsbefehl vorhanden ist, reduzieren sich die Effekte einer verkürzten Ausführungszeit der Durchlaufbewegung. Ein Nicht-Bewegungsbefehl bewirkt, daß sich der Roboter nicht bewegt.
Ein Beispiel eines Nicht-Bewegungsbefehls zwischen Durchlaufbewegungsbefehl und nächstem Bewegungsbefehl wird in Abb. 3-12 dargestellt. Wie in Abb. 3-13 dargestellt, wird der Nicht-Bewegungsbefehl beispielsweise während der Verzögerungszeit des Pfadbewegungsbefehls ausgeführt. Der Effekt der reduzierten Ausführungszeit der Durchlaufbewegung wird dadurch verringert.

```
PROGRAM PRO13
   MOVE L, @P P2     ' Durchlaufbewegungsbefehl
   SET IO1           ' Nicht-Bewegungsbefehl
   RESET I02         ' Nicht-Bewegungsbefehl
       :
   MOVE L, @0 P3     ' Endbewegungsbefehl
END
```

Abb. 3-12 Beispiel eines Nicht-Bewegungsbefehls zwischen Durchlaufbewegungsbefehl und nächstem Bewegungsbefehl

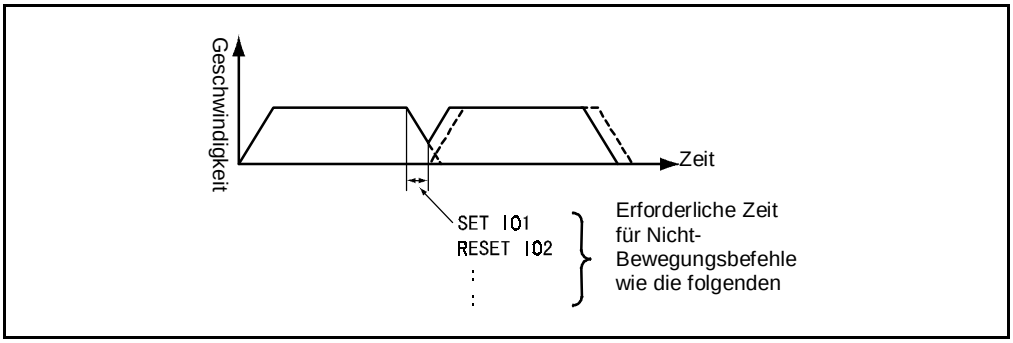


Abb. 3-13 Beispiel eines reduzierten Effektes bei der Verkürzung der Ausführungszeit einer Durchlaufbewegung

3.2.7.2 Wenn der Pfad nach dem Durchlauf kurz ist

Wenn das Geschwindigkeitsmuster aufgrund eines kurzen Pfades nach dem Durchlauf ein Dreieck bildet, verzögert sich die Durchlaufstartposition, so daß die Durchlaufbewegung nach Abschluß der Pfadverzögerung endet. Wenn die Pfadverzögerung vor dem Durchlauf so klein ist wie in Abb. 3-14 dargestellt, verringert sich deshalb der Durchlaufeffekt.

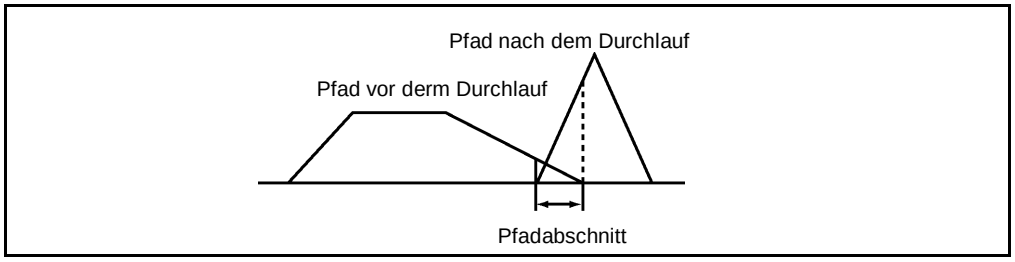


Abb. 3-14 Dreiecksmuster aufgrund eines kurzen Pfades nach dem Durchlauf

3.2.8 Wenn die Beschleunigung den Durchlaufbewegungspfad beeinflußt

Roboter der VS-D-Reihe legen automatisch die Geschwindigkeit zum Quadrat geteilt durch 100 für Beschleunigung und Verzögerung fest, wenn Sie die Geschwindigkeit einstellen.
Wenn Sie den SPEED-Befehl genauso wie oben verwenden, wird die Beschleunigung und Verzögerung eingestellt.
Wenn der Roboter mit der automatisch festgelegten Beschleunigung eine Durchlaufbewegung ausführt, ist sein Bewegungspfad immer konstant. Ein Beispiel, bei dem die Geschwindigkeit auf 60% und 80% eingestellt ist, wird in Abb. 3-15 dargestellt.

⚠VORSICHT: Wenn Sie die Beschleunigung festlegen, stellen Sie sicher, daß keine Gefahr einer Kollision durch die Änderung des Duchlaufbewegungspfads besteht.

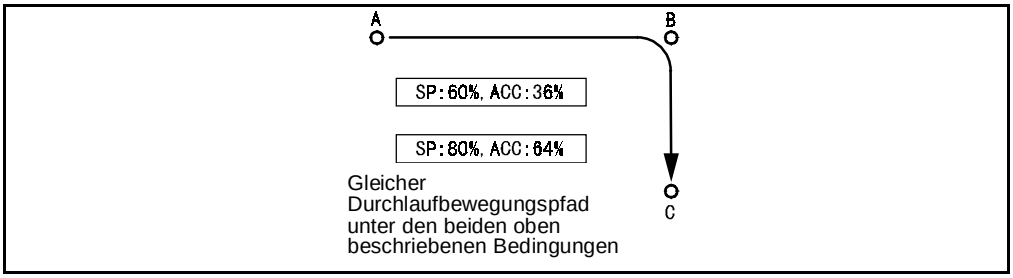


Abb. 3-15 Beispiel für automatische Beschleunigungseinstellung

Wenn Sie die Beschleunigung beliebig einstellen, kann sich evtl. der Bewegungspfad ändern. Ein Beispiel mit 60% Geschwindigkeit bei 100% Beschleunigung und 10% Verzögerung wird in Abb. 3-16 dargestellt.

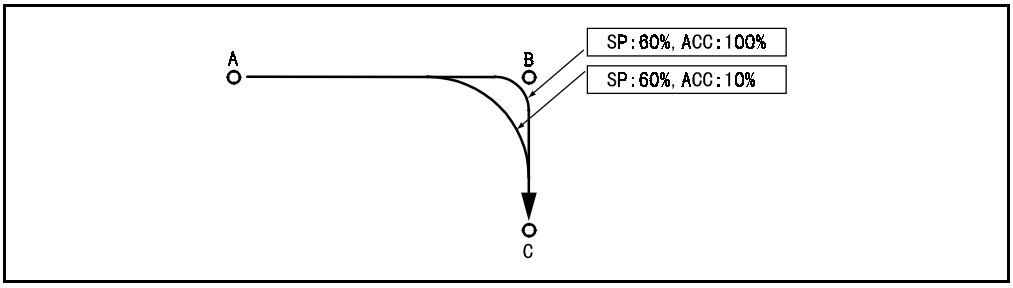


Abb. 3-16 Beispiel für beliebige Beschleunigungseinstellung

⚠VORSICHT: Wenn Sie bei der Verzögerung der Durchlaufbewegung einen sofortigen Stop ausführen und die Bewegung neu starten, bewegt sich der Roboter zur Position des Befehlswertes des nächsten Schrittes. Gehen Sie deshalb im Umgang mit Geschwindigkeit, Beschleunigung und Verzögerung mit größter Sorgfalt vor.

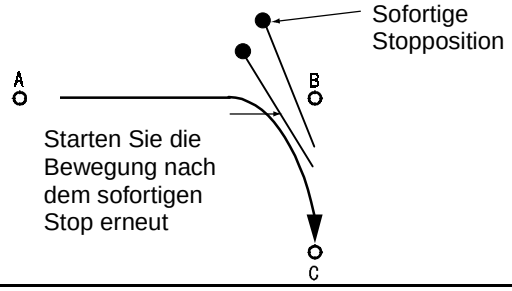


Abb. 3-17 Starten Sie die Bewegung nach dem sofortigen Stop erneut

3.2.9 Durchlaufstartverschiebung

Wenn Sie eine Durchlaufstartverschiebung mit “@P” angeben, beginnt die nächste Bewegung zusammen mit dem Beginn der Verzögerung der Bewegung, die bis zu diesem Zeitpunkt ausgeführt wird. Deshalb ändert sich je nach Geschwindigkeit und Beschleunigung die Distanz L zwischen der Pfadstartposition und dem Durchlaufpunkt B.

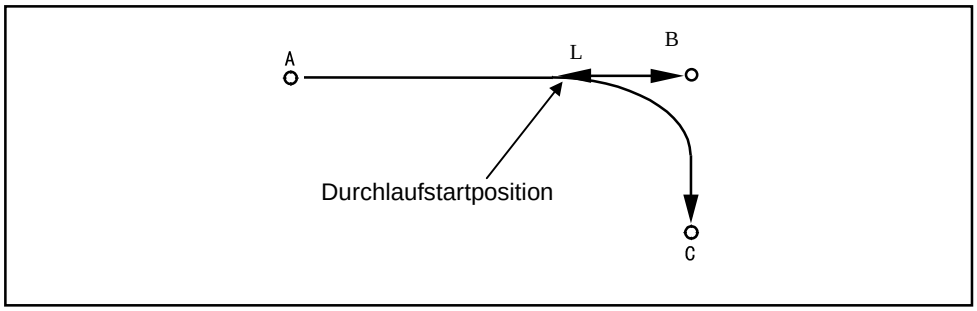


Abb. 3-18 Durchlaufstartposition

Wenn Sie eine Durchlaufstartverschiebung mit “@ numerischer Wert” angeben, beginnt die Durchlaufbewegung an einer Position mit einem Wert = L (mm).
Wenn der mit “@ numerischer Wert” angegebene Wert die Distanz der Verzögerungsbewegung überschreitet, wird die Verzögerungsstartposition korrigiert wie in Abb. 3-19 dargestellt.
Bei der Angabe “@ numerischer Wert” wird daher die Verzögerung reduziert, und der Bewegungspfad kann länger sein.

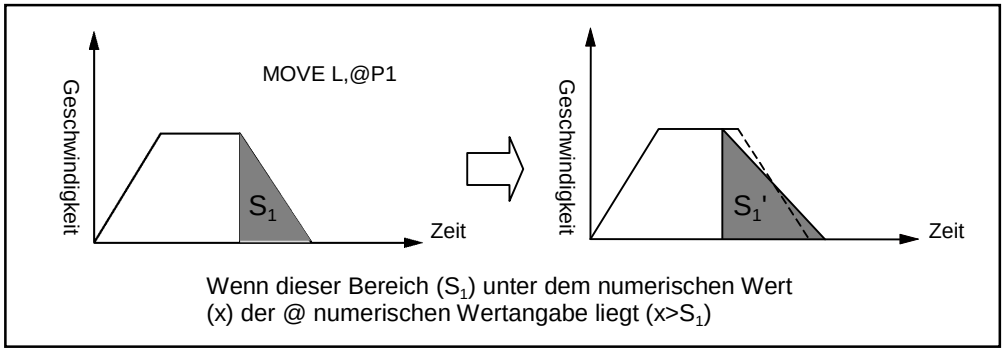


Abb. 3-19 Verzögerungsstartposition

Der mit “@ numerischer Wert” angegebene Wert überschreitet die Hälfte der Distanz der Verzögerungsbewegung; die Verzögerungsstartposition ist auf die halbe Bewegungsdistanz festgelegt, wie in Abb. 3-20 dargestellt.

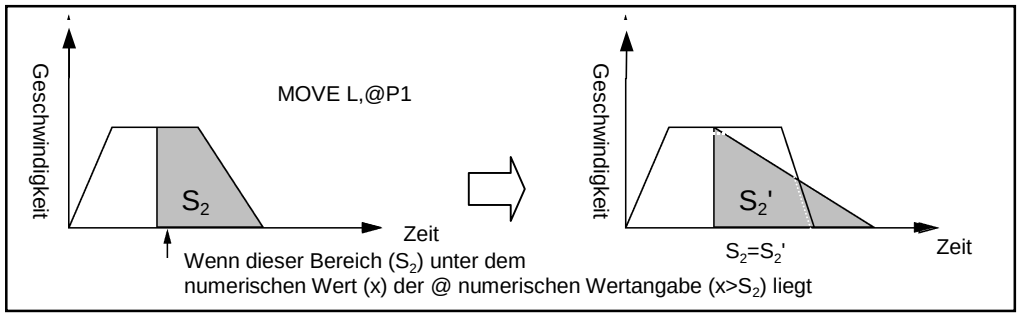


Abb. 3-20 Wenn der (@ numerische Wert) die halbe Bewegungsdistanz überschreitet

Wenn X>S2 ist, ändert sich deshalb niemals die Durchlaufbewegung, selbst wenn X verändert wird.
Und wenn X>S2, wird die Warnung “Pfadstartverschiebung erneut eingeben” am Programmiergerät angezeigt.

Wenn sich die Verzögerungsstartposition ändert, ändert sich auch die Beschleunigungszeit des nächsten Pfads. Wie in Abb. 3-21 dargestellt, kann die Bewegungszeit im Vergleich zur Angabe @P länger sein.

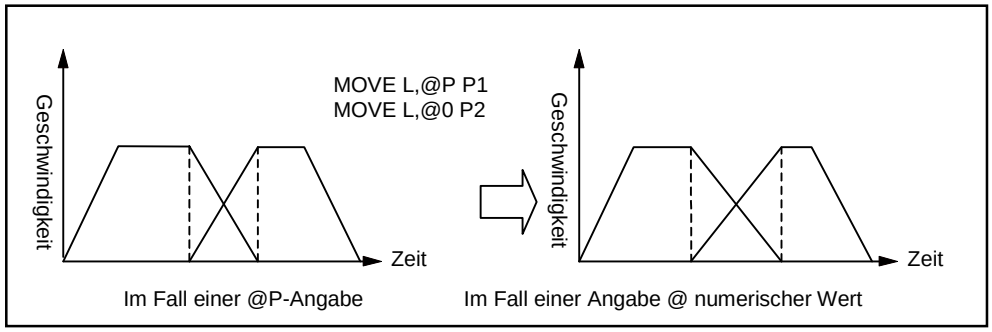


Abb. 3-21 Bewegungszeit wenn sich die Verzögerungsstartposition ändert.

Anmerkung: Im folgenden Fall beginnt die Durchlaufbewegung an der Position, die kleiner ist als die Position von numerischer Wert = L (mm), auch wenn Sie “@ numerischer Wert” angeben.

(1) Wenn aus dem Geschwindigkeitsmuster aufgrund seiner kurzen Kurve ein Dreieck wird.
Ähnlich wie in “Wenn der Pfadbewegungseffekt reduziert wird” ändert sich die Startposition für die Bewegung. Insbesondere wenn der Roboter die Pfadbewegung ausführt und sich die Bewegung dabei von niedriger zu schneller Geschwindigkeit ändert, kann sich die Startposition der Durchlaufbewegung ändern. Legen Sie bei der Durchlaufbewegung vor und nach der Bewegung eine konstante interne Geschwindigkeit fest.

3.3 Interpolationssteuerung

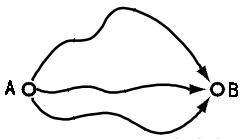
Wenn sich der Roboterarm bewegt, gibt es nicht nur einen Pfad. Sie können durch Kombination der Achsen verschiedene Pfade erstellen. Sie können den Roboter auch so steuern, daß er linien- oder kreisförmige Pfade beschreibt. Nachfolgend finden Sie eine Erklärung der Steuerungsmethoden entsprechend den Bewegungspfadtypen.
Verwenden Sie die unten stehenden Befehle, um eine Interpolationsmethode anzugeben.

Um Interpolationsmethoden zuzuweisen, stehen folgende Befehle zur Verfügung:

APPROACH, DEPART, DRAW, MOVE

3.3.1 Punkt-zu-Punkt-Steuerung

PTP (Punkt-zu-Punkt) kann als Bewegung von einem Punkt zum anderen definiert werden. Der Bewegungspfad des Roboters ist von der Roboterhaltung abhängig und nicht immer eine gerade Linie.
Ein Beispiel einer Punkt-zu-Punkt-Bewegung wird in Abb. 3-22 dargestellt. Wenn Sie bei der Angabe der Interpolationsmethode mit dem Bewegungssteuerbefehl “P” festlegen, führt der Roboter die Punkt-zu-Punkt-Bewegung aus.
Wenn Sie Variablen des Typs P oder T als Zielposition der PTP-Bewegung angeben und die Roboterarmstellung festlegen, bewegt sich der Roboter, bis er die angegebene Roboterarmstellung erreicht hat. Wenn Sie keine Roboterarmstellung angeben, bleibt die aktuelle Armstellung bestehen.



Der Bewegungspfad ist nicht immer eine gerade Linie (Beispiel zu MV)

Abb. 3-22 Punkt-zu-Punkt-Steuerungsbewegung

3.3.2 Linearsteuerung

Die Linearsteuerung kontrolliert die Interpolation, damit der Pfad zum Erreichen der Bewegungszielposition eine gerade Linie ergibt.
Ein Beispiel einer Linearsteuerungsbewegung wird in Abb. 3-23 dargestellt.
Wenn Sie bei der Angabe der Interpolationsmethode mit dem Bewegungssteuerbefehl “L” festlegen, führt der Roboter die Linearbewegung aus.

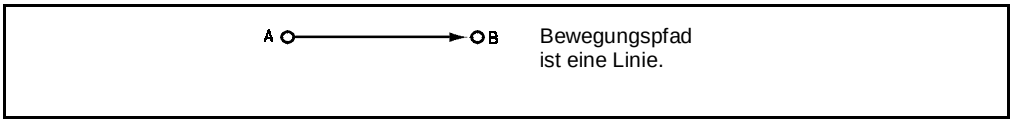


Abb. 3-23 Linearsteuerungsbewegung

- Wenn die Linearbewegung ausgeführt wird, kann der Roboter nicht einfach die Position einer anderen Armstellung aus der aktuellen Armstellung ausführen. Wenn Sie eine andere Armstellung angeben, tritt eventuell ein Fehler “607F Fehler bei Roboter-Armstellung” auf. Ist die Bewegung jedoch möglich, sollte dieser Fehler nicht auftreten.
- Eine Armstellung, die der aktuellen ähnelt, wird bei der Ausführung der Linearsteuerung als Roboter-Armstellung ausgewählt. Selbst wenn Sie die Roboterarmstellung mit einer Variablen des Typs P oder T angeben, müssen Sie deshalb nicht zwangsläufig die angegebene Armstellung erhalten. Wenn sich die Armstellung von der angegebenen unterscheidet, wird eventuell eine Warnung “601C Armstellung ändern” angezeigt.
- Wenn Sie den ersten Bewegungsbefehl in einem Programm mit der Linearsteuerung ausführen, ist eventuell - je nach der Position des Roboters - keine Bewegung verfügbar. Daher empfiehlt sich für den ersten Bewegungsbefehl im Programm eine Punkt-zu-Punkt-Steuerung.

3.3.3 Bogeninterpolationssteuerung

Die Bogeninterpolationssteuerung steuert die Interpolation so, daß der Pfad zum Erreichen der Bewegungszielposition einen Bogen bildet.
Ein Beispiel einer durch Bogeninterpolation gesteuerten Bewegung ist in Abb. 3-24 dargestellt.
Wenn Sie bei der Angabe der Interpolationsmethode mit dem Bewegungssteuerbefehl “C” festlegen, führt der Roboter eine Bogeninterpolationsbewegung aus.

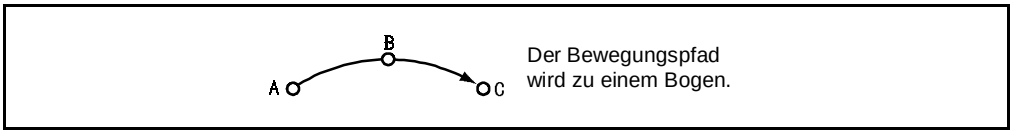


Abb. 3-24 Bogeninterpolations-Steuerungsbewegung

- Wenn eine Bogeninterpolationssteuerung ausgeführt wird, kann der Roboter nicht einfach aus der aktuellen Armstellung in die Position einer anderen Armstellung wechseln, wie bei der Linearbewegung. Wenn Sie eine andere Armstellung angeben, tritt eventuell ein Fehler “607F Fehler bei Roboter-Armstellung” auf. Ist die Bewegung jedoch möglich, sollte dieser Fehler nicht auftreten.
- Eine Armstellung, die der aktuellen ähnelt, wird bei der Ausführung der Bogeninterpolationssteuerung als Roboter-Armstellung ausgewählt. Selbst wenn Sie die Roboterarmstellung mit einer Variablen des Typs P oder T angeben, müssen Sie deshalb nicht zwangsläufig die angegebene Armstellung erhalten. Wenn sich die Armstellung von der angegebenen unterscheidet, wird eventuell eine Warnung “601C Armstellung ändern” angezeigt.
- Wenn Sie den ersten Bewegungsbefehl in einem Programm mit der Bogeninterpolationssteuerung ausführen, ist die Bewegung eventuell - je nach der Position des Roboters - nicht verfügbar. Daher empfiehlt sich für den ersten Bewegungsbefehl im Programm eine Punkt-zu-Punkt-Streuerung.

3.4 Wenn Ausgabebefehl nach der Bewegungsanweisung vorhanden ist

Wenn der Bewegungsbefehl ausgeführt wird, wird die aktuelle Position des Roboters normalerweise verzögert, während er zur Position bewegt wird, die durch die Robotersteuerung vorgegeben wurde.
Bei der Endbewegung fährt die Robotersteuerung mit dem nächsten Befehl fort, nachdem die vorgegebene Position die Zielposition des Bewegungsbefehls erreicht hat. Gibt es nach dem Bewegungsbefehl einen Ausgabebefehl, kann deshalb der Ausgabebefehl möglicherweise ausgeführt werden, bevor der Roboter die Zielposition erreicht.
Es gibt folgende Methoden, um den nächsten Befehl auszuführen, nachdem der Roboter die Zielposition erreicht hat:

- Stellen Sie die Wartezeit mit einem Befehl DELAY ein (Ein Programmbeispiel hierzu ist in Abb. 3-25 dargestellt).
- Verwenden Sie keine Endbewegung. Führen Sie stattdessen eine Codiererwert-Prüfbewegung aus. Ein Programmbeispiel ist in Abb. 3-26 dargestellt.

Anmerkung: Bei der Codiererwert-Prüfbewegung kann der Codiererwert evtl. nicht die Zielposition erreichen, falls die Nutzlast schwer ist.

```
PROGRAM PRO13
       :
       :
    MOVE L, @0 P3     'Endbewegungsbefehl
    DELAY 500         'Wartet 0,5 Sekunden
    SET IO1           'Stellt E/A1 auf ON.
    RESET IO2         'E/A2 auf OFF.
       :
       :
END
```

Abb. 3-25 Programmbeispiel zum Einstellen der Wartezeit wegen eines DELAY-Befehls

```
PROGRAM PRO13
       :
       :
    MOVE L, @E P3     'Codiererwert-
    SET IO1           'Prüfbewegungsbefehl
    RESET IO2         'Stellt E/A1 auf ON.
       :              'Stellt E/A2 auf OFF.
       :
END
```

Abb. 3-26 Programmbeispiel einer Codiererwert-Prüfbewegung

3.5 Nachgiebigkeits-Steuerungsfunktion

3.5.1 Überblick

Die Nachgiebigkeits-Steuerungsfunktion bietet Nachgiebigkeit für einen Roboter über die Software. Diese Funktion absorbiert eine Positionsverschiebung beim Umgang mit dem Roboter und verhindert, daß auf einen Roboter oder ein Werkstück eine zu große Kraft angewendet wird. Diese Funktion ist in zwei Teile untergliedert: das Einstellen der Nachgiebigkeit für einzelne Gelenke (Strombegrenzungsfunktion) und die Nachgiebigkeitsfunktion für einzelne Elemente des Koordinatensystems an der Spitze eines Roboters (Spitzen-Nachgiebigkeits-Steuerfunktion).
Die Stromgrenzwertfunktion ist verfügbar für Ver. 1.20 oder neuer, die Spitzen-Nachgiebigkeits-Steuerfunktion ist verfügbar für den vertikal ausgerichteten Roboter Ver. 1.40 oder neuer (VM-6070D nicht enthalten).

3.5.2 Stromgrenzwertfunktion für einzelne Achsen [V1.2 oder neuer]

Diese Funktion stellt die Nachgiebigkeit für einzelne Achsen ein. Durch die Begrenzung des Drehmoments von Motoren für einzelne Achsen wird eine Nachgiebigkeit erzielt. Sie können diese Funktion insbesondere verwenden, um zu verhindern, daß eine zu große Kraft auf einen Roboter oder ein Werkstück einwirkt, oder um einen Ausfall durch eine Überlastung oder einen zu starken Strom zu vermeiden.

3.5.2.1 Verwenden der Stromgrenzwertfunktion

Sie können die Bibliotheksfunktion verwenden und die Bibliotheken für die Strombegrenzung (SetCurLmt, ResetCurLmt) ausführen, um diese Funktion zu aktivieren oder zu deaktivieren. Stellen Sie sicher, daß Sie die folgenden Einstellungen bereitstellen. Weitere Informationen zur Bibliothek finden Sie im Abschnitt zur PAC-Bibliohek.

(1) Spezifikation für die Spitzen-Nutzlast (erforderlich)
Sie müssen die genaue Masse der Nutzlast und den Nutzlastschwerpunkt der Spitze angeben. Sie müssen Werte für eine Hand angeben und ein Werkstück, das Sie auf die Masse der Nutzlast [g], den Nutzlastschwerpunkt X6 [mm], den Nutzlastschwerpunkt Y6 [mm] und den Nutzlastschwerpunkt Z6 [mm] anwenden. Wenn sich die Masse beim Einspannen oder Loslassen eines Werkstücks ändert, verwenden Sie aspACLD zum Ändern der Einstellung.
Weitere Informationen finden Sie in der Beschreibung zur Einstellung der optimalen Lastkapazität im Abschnitt 4.7 “Einstellen der Master-Steuer-parameter in den Benutzereinstellungen” in Kapitel 4 dieses Handbuches.
Anmerkungen: Wenn Sie keine exakten Werte angeben, fällt Ihr Roboter eventuell in die Richtung der Schwerkraft, wenn Sie den Stromgrenzwert eingeben.

(2) Spezifikation Schwerkraft-Offset (erforderlich)
Sie müssen die Spezifikation für den Schwerkraft-Offset angeben. Andernfalls wird der Fehler “665a Aktuelle Begrenzungsspezifikation ist nicht verfügbar” ausgegeben. Zum Aktivieren des Schwerkraft-Offset verwenden Sie das Programmiergerät; setzen Sie die Einstellung für Aktivieren/Deaktivieren des Schwerkraft-Offset auf 1. Gehen Sie hierzu anhand der folgenden Schritte vor:
1) Drücken Sie in der Eingangsanzeige des Programmiergeräts [F2 Arm]→[F6 Hilfs.]→[F7 Konfig.]. Daraufhin wird das Konfigurationsparameterfenster angezeigt.
2) Drücken Sie [F3 Springen] und geben Sie 24 ein. Daraufhin wird die Einstellungszeile für den Schwerkraft-Offset angezeigt.
3) Geben Sie 1 für die Einstellung zum Aktivieren/Deaktivieren des Schwer-kraft-Offset ein und drücken Sie OK.

Wenn Sie 1 für die Einstellung zum Aktivieren/Deaktivieren des Schwerkraft-Offset eingeben, ist der Schwerkraft-Offset nach Abschluß der Kalibrierung immer aktiviert.

Sie können zwar auch SetGravity und ResetGravity zum Aktivieren/Deaktivieren des Schwerpunkt-Offset verwenden, Ihr Roboter kann jedoch eine kleine Bewegung aufweisen, während er zwischen den Modi Aktiviert und Deaktiviert umschaltet. Wir empfehlen die Verwendung der Konfiguration des Programmiergeräts zum Einstellen der Aktivierung/Deaktivierung des Schwerkraft-Offset auf 1 und den Aufruf von SetGravity, sobald Ihr Initialisierungsprogramm angezeigt wird.

(3) Schwerkraft-Offset Kompensierung (falls erforderlich)
Der Unterschied zwischen der Einstellung der Masse der Nutzlast an der Spitze und der tatsächlichen Masse der Nutzlast kann zu einem Fehler bei der Schwerkraft-Balance entsprechend dem Schwerkraft-Offset führen. Wenn Sie eine niedrigere Strombegrenzung für eine Achse festlegen, für die die Schwerkraft angewendet wird, fällt Ihr Roboter eventuell in die Richtung der Schwerkraft. Die Funktion zur Kompensierung der Schwerkraft vergleicht das Drehmoment im stationären Status Ihres Roboters mit dem Drehmoment am Schwerkraft-Offset, um den Fehler auszugleichen. Wenn Sie den Stromgrenzwert auf 30% oder weniger für eine Achse festlegen, auf die die Schwerkraft angewendet wird, legen Sie immer die Kompensierung des Schwerkraft-Offset fest.
Sie müssen SetGrvOffset ausführen, damit der Kompensierungswert berechnet wird, wenn der Roboter gestoppt wird. Der Wert, den Sie erhalten, gilt für die Haltung bei der Ausführung von SetGrvOffset. Sie sollten SetGrvOffset erneut ausführen, wenn sich die Haltung deutlich von der kompensierten unterscheidet.
Die Ausführung von SetGrvOffset kann zu einem Kompensierungswert mit einem Fehler führen, wenn eine externe Kraft wie beispielsweise ein Kontakt auf den Roboter angewendet wird. Führen Sie SetGrvOffset aus, wenn keine externen Kräfte (mit Ausnahme der Schwerkraft) wirken.

(4) Spezifikation der zulässigen Abweichung (falls erforderlich)
Wenn eine externe Kraft angewendet wird, während der Strom begrenzt ist, dreht sich das Gelenk in die Richtung der Kraft. Dadurch wird die Abweichung des Gelenkwinkels verstärkt; dies kann zu einem Fehler “612* J *Zu starke Abweichung” führen. Wenn dies der Fall ist, verwenden Sie SetEralw zum Einstellen der Spezifikation für die zulässige Abweichung. Die Funktionsprüfung der Abweichung wird aus Sicherheitsgründen bereitgestellt. Sie sollten den Bereich nicht vergrößern.

(5) Prozedur zur Ausführung der Bibliothek

■ Wenn der Stromgrenzwert aktiviert ist

1) Verwenden Sie das Programmiergerät zum Einstellen der Aktivierung/ Deaktivierung des Schwerkraft-Offset auf 1 in der Konfiguration.
2) Fügen Sie Ihrem Initialisierungsprogramm die folgende Anweisung hinzu.
   CALL SetGravity
3) Aktivieren Sie die Strombegrenzung. Geben Sie ggf. (Anweisung) an.
   Warten Sie, bis eine Aktion abgeschlossen ist.
   (CALL SetGrvOffset) ‘Berechnen des Kompensationswertes für den Schwerkraft-Offset
   CALL SetCurLmt(n1, m1) ‘Einstellen des Stroms für n1-Achse auf m1 (%) der Bewertung
   (CALL SetEralw(n2, m2)) ‘Einstellen des Stroms für n2-Achse auf m2 (%) der Bewertung

■ Wenn der Stromgrenzwert deaktiviert ist
   CALL ResetCurLmt(0) ‘Zurücksetzen der Stromgrenzwerte für alle Achsen und Initialisieren der zulässigen Abweichung
   (CALL ResetGrvOffset) ‘Initialisieren der Kompensierungswerte für den Schwerkraft-Offset

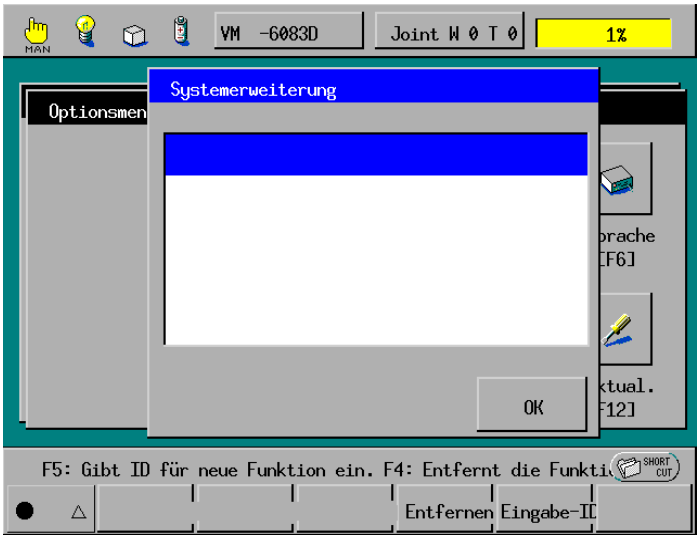
3.5.3 Steuerfunktion für die Spitzen-Nachgiebigkeit [V1.4 oder neuer]

Diese Funktion definiert die Nachgiebigkeit für die einzelnen Elemente des Koordinatensystems an der Spitze eines Roboters. Das Steuern der Motor-Drehmomentwerte für einzelne Achsen auf der Basis der Kraftbegrenzung an der Spitze ergibt die Nachgiebigkeit der Spitze. Sie können ein Koordinatensystem aus dem Basis-, dem Werkzeug- oder dem Arbeitskoordinatensystem auswählen. Sie können diese Funktion verwenden, um Ihren Roboter entsprechend einer externen Kraft in einer bestimmten Richtung auszurichten oder um Ihren Roboter gegen ein Objekt drücken zu lassen, um eine Höhe zu prüfen.

3.5.3.1 Aktivieren der Funktion

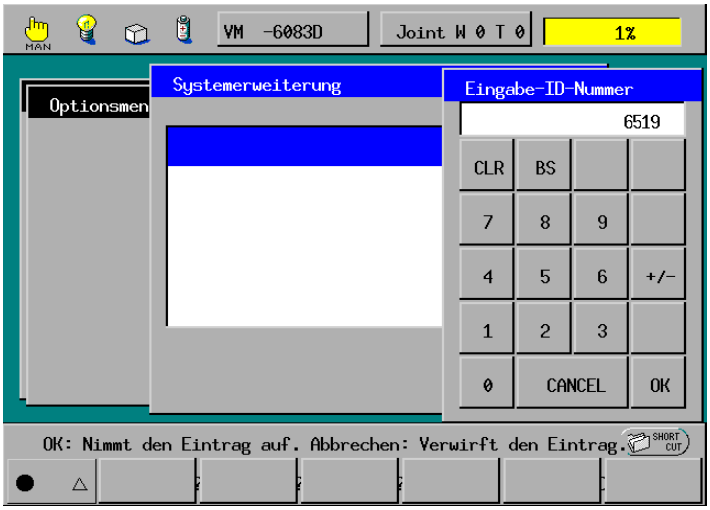
Bei dieser Funktion handelt es sich um eine erweiterte Funktion. Sie sollten daher das Programmiergerät verwenden, um die Nachgiebigkeits-Steuer-funktion zu aktivieren. Sobald Sie eine erweiterte Funktion aktivieren, wird die Einstellung beibehalten, nachdem Sie den Controller ausschalten. Sie brauchen die Einstellung nicht erneut vorzunehmen. Führen Sie zum Aktivieren der Nachgiebigkeits-Stsuerfunktion die nachfolgend beschriebenen Schritte aus.

(1) Rufen Sie die Systemerweiterungsanzeige des Programmiergeräts auf.
Bedienungsablauf: Hauptmenü -[F6 Einstellen]-[F7 Option]-[F8 Erweitern]



(2) Drücken Sie [F5 Eingabe-ID], um eine Funktion hinzuzufügen. Daraufhin wird das ID-Nummernfenster angezeigt.

(3) Geben Sie die ID-Nummer ein. Die ID-Nummer lautet 6519.



(4) Drücken Sie OK, überprüfen Sie die angezeigte Systemmeldung und drücken Sie erneut OK.

3.5.3.2 Verwenden der Steuerfunktion für die Spitzennachgiebigkeit

Sie können die Bibliotheksfunktion verwenden und die Bibliotheken für die Nachgiebigkeitssteuerung (SetCompControl, ResetCompControl) ausführen, um diese Funktion zu aktivieren oder zu deaktivieren. Stellen Sie sicher, daß Sie die folgenden Einstellungen bereitstellen. Weitere Informationen zur Bibliothek finden Sie im Abschnitt zur PAC-Bibliohek.

(1) Spezifikation für die Spitzen-Nutzlast (erforderlich)
Siehe 3.5.2.1. “Verwenden der Stromgrenzwertfunktion”.

(2) Spezifikation Schwerkraft-Offset (erforderlich)
Siehe 3.5.2.1. “Verwenden der Stromgrenzwertfunktion”.

(3) Nachgiebigkeits-Steuerparameter (erforderlich)
Die folgenden fünf Parameter sind verfügbar.

1) Parameter zur Auswahl des Koordinatensystems für die Kraftbegrenzung
Wählen Sie das Basis-, das Werkzeug- oder das Arbeitskoordinatensystem als Koordinatensystem für die Kraftbegrenzung aus. Verwenden Sie zur Spezifikation die Bibliothek für die Nachgiebigkeitssteuerung (SetFrcCood).

2) Kraftbegrenzungsrate
Geben Sie die Rate für die Kraftbegrenzung für die einzelnen Elemente (Umsetzungen entlang und Drehungen um die X, Y und Z-Achsen) des unter 1) angegebenen Koordinatensystems an. Verwenden Sie zur Spezifikation die Bibliothek für die Nachgiebigkeitssteuerung (SetFrcLimit). Wenn Sie SetFrcLimit angeben, sind 3) die Nachgiebigkeitsrate und 4) die Dämpfungsrate gleich.

3) Nachgiebigkeitsrate (falls erforderlich)
Geben Sie die Nachgiebigkeitsrate für die einzelnen Elemente (Umsetzungen entlang und Drehungen um die X, Y und Z-Achsen) des unter 1) angegebenen Koordinatensystems an. Verwenden Sie zur Spezifikation die Bibliothek für die Nachgiebigkeitssteuerung (SetCompRate). Wenn Sie SetCompRate angeben, ist 4) die Dämpfungsrate gleich.

4) Dämpfungsrate (falls erforderlich)
Geben Sie die Dämpfungsrate für die einzelnen Elemente (Umsetzungen entlang und Drehungen um die X, Y und Z-Achsen) des unter 1) angegebenen Koordinatensystems an. Verwenden Sie zur Spezifikation die Bibliothek für die Nachgiebigkeitssteuerung (SetDampRate). Sie verwenden diese Funktion nur selten außer im Geschwindigkeitssteuermodus, einer speziellen Nachgiebigkeits-Steuerfunktion. Geben Sie keine Werte an, die kleiner sind als die Nachgiebigkeitsraten, die mit SetFrcLimit und SetCompRate angegeben wurden.

5) Zulässige Positions-/Haltungsabweichung unter Nachgiebigkeitssteuerung (falls erforderlich)
Geben Sie die zulässige Positions-/Haltungsabweichung für die einzelnen Elemente (Umsetzungen entlang und Drehungen um die X, Y und Z-Achsen) des unter 1) angegebenen Koordinatensystems an. Verwenden Sie zur Spezifikation die Bibliothek für die Nachgiebigkeitssteuerung (SetCompEralw). Der Anfangswert für die Positionsabweichung beträgt 100 (mm), die Haltungsabweichung beträgt 30 (Grad). Die Funktionsprüfung der Abweichung wird aus Sicherheitsgründen bereitgestellt. Sie sollten diesen Wertebereich nicht unnötig vergrößern.

(4) Schwerkraft-Offset Kompensierung (falls erforderlich)
Die Funktion zur Kompensierung der Schwerkraft vergleicht das Drehmoment im stationären Status Ihres Roboters mit dem Drehmoment am Schwerkraft-Offset, um den Fehler auszugleichen, sofern SetCompControl angegeben ist. Die Ausführung von SetCompControl kann zu einem Fehler beim Kompensierungswert führen, wenn eine externe Kraft wie beispielsweise ein Kontakt auf den Roboter angewendet wird. Führen Sie SetCompControl aus, wenn keine externen Kräfte (mit Ausnahme der Schwerkraft) wirken. Verwenden Sie SetGrvOffset und SetCompFControl und führen Sie die Bibliotheken wie nachfolgend beschrieben aus, wenn Sie die Nachgiebigkeitssteuerung unter externen auf den Roboter angewendeten Kräften aktivieren wollen.

(SetCompControl = SerGrvOffset + SetCompFControl)
Während Ihr Roboter stationär ist und keine Kontakte (Interferenzen) zu externen Objekte hat, führen Sie folgendes aus:
CALL SetGrvOffset
Bewegen Sie Ihren Roboter zu einem Kontakt (Interferenz) mit externen Objekten
CALL CompFControl

(1) Prozedur zur Ausführung der Bibliothek
■ Wenn die Nachgiebigkeitssteuerung aktiviert ist
1) Verwenden Sie das Programmiergerät zum Einstellen der Aktivierung/ Deaktivierung des Schwerkraft-Offset auf 1 in der Konfiguration.
2) Fügen Sie Ihrem Initialisierungsprogramm die folgende Anweisung hinzu.
CALL SetGravity
3) Aktivieren Sie die Nachgiebigkeitssteuerung. Geben Sie ggf. (Anweisung) an.
CALL SetFrcCoord(n) ‘Koordinatensystem für Kraftbegrenzung auf n setzen
CALL SetFrcLimit(fx, fy, fz, mx, my, mz)
‘Rate der Kraftbegrenzung auf (fx, fy, fz, mx, my, mz) [%] setzen
(CALL SetCompRate(cx, cy, cz, crx, cry, crz))
‘Rate der Nachgiebigkeit auf (cx, cy, cz, crx, cry, crz) [%] setzen
(CALL SetDumpRate (dx, dy, dz, drx, dry, drz))
‘Dämpfungsrate auf (dx, dy, dz, drx, dry, drz) [%] setzen
CALL SetCompControl ‘Nachgiebigkeitssteuerung aktivieren
(CALL SetCompEralw (ex, ey, ez, erx, ery, erz))
‘Zulässige Positionsabweichung auf (ex, ey, ez) (mm) einstellen und die zulässige Haltungsabweichung auf (erx, ery, erz) (Grad).
(CALL SetEralw (n1, m1))
‘Zulässige Abweichung für n1-Achse auf m1 (Grad) einstellen

■ Wenn die Nachgiebigkeitssteuerung deaktiviert ist. Geben Sie ggf. (Anweisung) an.

CALL ResetCompControl	‘Nachgiebigkeitssteuerung deaktivieren
CALL ResetCompEralw	‘Positionsabweichung initialisieren
(Call ResetEralw(0))	‘Winkelabweichung initialisieren

3.5.3.3 Verwenden der Bibliothek zur Steuerung der Spitzennachgiebigkeit

Sie können die Bibliothek zur Steuerung der Spitzennachgiebigkeit verwenden, um die Nachgiebigkeitssteuerfunktion besser zu nutzen. Da diese Funktion jedoch die Daten der Robotersteuerung direkt ändert, kann Ihr Roboter je nach den angegebenen Werten eventuell eine abnormale Aktion ausführen. Stellen Sie zur Anpassung die zulässige Abweichung auf kleine Werte ein. Sie können für die Bibliothek der speziellen Nachgiebigkeitssteuerung die folgenden Parameter angeben.

(1) Kraftunterstützung unter Nachgiebigkeitssteuerung
Das Anwenden einer Kraftunterstützung in der Richtung der Ausrichtung ermöglicht eine Profilierungsrichtung, wenn Ihr Roboter beispielsweise aufgrund der Reibung keine Profilierung durchführen kann. Sie können mit der Bibliothek der Nachgiebigkeitssteuerung (SetFrsAssis) Unterstützungswerte in einzelnen Richtungen angeben. Die Anfangswerte nach dem Einschalten betragen Null.

(2) Einstellung der Strombegrenzung unter Nachgiebigkeitssteuerung
Unter der Bedingung der Nachgiebigkeitssteuerung wird die Kraft an der Spitze über die Bereitstellung von Motoren mit einem Drehmoment entsprechend der Positionsabweichung gesteuert. Die konventionelle Steuerung entsprechend den Winkelabweichungen für einzelne Achsen ist somit nicht aktiv (Strombegrenzung = 0), und die Roboterbewegung kann oszillieren. In diesem Fall können Sie die Einstellung der Strombegrenzung anstatt der Kraftbegrenzungsrate unter der Nachgiebigkeitssteuerung anpassen, um eine weichere Steuerung zu erzielen. Wenn Sie die Kraft an der Spitze in einer bestimmten Richtung über die Einstellung der Strombegrenzung anstatt über die Einstellung der Kraftbegrenzung begrenzen, stellen Sie die Kraftbegrenzung in dieser Richtung auf Null ein, und passen Sie die Strombegrenzung für den Drehungswinkel für die Bewegung in der begrenzten Richtung an. Sie können mit der Bibliothek der Nachgiebigkeitssteuerung für die Spitze (SetCompLimit) die Strombegrenzung für die einzelnen Achsen festlegen. Die Anfangswerte nach dem Einschalten betragen Null.

(3) Parameter zur Auswahl, wie die Nachgiebigkeit gesteuert werden soll
Die Steuerung entsprechend der Geschwindigkeitsabweichung statt der Positionsabweichung erhöht die Stabilität bei Kontaktbewegungen. Sie können eine spezielle Bibliothek zur Nachiebigkeitssteuerung an der Spitze (SetCompVMode) verwenden, um den Steuermodus für die Geschwindigkeit festzulegen; mit SetFrcLimit und SetDampRate legen Sie die Steuerungsrichtung fest.
(Beispiel) Einstellen des Geschwindigkeits-Steuermodus in ZAchsenrichtung

```
CALL SerFrcCoord(0)
CALL SetCompVMode
CALL SetFrcLimit (100, 100, 0, 100, 100, 100)
CALL SetDampRate (100, 100, 100, 100, 100, 100)
CALL SetCompControl
```

3.5.3.4 Hinweise für Ihre Sicherheit

(1) Funktionen zur Fehlererkennung
Wenn die Kraft über die Nachgiebigkeitsfunktion begrenzt wird, kann sich ein Roboter aufgrund einer externen Kraft in die Richtung der Kraftbegrenzung bewegen. Außerdem kann sich ein Roboter aufgrund der Kraftkomponente in der Begrenzungsrichtung einer angewendeten externen Kraft in eine andere Richtung als der Kraftbegrenzungsrichtung bewegen.
Die folgenden Funktionen wurden hinzugefügt, um aus Sicherheitsgründen Anomalien beim Anpassen eines Roboters zu erkennen.

1) Fehler bei Positionsabweichung unter Nachgiebigkeitssteuerung (60F8)
Dieser Fehler tritt auf, wenn die Positionsabweichung an der Werkzeugspitze des Roboters die Begrenzung überschreitet. Die Anfangswerte für die zulässige Abweichung lauten 100 (mm) für die Umsetzung entlang und 30 (Grad) um die X-, Y- und Z-Achse. Sie können mit Set-CompEralw die zulässige Abweichung ändern. Da eine hohe zulässige Abweichung eventuell das Erkennen eines Fehlers verhindert, sollten Sie diese Begrenzungen nicht zu hoch wählen.

2) Fehler bei Kraftbegrenzungsreferenz (60FB)
Dieser Fehler tritt auf, wenn eine zu große Kraft auf die Spitze des Roboterwerkzeugs einwirkt. Dieser Fehler tritt auf, wenn ein Roboterkontakt in der Richtung vorliegt, in der Sie eine hohe Kraftbegrenzungsrate angegeben haben. Wenn dieser Fehler auftritt, stellen Sie sicher, daß die Richtung der Kraftbegrenzung und die Kraftbegrenzungsparameter richtig eingestellt sind.

3) Geschwindigkeitsbegrenzungsreferenz unter Nachgiebigkeitssteuerung überschritten (60FC)
Unter der Nachgiebigkeitssteuerung ist keine Bewegung mit hoher Geschwindigkeit möglich. Dieser Fehler tritt auf, wenn das Produkt der internen und der externen Geschwindigkeit 50% überschreitet. Wenn dieser Fehler auftritt, reduzieren Sie die Einstellung der internen Geschwindigkeit.

4) Unter der Nachgiebigkeitssteuerung ist keine Punkt-zu-Punkt-Bewegung möglich (60FD)
Die Punkt-zu-Punkt-Bewegung (siehe Abschnitt 3.3.1 “Punkt-zu-Punkt-Steuerung”) ist unter der Nachgiebigkeitssteuerung nicht zulässig. Dieser Fehler tritt auf, wenn Sie eine Punkt-zu-Punkt-Bewegung ausführen. Sie sollten zur Linearbewegung wechseln oder eine Punkt-zu-Punkt-Bewegung ausführen, nachdem Sie die Nachgiebigkeitssteuerung deaktiviert haben. Außerdem tritt dieser Fehler auf, wenn Sie eine laufende Bewegung starten oder im Modus “Pfad fortsetzen” eine Bewegung erneut starten.

5) Die Aktuelle Position überschreitet J* Soft-Limit unter der Nachgiebigkeitssteuerung (66D*)
Ein Roboter kann unter der Nachgiebigkeitssteuerung aufgrund einer externen Kraft an eine Position bewegt werden, die ein Soft-Limit überschreitet. In diesem Fall tritt dieser Fehler auf. Ein Motor wird beim Überschreiten eines Soft-Limits im manuellen Modus zwar nicht ausgeschaltet; er wird jedoch ausgeschaltet, wenn dieser Fehler unter der Nachgiebigkeitssteuerung auftritt.

(2) Sicheres Anpassen
Ein Roboter kann wegen eines Fehlers in den Parametereinstellungen in Richtung der Schwerkraft fallen oder aufgrund einer externen Kraft unter der Nachgiebigkeitssteuerung eine Bewegung in einer unerwarteten Richtung ausführen.
Im Interesse einer sicheren Anpassung sollten Sie keine sehr großen Werte für die zulässige Winkelabweichung und die zulässige positive Abweichung eingeben. Verwenden Sie außerdem SetCompEralw zum Einstellen kleinerer Werte als der Anfangswerte (10 [mm] für die Umsetzung entlang und 5 [Grad]) für die Rotation um die X-, Y- und Z-Achsen) für die zulässige Abweichung in anderen Richtungen als der Richtung der Kraftbegrenzung, bevor Sie eine Anpassung durchführen.

(3) Anpassen der Parameter für die Nachgiebigkeitssteuerungsfunktionen
Passen Sie die begrenzte Kraft unter der Nachgiebigkeitssteuerung entsprechend der Kraftbegrenzungsrate an. Die Relation zwischen den Kraftbegrenzungswerten und der Kraftbegrenzungsrate variiert je nach der Haltung und Richtung eines Roboters. Passen Sie die Kraftbegrenzungsparameter für jede Kraftbegrenzungshaltung und -richtung an.
Ein Roboter führt eine Profilierung in Richtung einer externen Kraft durch, wenn ein Gelenks-Drehmoment aufgrund dieser externen Kraft größer wird als die statische Reibung eines Gelenks. Wenn andererseits die statische Reibung aufgrund einer externen Kraft größer ist als das Gelenks-Drehmoment, bleibt der Roboter im Gleichgewicht.
Ein Roboter kann sich in Richtung der Kraftbegrenzung (die Richtung des Referenzwertes) bewegen, wenn sich das Gleichgewicht zwischen der Reibung und der begrenzten Kraft als Ergebnis der Reibungsänderung aufgrund einer Änderung der Roboterhaltung im Betrieb unter der Nachgiebigkeitssteuerung ändert. Beachten Sie die Roboterbewegung in Richtung der Kraftbegrenzung, wenn Sie den Roboter betreiben, während Sie die Nachgiebigkeit steuern.

(4) Positionsprogrammierung
Sie können die Funktion zum Einstellen von Positionsvariablen verwenden, um die Position eines Roboters zu programmieren. Die Positionseinstellung in Variablen ist die aktuelle Positionsreferenz eines Roboters. Wenn Sie unter der Nachgiebigkeitssteuerung einen Roboter schieben, um ihn zu bewegen und seine Position festzulegen (direkte Programmierung), gibt es Unterschiede zwischen der Positionsreferenz und der tatsächlichen Position. Gehen Sie anhand der folgenden Prozedur vor, um unter der Nachgiebigkeitssteuerung Positionen zu programmieren.

1) Legen Sie eine Variable des Typs I als Programmierungsnummer fest. In diesem Beispiel verwenden wir I10.
2) Bereiten Sie das folgende Programm vor.
Wenn Sie eine Variable des Typs P für die Programmierung verwenden:

```
Program DirectTeach
    P(I10)=CURPOS
end
```

Wenn Sie eine Variable des Typs J für die Programmierung verwenden:

```
Program DirectTeach
    J(I10)=CURJNT
end
```

3) Ordnen Sie I10 eine Programmierungspositionsnummer zu, bewegen Sie den Roboter an den Programmierungspunkt und führen Sie DirectTeach aus.

(5) Einstellen von Werkzeugkoordinate und Arbeitskoordinate
Wenn Sie mit SetFrcCoord die Kraftbegrenzungskoordinate auf die Werkzeugkoordinate festlegen, die Nachgiebigkeitssteuerung ausführen und mit Changetool die Werkzeugkoordinate unter der Nachgiebigkeitssteuerung ändern, wird die Richtung der Kraftbegrenzung nicht geändert. Die Werkzeugkoordinate ändert sich jedoch hinsichtlich der Roboterbewegung. Dies führt zu einer Diskrepanz zwischen dem Koordinatensystem der Kraftbegrenzung und dem Koordinatensystem der Bewegung. Dies gilt für die Arbeitskoordinate ebenso wie für die Werkzeugkoordinate.
Wenn Sie große Werte für X, Y und Z der Werkzeugkoordinate zur Ausführung der Nachgiebigkeitssteuerung angeben, ist die Empfindlichkeit hinsichtlich der Änderungen der Roboterhaltung hoch, und ein Roboter kann eine Oszillation aufweisen, die zu einem fehlerbedingten Stop führt. Verwenden Sie in diesem Fall SetFrcLimit und SetCompRate, um die Parameter allmählich von 100 zu verringern.

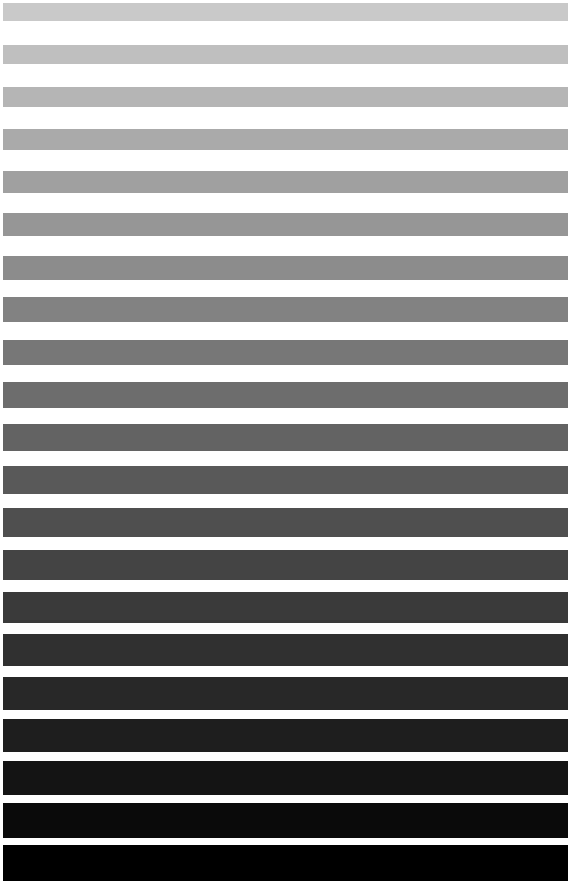
(6) Fortsetzungsfunktion
Unter der Nachgiebigkeitssteuerung ist keine fortlaufende Bewegung möglich. Wenn Sie einen Motor ausschalten und in einen Status bringen, der ihn für eine fortlaufende Bewegung vorbereitet, wird ein Fehler “66EF Nachgiebigkeitssteuerung ist deaktiviert” angezeigt. Wenn Sie nach einem vorübergehenden Stop die Fortsetzung starten, wird ein Fehler “66FD Punkt-zu-Punkt-Bewegung ist unter der Nachgiebigkeitssteuerung nicht verfügbar” angezeigt.

(7) Wiederherstellung nach Stromausfall
Die Funktion für die Wiederherstellung nach einem Stromausfall ist nicht verfügbar, wenn die Stromversorgung des Controllers unter der Nachgiebigkeitssteuerung unterbrochen wird.

(8) Genauigkeit der Roboterbewegung unter der Nachgiebigkeitssteuerung
Wenn ein Bewegungsbefehl unter der Nachgiebigkeitssteuerung ausgeführt wird, sinkt die Genauigkeit der Bewegungskurve eines Roboters. Die Position kann in Richtung der Kraftbegrenzung eine große Abweichung ergeben. Wenn Sie eine exakte Bewegung benötigen, deaktivieren Sie die Nachgiebigkeitssteuerung.
Die Stopgenauigkeit des Roboters sinkt unter der Nachgiebigkeitssteuerung. Wenn Sie eine Codiererwert-Prüfbewegung (siehe Abschnitt 4.2.3 “Codiererwert-Prüfbewegung”) ausführen, kann ein Roboter für einen längeren Zeitraum stoppen, oder es kann ein Fehler “6651 Prüfbefehl Zeit überschritten” angezeigt werden. Wenn Sie beim Stoppen eine hohe Genauigkeit benötigen, deaktivieren Sie die Nachgiebigkeitssteuerung.

Kapitel 4

Bestimmen von Geschwindigkeit, Beschleunigung und Verzögerung



Die maximalen Raten für Geschwindigkeit, Beschleunigung und Verzögerung müssen festgelegt werden.
Dieses Kapitel erläutert die Bedeutung und die Einstellungen von Geschwindigkeit, Beschleunigung und Verzögerung.

4.1 Externe Geschwindigkeit und interne Geschwindigkeit

Für die Roboter der VS-Serie wird zwischen interner und externer Geschwindigkeit unterschieden.
Die externe Geschwindigkeit ist die Geschwindigkeit, die vor der Ausführung eines Programmes über das Programmiergerät oder ein externes Gerät eingestellt wurde.
Die interne Geschwindigkeit ist die Geschwindigkeit, die mit einem Befehl im Programm eingestellt wird.

4.2 Angabe der Geschwindigkeit

Das Produkt aus externer Geschwindigkeit und interner Geschwindigkeit bestimmt die tatsächliche Geschwindigkeit der Roboterbewegung.
Es können beispielsweise die folgenden Bedingungen gesetzt sein:

Externe Geschwindigkeit: 70%
Interne Geschwindigkeit: 30%

Dies ergibt den folgenden Ausdruck.

Tatsächliche Geschwindigkeit = Maximale Geschwindigkeit $\times$ 0,7 $\times$ 0,3

Die tatsächliche Geschwindigkeit beträgt somit 21% der maximalen Geschwindigkeit.

Wenn Sie die interne Geschwindigkeit mit dem Befehl SPEED in einem Programm ändern, werden auch die interne Beschleunigung und die interne Verzögerung eingestellt. Die interne Beschleunigung und die interne Verzögerung entsprechen dem Quadrat der internen Geschwindigkeit geteilt durch 100.
Der Befehl JSPEED ändert die interne Achsengeschwindigkeit. Wenn Sie die interne Achsengeschwindigkeit ändern, werden auch die interne Achsenbeschleunigung und die interne Achsenverzögerung geändert. Die interne Achsenbeschleunigung und die interne Achsenverzögerung entsprechen dem Quadrat der internen Achsengeschwindigkeit geteilt durch 100.

4.3 Externe Beschleunigung, Externe Verzögerung, Interne Beschleunigung und Interne Verzögerung

Die Beschleunigung und Verzögerung werden nach externer Beschleunigung/Verzögerung und interner Beschleunigung/Verzögerung unterschieden.
Die externe Beschleunigung und die externe Verzögerung werden vor der Ausführung eines Programmes über das Programmiergerät oder ein externes Gerät eingestellt.
Die interne Beschleunigung und die interne Verzögerung werden mit einem Befehl in einem Programm eingestellt.

4.4 Festlegen der Beschleunigung und Verzögerung

Das Produkt der externen Beschleunigung und der internen Beschleunigung bestimmt die tatsächliche Beschleunigung und Verzögerung.
Es können beispielsweise die folgenden Bedingungen gesetzt sein:

Externe Verzögerung: 70%
Interne Verzögerung: 30%

Dies ergibt den folgenden Ausdruck.

Tatsächliche Verzögerung = Maximale Verzögerung $\times$ 0.7 $\times$ 0.3

Die tatsächliche Verzögerung beträgt somit 21% der maximalen Verzögerung.
Die Befehle zum Einstellen der Beschleunigung und Verzögerung sind in Tabelle 4-1 aufgelistet.

Tabelle 4-1 Befehle zum Einstellen von Beschleunigung und Verzögerung

Befehl	Funktion
SPEED	Einstellen der internen Geschwindigkeit
ACCEL	Einstellen der internen Beschleunigung
DECEL	Einstellen der internen Verzögerung
JSPEED	Einstellen der internen Achsengeschwindigkeit
JACCEL	Einstellen der internen Achsenbeschleunigung
JDECEL	Einstellen der internen Achsenverzögerung

4.5 Beispiel zum Einstellen der Geschwindigkeit und Beschleunigung

Das in Abb. 4-1 gezeigte Programm stellt die interne Geschwindigkeit nicht ein. Wenn Sie diese Art von Programm bei 80% externer Geschwindigkeit ausführen, ergeben sich die folgenden Ergebnisse.

```
PROGRAM PRO1
TAKEARM
   MOVE P, P1
END

Tatsächliche Geschwindigkeit = externe Geschwindigkeit x interne
Geschwindigkeit
                              = 80% x 100%
                              = 80%

Tatsächliche Beschleunigung = externe Beschleunigung x interne
Beschleunigung
                             = 64% x 100%
                             = 64%

Tatsächliche Verzögerung = externe Verzögerung x interne Verzögerung
                         = 64% x 100%
                         = 64%
```

Abb. 4-1 Beispiel ohne Einstellen der internen Geschwindigkeit

Die interne Geschwindigkeit wird wie in Abb. 4-2 gezeigten Programm eingestellt.
Wenn Sie ein ähnliches Programm bei 80% externer Geschwindigkeit ausführen, ergeben sich die folgenden Ergebnisse.

```
PROGRAM PRO2
TAKEARM
   SPEED 50            'Interne Geschwindigkeit 50%
   MOVE P, P1
END

Tatsächliche Geschwindigkeit = externe Geschwindigkeit x interne
Geschwindigkeit
                              = 80% x 50%
                              = 40%

Tatsächliche Beschleunigung = externe Beschleunigung x interne
Beschleunigung
                             = 64% x 25%
                             = 16%

Tatsächliche Verzögerung = externe Verzögerung x interne Verzögerung
                           = 64% x 25%
                           = 16%
```

Abb. 4-2 Einstellungsbeispiel der internen Geschwindigkeit

Das in Abb. 4-3 gezeigte Programm stellt die interne Beschleunigung und Verzögerung ein.
Wenn Sie dieses Programm bei 80% externer Geschwindigkeit ausführen, ergeben sich die folgenden Ergebnisse.

```
PROGRAM PRO1
TAKEARM
   ACCEL 50            ' Interne Beschleunigung:  50%
   DECEL 25            ' Interne Verzögerung:  25%
   MOVE P, P1
END

Tatsächliche Geschwindigkeit = interne Geschwindigkeit x interne
Geschwindigkeit
                              = 80% x 100%
                              = 80%

Tatsächliche Beschleunigung = externe Beschleunigung x interne
Beschleunigung
                             = 64% x 50%
                             = 32%

Tatsächliche Verzögerung = externe Verzögerung x interne Verzögerung
                           = 64% x 25%
                           = 16%
```

Abb. 4-3 Einstellungsbeispiel der internen Beschleunigung und Verzögerung

Achtung:	**Wenn Sie die Beschleunigung und Verzögerung festlegen, stellen Sie sicher, daß nicht die Gefahr einer Kollision durch die Änderung des Duchlaufs besteht.** **Wenn Sie die Geschwindigkeit oder Beschleunigung mit einer Bewegungsoption oder während einer Durchlaufbewegung mit einem Befehl ändern, beachten Sie die Möglichkeit einer Kollision.**

4.6 Steuersätze der Bewegungsoptimierung

Diese Funktion wird zum Einstellen der richtigen Geschwindigkeit und Beschleunigung entsprechend der Masse der Nutzlast und der Haltung des Roboters verwendet. Sie können einen Steuersatz der Bewegungsoptimierung aus 4 in der Tabelle 4-1 aufgelisteten Sätzen auswählen.

Tabelle 4-1 Steuersätze der Bewegungsoptimierung

Steuersatz	Einstellungsbedingung	Beschreibung	
		Punkt-zu-Punkt-Bewegung	Linearbewegung
0	Masse der Nutzlast	Maximale Beschleunigung	Maximale Beschleunigung
1	Masse der Nutzlast und Roboterhaltung	Maximale Geschwindigkeit, Beschleunigung	Wie Steuersatz 0
2		Wie Steuersatz 0	Maximale Geschwindigkeit, Beschleunigung
3		Wie Steuersatz 1	Wie Steuersatz 2

4.6.1 Steuersatz 0

Dieser Steuersatz entspricht dem Standardwert beim Starten des Controllers. Stellen Sie die maximale Beschleunigung der Punkt-zu-Punkt-Bewegung und der Linearbewegung entsprechend dem Lastbedingungswert des Roboters ein. Für die Roboter der VS-D-Serie ist die Relation zwischen der Lastbedingung und der Mindestbewegungszeit für Steuersatz 0 in Abb. 4-4 bis 4-9 dargestellt. Verwenden Sie bei der Konzeption diese Angaben zur Berechnung der Zykluszeit.
Schlagen Sie für andere Roboter im “Installations- und Wartungshandbuch” nach.

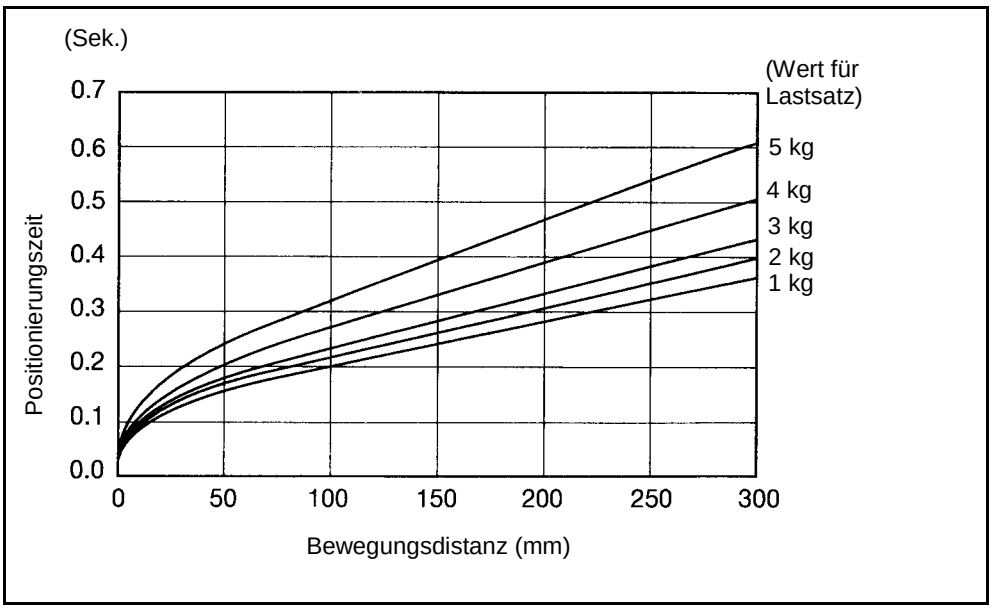


Abb. 4-4 Linearbewegung Positionierungszeit (VS-D)

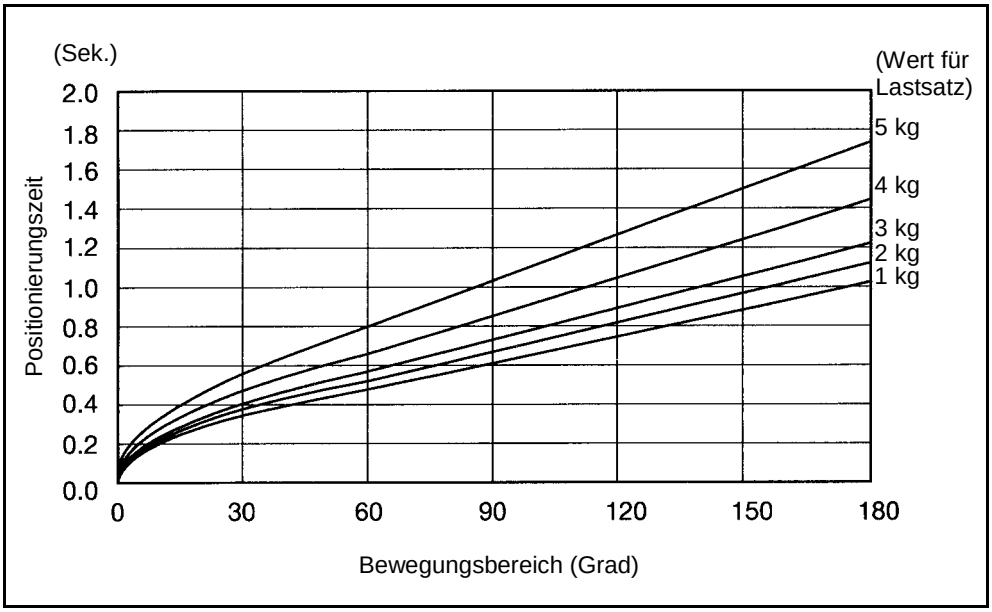


Abb. 4-5 Positionierungszeit 1. und 2. Achse bei Punkt-zu-Punkt-Bewegung (VS-D)

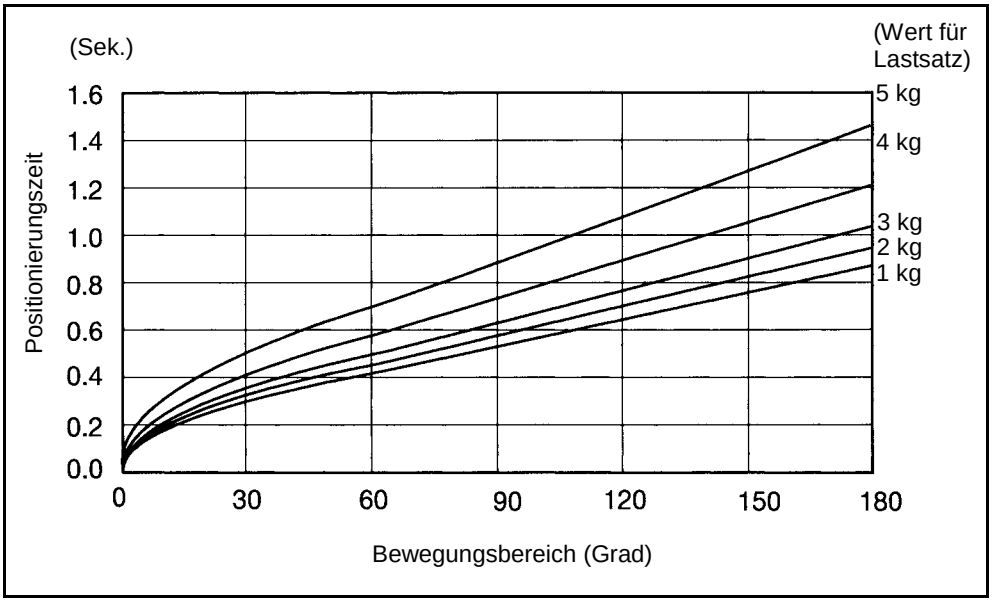


Abb. 4-6 Positionierungszeit 3., 4. und 5. Achse bei Punkt-zu-Punkt-Bewegung (VS-D)

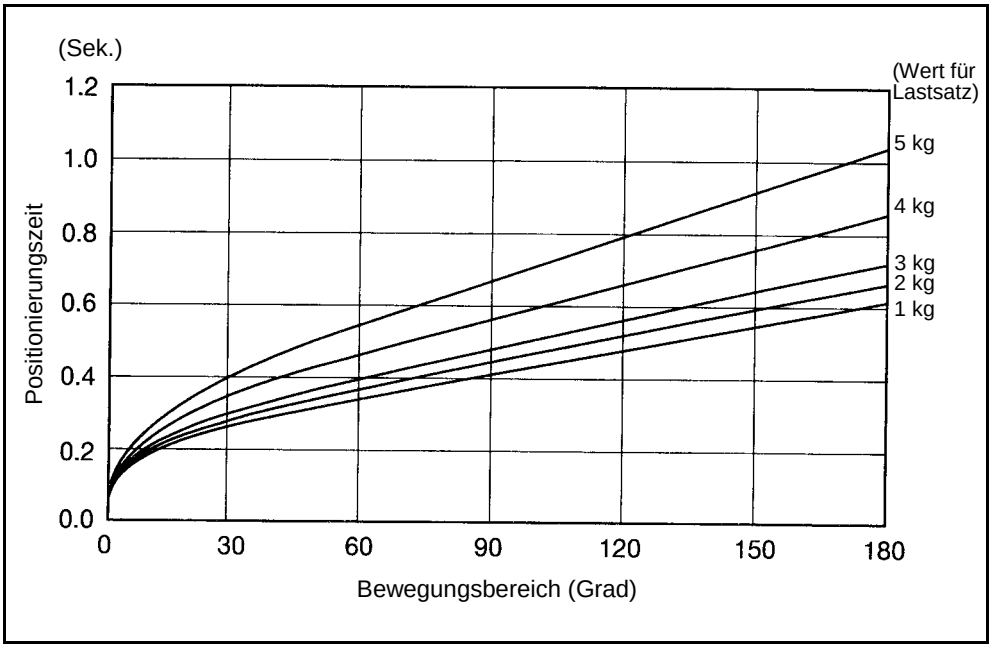


Abb. 4-7 Positionierungszeit 6. Achse bei Punkt-zu-Punkt-Bewegung (VS-D)

4.6.2 Steuersatz 1

Stellen Sie die maximale Geschwindigkeit und Beschleunigung für die 1., 2. und 3. Achse bei der Punkt-zu-Punkt-Bewegung entsprechend dem Wert für die Lastbedingung des Roboters und der bewegten Roboterarmstellung ein. Für die 4., 5. und 6. Achse bei der Punkt-zu-Punkt-Bewegung sowie für die Linearbewegung entspricht dies den Angaben von Steuersatz 0.

4.6.2.1 Verwenden von Steuersatz 1

Wenn Sie die Bewegungszeit bei der Punkt-zu-Punkt-Bewegung verringern müssen, wählen Sie Steuersatz 1.
Überprüfen Sie die Bewegungszeit für Steuersatz 1 im Betrieb mit Maschinenverriegelung.

4.6.2.2 Vorsichtsmaßnahmen bei der Verwendung von Steuersatz 1

Bei der Bewegung kann ein Überlastungsfehler oder ein Fehler durch eine zu große Abweichung auftreten. Überprüfen Sie für den Lastfaktor den geschätzten Überlastungswert am Programmiergerät. (siehe Abschnitt 5.3 “Anzeigen der erwarteten Überlastungen für die Kapazität von Motoren und für den Bremswiederstand des Roboter-Controller, [F2]—[F6]—[F10]” in Kapitel 5 des EINSTELLUNGSHANDBUCHES) Oder überprüfen Sie den Lastfaktor mit der Protokollfunktion von WINCAPSII (siehe Abschnitt 10.4 “Aktionsmenü” in Kapitel 10 des WINCAPSII-HANDBUCHES).

Wenn ein Überlastungsfehler auftritt, passen Sie die Motorlast an, indem Sie die entsprechenden Werte für Timer, interne Geschwindigkeit und Beschleunigung einstellen.
Wenn eine zu starke Abweichung auftritt, passen Sie die Geschwindigkeit und die Beschleunigung an.

Abhängig von der Bewegungsgeschwindigkeit kann sich der Durchgangsort um ca. 20 mm ändern. Da die Durchlaufbewegung in der Nähe eines Hindernisses eventuell zu einer Kollision mit dem Hindernis führen kann, sollten Sie die Bewegung mit Steuersatz 0 durchführen.

4.6.3 Steuersatz 2

Stellen Sie die maximale Geschwindigkeit und Beschleunigung entsprechend dem Wert für die Lastbedingung des Roboters und der bewegten Roboterarmstellung ein. Dies entspricht der Einstellung von Steuersatz 0 bei der Punkt-zu-Punkt-Bewegung.

4.6.3.1 Verwenden von Steuersatz 2

Verwenden Sie Steuersatz 2 in den beiden folgenden Fällen.

- Wenn Sie die Bewegungszeit bei der Punkt-zu-Punkt-Bewegung verringern müssen.
  Die Relation zwischen der Lastbedingung und der kürzesten Bewegungszeit in Steuersatz 2 wird in Abb. 4-8 dargestellt. Da jedoch der Roboter während der Bewegung die Geschwindigkeit eventuell automatisch reduziert, überprüfen Sie die Bewegungszeit im Betrieb mit Maschinenverriegelung.

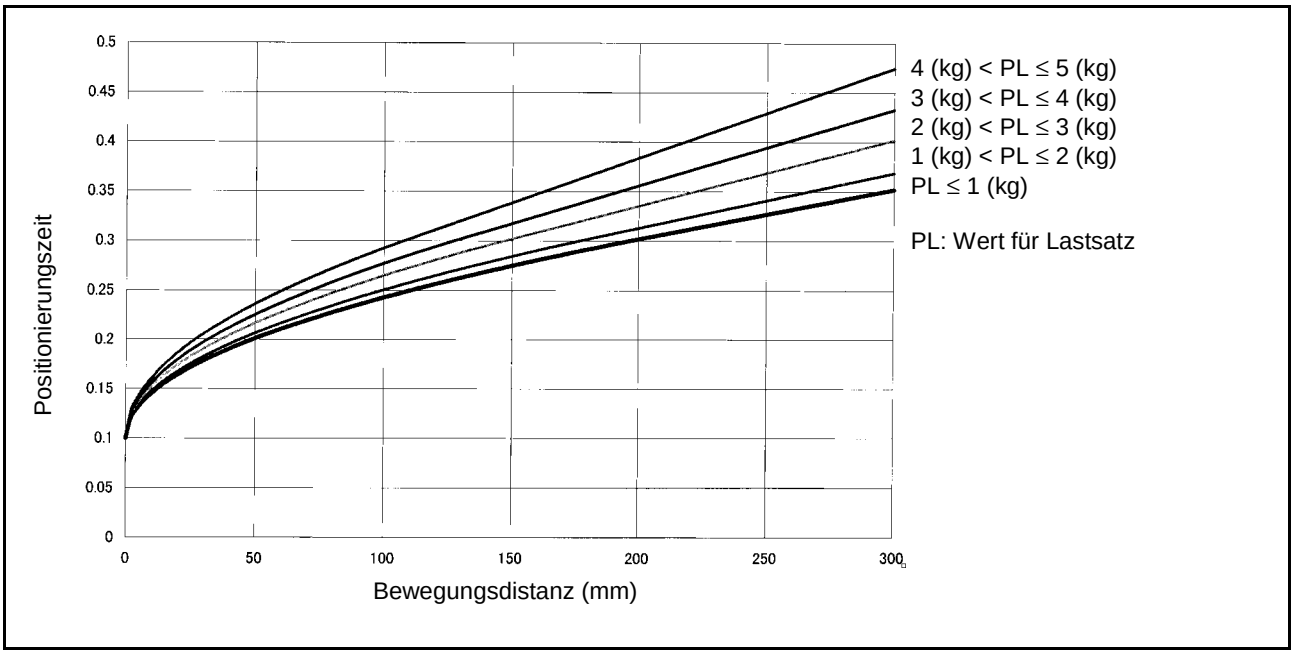


Abb. 5-8 Positionierungszeit bei Linearbewegung (Kürzeste Zeit); (VS-D)

- Wenn Sie Steuersatz 0 oder 1 wählen, tritt bei der Linearbewegung ein Fehler “Befehlsgeschwindigkeit überschritten” (6081 bis 6086) auf. Wenn bei Steuersatz 0 oder 1 der Pfad dicht an einem singulären Punkt (siehe Abschnitt 4.1.3. SETTING-UP, Teilabschnitt 4.1.3, “[2] Grenzen der Roboter-Armstellungen” in Kapitel 4 des EINSTELLUNGSHANDBUCHES) oder in der Nähe der Begrenzung des Bewegungsbereiches vorbeiführt, kann ein Fehler “Befehlsgeschwindigkeit überschritten” auftreten, und der Roboter wird gestoppt. Bei Steuersatz 2 liegt die Geschwindigkeit dagegen automatisch innerhalb der Begrenzung der Befehlsgeschwindigkeit, so daß Sie den Roboter ohne diesen Fehler betreiben können.

4.6.3.2 Vorsichtsmaßnahmen bei der Verwendung von Steuersatz 2

- Bei diesem Steuersatz kann während der Roboterbewegung ein Überlastungsfehler auftreten. Wenn Sie die Geschwindigkeit anpassen, überprüfen Sie die Lastrate mit der Protokollfunktion des geschätzten Lastwertes am Programmiergerät. (Siehe Abschnitt 5.3. “Anzeigen der erwarteten Überlastungen für die Kapazität von Motoren und für den Bremswiederstand des Roboter-Controller, [F2]—[F6]—[F10]” in Kapitel 5 des EINSTELLUNGSHANBUCHES.) Oder prüfen Sie die Lastrate mit der Protokollfunktion von WINCAPSII (siehe Abschnitt 10.1 “Überblick über den Log Manager” in Kapitel 10 des WINCAPSII-HANDBUCHES). Wenn ein Überlastungsfehler auftritt, passen Sie die Motorlast an, indem Sie die entsprechenden Werte für Timer oder die interne Geschwindigkeit und Beschleunigung einstellen.

- Abhängig von der Bewegungsgeschwindigkeit kann sich der Pfad um ca. 20 mm ändern. Da die Durchlaufbewegung in der Nähe eines Hindernisses eventuell zu einer Kollision führen kann, verwenden Sie Steuersatz 0.

- Da sich die Geschwindigkeit bei der Linearbewegung in einem Abschnitt mit konstanter Geschwindigkeit ändern kann, führen Sie Arbeiten, die eine konstante Bewegungsgeschwindigkeit erfordern, mit dem Steuersatz 0 oder 1 aus.

- Fehler zur Überschreitung der Begrenzung der Befehlsbeschleunigung (6761 bis 6766) und starke Abweichungen (6111 bis 6116) können bei der Linearbewegung auftreten. Wenn ein solcher Fehler auftritt, passen Sie die Beschleunigung mit der internen Geschwindigkeit und der internen Beschleunigung an. Bei einer Hochgeschwindigkeitsbewegung kann außerdem eine Pfadverschiebung um bis zu 5 mm auftreten. Wenn in der Nähe des Bewegungspfads Hindernisse sind, sollten Sie daher die Geschwindigkeit des Roboters verringern.

- Wenn Sie den Roboter in der Nähe eines singulären Punkts sofort stoppen (siehe Abschnitt 4.1.3 “SETTING-UP MANUAL” in Kapitel 4 des EINSTELLUNGSHANDBUCHES), kann sich die Zeit für den sofortigen Stop verlängern. Die Distanz für den sofortigen Stop bleibt jedoch unverändert.

4.6.4 Steuersatz 3

Bei diesem Steuersatz bewegt sich der Roboter genau wie mit Steuersatz 1 bei der Punkt-zu-Punkt-Bewegung und mit Steuersatz 2 bei der Linearbewegung. Siehe hierzu Abschnitt 4.6.2 “Steuersatz 1” und 4.6.3 “Steuersatz 2” weiter oben.

4.6.5 Verwenden von Bedingungen, die Sie setzen müssen

Wenn Sie die Masse der Nutzlast ermitteln (Endwerkzeug und Werkstück), die auf die 6. Achse des Roboters angewendet wird, stellen Sie die Masse der Nutzlast und die Position des Schwerpunktes ein. Stellen Sie die Masse der Nutzlast und die Position des Schwerpunktes in der Verwendungsbedingung des Programmiergeräts oder in WINCAPSII ein. Sie können diese Angaben auch mit einem Programm ändern, um das Werkzeug oder Werkstück zu ändern. Zum Ändern der Angaben mit einem Programm, siehe 4.7.2 “Einstellen der internen Lastbedingungswerte (Masse der Nutzlast und Schwerpunkt) und Interner Modus”.

4.6.6 Hinweise zum Einstellen

Es gibt externe Lastbedingungswerte (Masse der Nutzlast, Position des Schwerpunktes) und einen externen Modus, die mit dem Programmiergerät und WINCAPSII eingestellt werden, interne Lastbedingungswerte und einen internen Modus, die mit einem Programm eingestellt werden, sowie Lastbedingungswerte (Masse der Nutzlast, Position des Schwerpunktes) und Modus, die mit dieser Funktion eingestellt werden.

Wenn Sie die externen Lastbedingungswerte und den externen Modus einstellen, haben die internen Lastbedingungswerte und der interne Modus die gleichen Werte.

Der Roboter wird durch Einstellen der maximalen Geschwindigkeit und Beschleunigung aus der internen Lastbedingung und dem internen Modus bewegt.

Sie können die aktuellen Lastbedingungswerte und den internen Modus am Programmiergerät einstellen (siehe Abschnitt 4.7 “Einstellen der Master-Steuerparameter in den Benutzereinstellungen”).

Wenn Sie die Hauptstromversorgung des Controllers einschalten, werden der interne und der externe Modus auf “0” gesetzt. (Anmerkung: nur bis Ver. 1.3*. Siehe Abschnitt 4.7.4.)

Die internen Lastbedingungswerte werden mit den externen Lastbedingungswerten gleichgesetzt. Stellen Sie daher den internen Modus und die internen Lastbedingunswerte in einem Programm entsprechend Ihren Anforderungen ein.

Die externen Lastbedingungswerte vor der Auslieferung lauten 7000 [g] für die Masse der Endlast und X6=0 [mm], Y6=80 [mm] und Z6=100 [mm] für die Position des Lastschwerpunktes.

Anmerkung (1):	**Stellen Sie sicher, daß die richtigen Lastbedingungswerte für die Last eingestellt werden. Falsche Einstellungen können beim Betrieb des Roboters zu Überspannungen, starken Abweichungen, Überlastungen und anderen Problemen und damit zu einem Roboterfehler führen. Darüber hinaus kann, wenn der Roboter nach dem Empfangen eines Roboter-Stopsignals stoppt, die Stopdistanz verlängert werden, und der Roboter kann mit Peripheriegeräten kollidieren.**
Anmerkung (2):	**Stellen Sie die richtige Bedingung für die Roboterinstallation ein. Verwenden Sie hierzu das Programmiergerät oder WINCAPSII (siehe 4.7.3 “Einstellen der Roboterinstallationsbedingungen”).** **Falsche Einstellungen können beim Betrieb des Roboters zu Überspannungen, starken Abweichungen, Überlastungen und anderen Problemen und damit zu einem Roboterfehler führen. Darüber hinaus kann, wenn der Roboter nach dem Empfangen eines Roboter-Stopsignals stoppt, die Stopdistanz verlängert werden, und der Roboter kann mit Peripheriegeräten kollidieren.**

4.7 Einstellen der Master-Steuerparameter in den Benutzereinstellungen

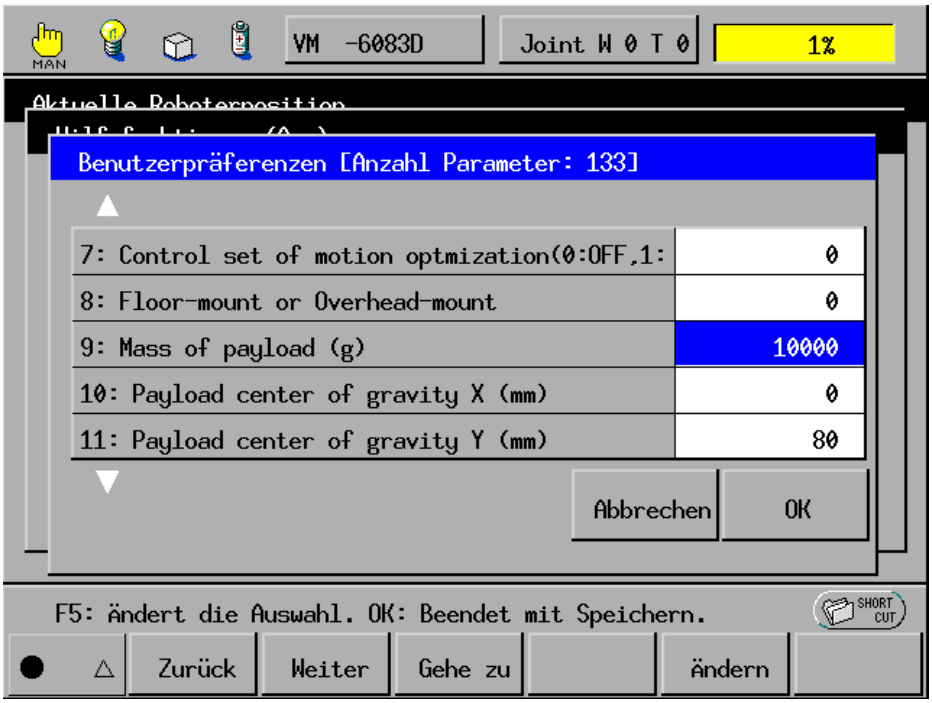
4.7.1 Einstellen der Master-Steuerparameter für die Masse der Nutzlast, den Schwerpunkt und den Steuersatz der Bewegungsoptimierung

■ Einstellen mit dem Programmiergerät

Zugriff: Hauptanzeige

└ [F2 Arm] ┐
└ [F6 Hilfs.] ┐
└ [F7 Konfig.]

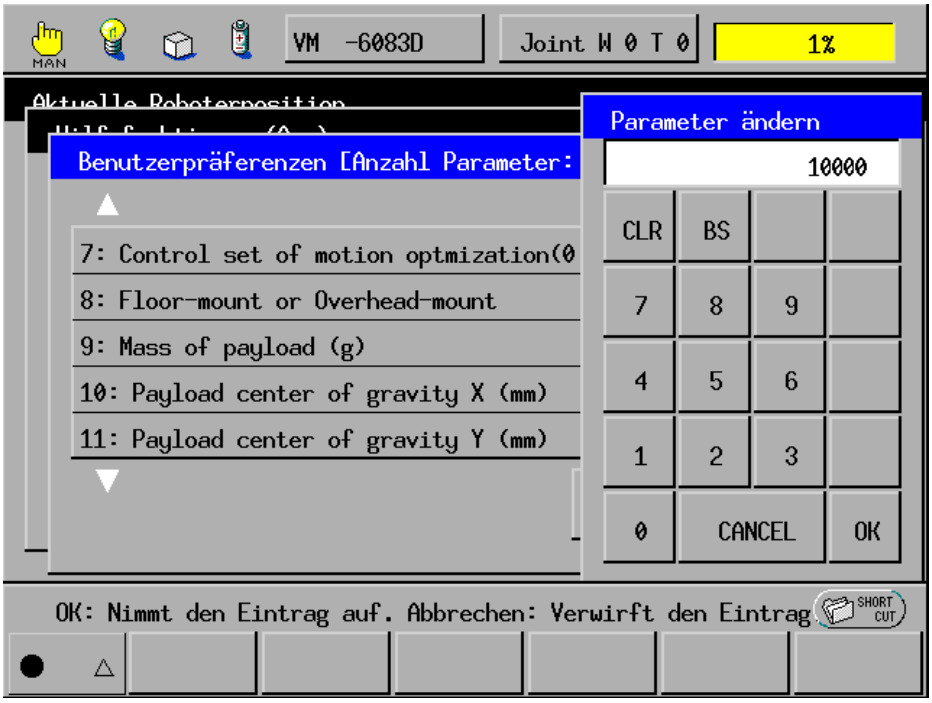
Wenn Sie das Programmiergerät verwenden und anhand der folgenden Prozedur vorgehen, wird das Fenster Benutzereinstellungen angezeigt, in dem Sie die Master-Steuerparameter wie beispielsweise den Steuersatz der Bewegungsoptimierung und die Masse der Nutzlast einstellen können. Siehe Abschnitt Abschnitt 2.9 in Kapitel 2 des EINSTELLUNGSHANDBUCHES.



Wählen Sie die folgenden Elemente im Fenster der Benutzereinstellungen aus und drücken Sie [F5 Ändern], um das numerische Tastenfeld aufzurufen, in dem Sie neue Werte eingeben können.

Einstellen von Element:
“Steuersatz der Bewegungsoptimierung”
“Masse der Nutzlast (g)”
“Nutzlastschwerpunkt X (mm)”
“Nutzlastschwerpunkt Y (mm)”
“Nutzlastschwerpunkt Z (mm)”

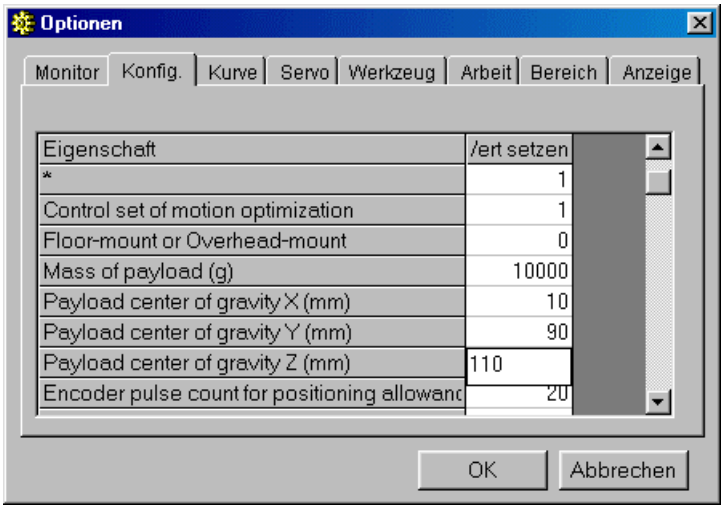
Der Eingabebereich für “Steuersatz der Bewegungsoptimierung” reicht von 0 bis 3. Wenn Sie einen Wert außerhalb dieses Bereiches eingeben, kann der folgende Fehler auftreten: “Fehler 6003 Gültiger numerischer Bereich überschritten.”
Der Eingabebereich für “Masse der Nutzlast” ist in jedem Robotermodell angegeben. Wenn Sie einen Wert außerhalb dieses Bereiches eingeben, tritt der folgende Fehler auf: “Fehler 60d2 Wert für Endlast überschreitet den zulässigen Wert”.
Geben Sie für “Nutzlastschwerpunkt” einen Wert entsprechend dem folgenden Bereich ein. Wenn der Wert außerhalb des folgenden Bereiches liegt, tritt der “Fehler 60d2 Wert für Endlast überschreitet den zulässigen Wert” auf.



■ Einstellen mit WINCAPSII

In diesem Abschnitt werden die Methoden erläutert, mit denen die externen Lastbedingungswerte (Masse der Nutzlast und Schwerpunkt) und der externe Modus mit dem PC-Programmiergerät WINCAPSII eingestellt werden (siehe Abschnitt 2.9 in Kapitel 2 des EINSTELLUNGSHANDBUCHES und in Abschnitt 8.6.1.2 “Benutzereinstellungen” in Kapitel 8 des WINCAPSII HANDBUCHES).

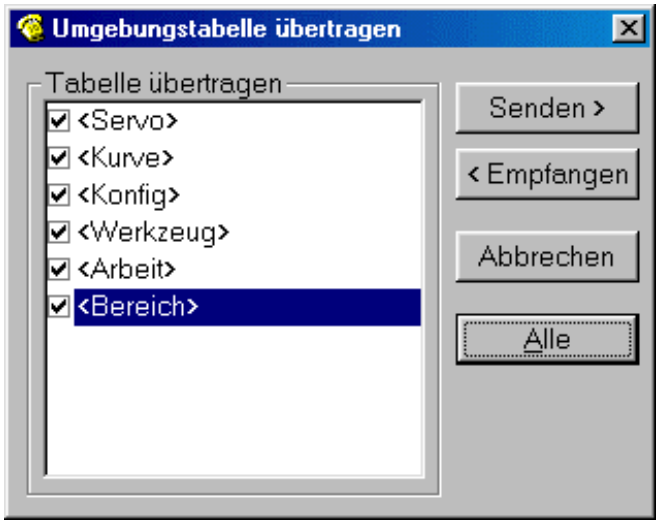
Wählen Sie “Optionen” aus dem Werkzeugmenü des Arm Manager aus, um das Optionsfenster anzuzeigen.
Klicken Sie das Register “Benutzereinstellungen” im Optionsfenster an, um das Fenster “Benutzereinstellungen” anzuzeigen.



Wenn Sie die Felder zum Einstellen von Werten für die folgenden Elemente in dieser Anzeige [Benutzereinstellungen (Anzahl Parameter:)] doppelt anklicken, können Sie die entsprechenden Parameterwerte ändern.

Einstellen von Element:
“Steuersatz der Bewegungsoptimierung”
“Masse der Nutzlast (g)”
“Nutzlastschwerpunkt X (mm)”
“Nutzlastschwerpunkt Y (mm)”
“Nutzlastschwerpunkt Z (mm)”

Nachdem Sie alle Parameterwerte eingestellt haben, übertragen Sie die Daten an die Robotersteuerung.
Schalten Sie zunächst die Stromversorgung des Motors über die Taste MOTOR am Programmiergerät aus. Stellen Sie den Arm Manager auf den Verbindungsstatus ein und klicken Sie **Senden** im Arm Manager an, um die Umgebungstabelle für die Übertragung anzuzeigen. Führen Sie anschließend die Übertragung aus.
Weitere Informationen zur Übertragung, siehe Abschnitt 8.2.5 “Projekt übertragen” in Kapitel 8 des WINCAPSII Handbuches.



Anmerkung (1):	**Klicken Sie < “Empfangen” in der Umgebungstabelle für die Übertragung an, um die Daten aus dem Arm Manager des Controllers zu empfangen. Der externe Lastbedingungswert und der externe Modus kann in diesem Fall empfangen werden. Die in einem Programm geänderten internen Lastbedingungswerte und die internen Modi können jedoch nicht an WINCAPSII gesendet werden.**
Anmerkung (2):	**Geben Sie den Nutzlastschwerpunkt in die Werkzeug-0-Koordinaten ein (siehe Abb. 4-9). Die Einheit lautet mm. Die Referenzposition der Werkzeug-0-Koordinaten ist das Flanschzentrum der 6. Achse. Für die Komponente Y ist es die Richtung vom Flanschzentrum zu einer Bohrung von 6H7 (Richtung des Orientierungsvektors). Für die Komponente Z ist es die Richtung vertikal zur Flanschfläche durch den Flanschmittelpunkt (Richtung des Anfahrvektors). Für die Komponente X ist es die Richtung der X-Achse (Richtung des Normalvektors) im rechten Koordinatensystem, wenn der Orientierungsvektor auf die Y-Achse gesetzt ist und der Anfahrvektor auf die Z-Achse (siehe Abb. 4-10).**

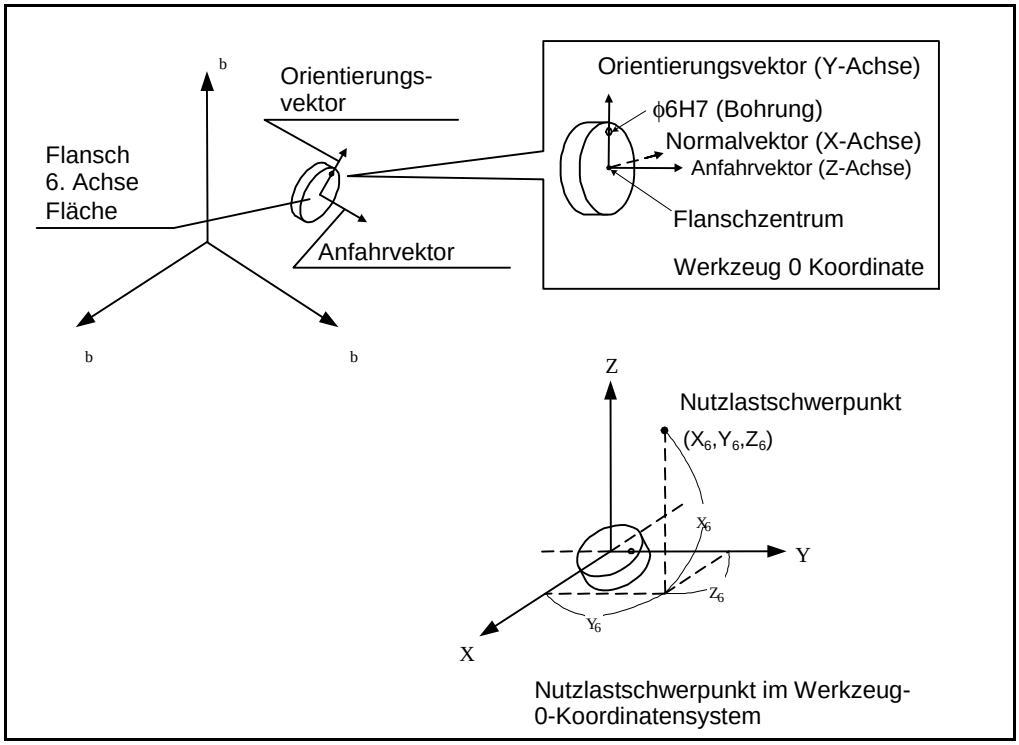


Abb. 4-9 Nutzlastschwerpunkt

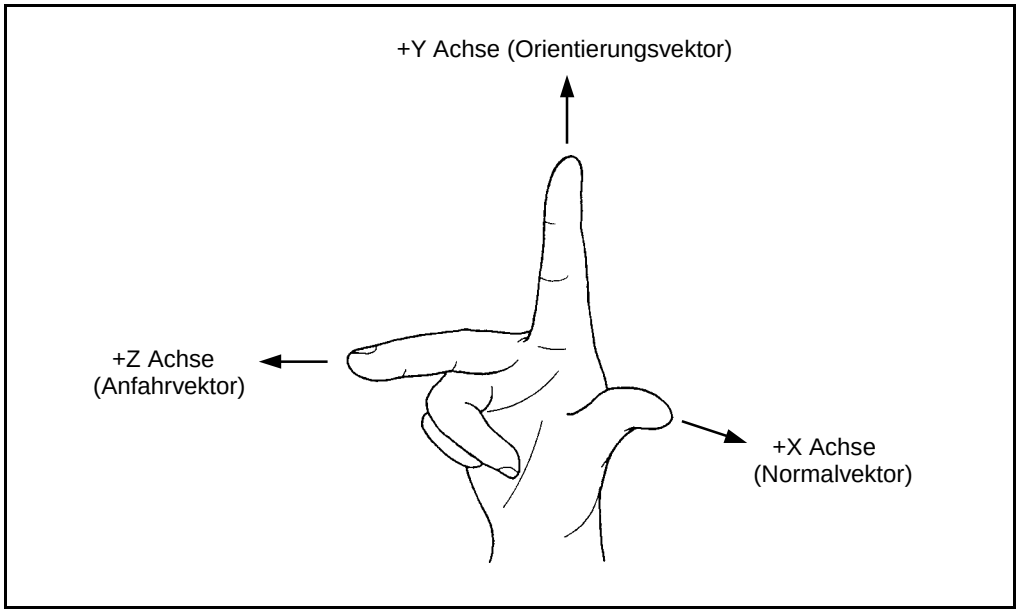


Abb. 4-10 Rechtes Koordinatensystem

4.7.2 Einstellen der internen Lastbedingungswerte (Masse de Nutzlast und Schwerpunkt) und Interner Modus

4.7.2.1 Einstellen der internen Lastbedingungswerte

Stellen Sie diese Werte ein, indem Sie die Bibliothek für die konventionelle Sprache aspACLD ausführen. Weitere Informationen, siehe Abschnitt 22.1 “Konventionelle Sprache” in Kapitel 22 dieses Handbuches.

4.7.2.2 Einstellen des internen Modus

Stellen Sie diesen Modus ein, indem Sie die Bibliothek für die konventionelle Sprache aspChange ausführen. Weitere Informationen, siehe Abschnitt 22.1.2 “aspChange” in Kapitel 22 dieses Handbuches.

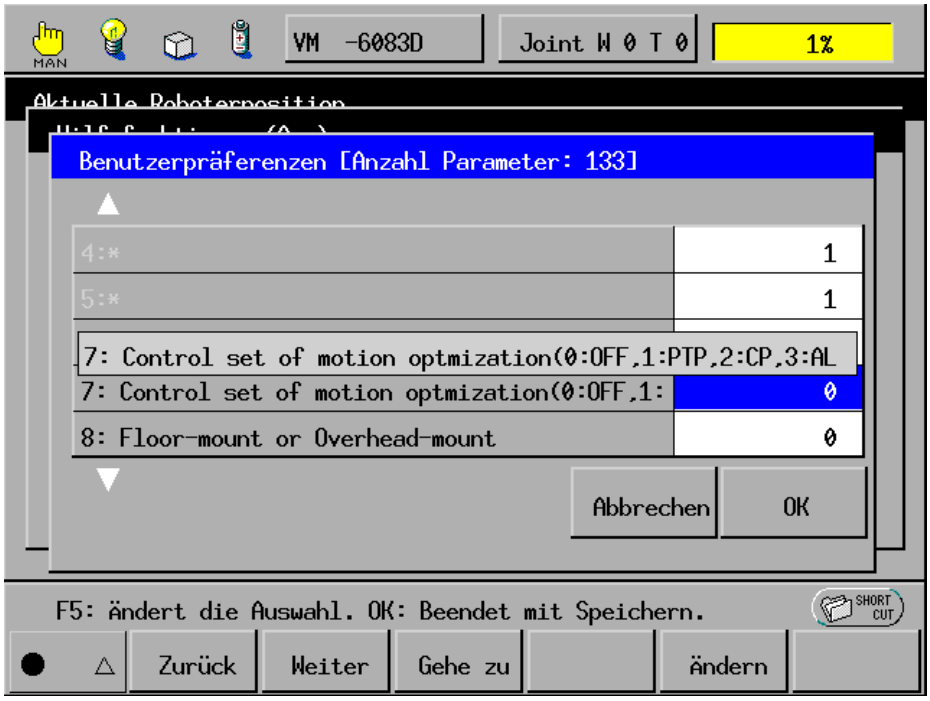
4.7.3 Einstellen der Roboterinstallationsbedingung

Dies ist eine Einstellung zur Verwendung bei Bodenmontage und im “Portal”-Modus. Stellen Sie “0” für die Bodenmontage und “1” für den Portal-Modus ein. Vor Auslieferung ist der Wert auf “0” (Bodenmontage) eingestellt. Diese Einstellung muß geändert werden, wenn der Roboter in einer Überkopfmontage angebracht wird.

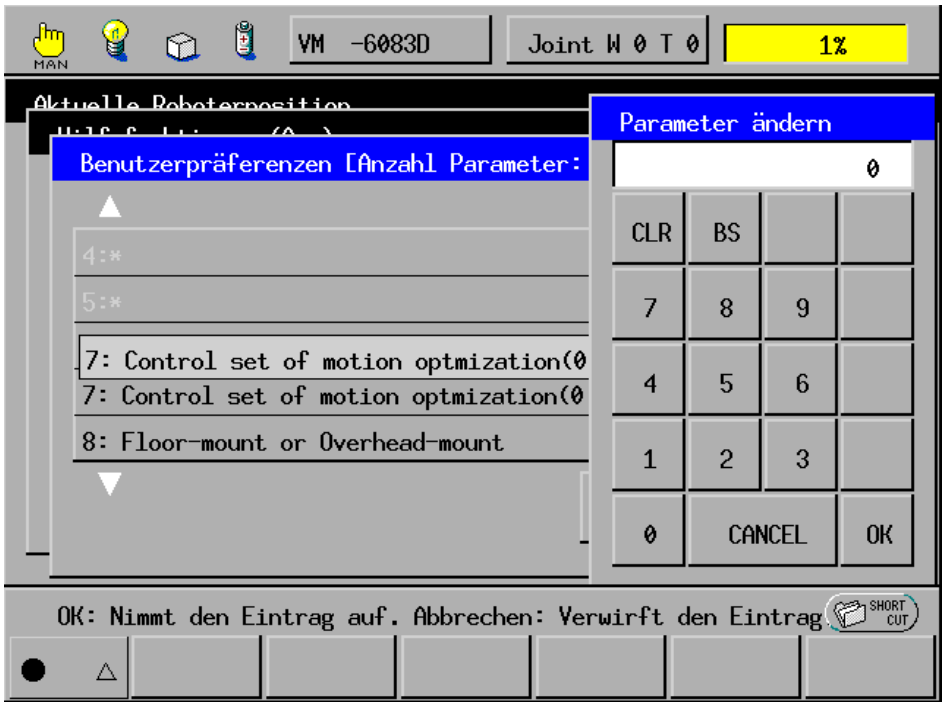
■ Einstellen mit dem Programmiergerät

Bedienungsablauf: Hauptmenü
└[F2 Arm] ┐
[F6 Hilfs.] ┐
└[F7 Konfig.]

Wenn Sie das Programmiergerät verwenden und anhand der folgenden Prozedur vorgehen, wird das Fenster “Benutzereinstellungen” angezeigt, in dem Sie die Master-Steuerparameter wie beispielsweise den Steuersatz der Bewegungsoptimierung und die Masse der Nutzlast einstellen können. (Siehe Abschnitt 2.10 in Kapitel 2 des EINSTELLUNGSHANBUCHES.)



Wählen Sie das Element “Bodenmontage oder Überkopfmontage” in diesem Benutzereinstellungsfenster aus und drücken Sie anschließend [F5 Ändern.], um das numerische Tastenfeld aufzurufen, in dem Sie neue Werte eingeben können.
Geben Sie “0” oder “1” an. Andernfalls wird der folgende Fehler angezeigt: “6003 Effektiver Wertebereich überschritten”.

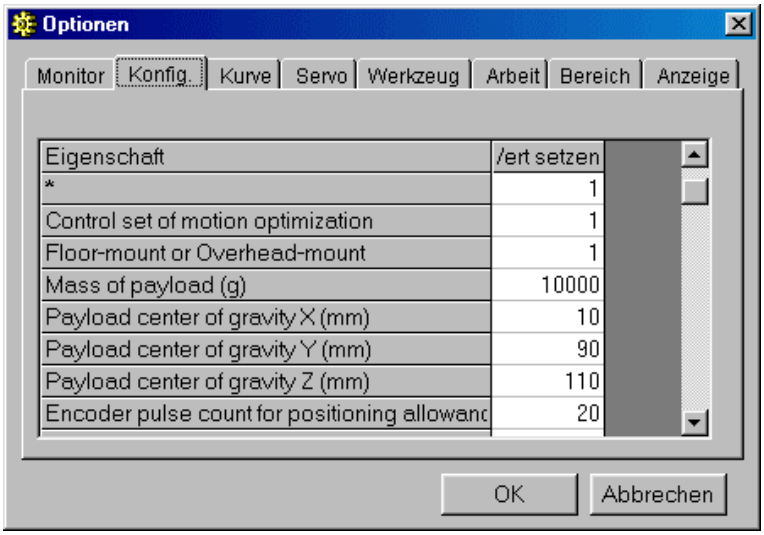


Anmerkung:	**Übertragen Sie das Installationsbedingungs-Set mit dem Programmiergerät an WINCAPSII.** **Weitere Informationen zu der Übertragungsprozedur, siehe Anmerkung (1).**

■ Einstellen mit WINCAPSII

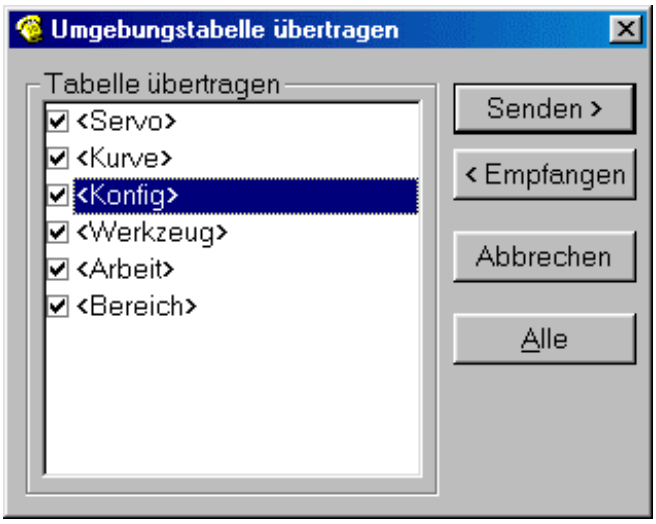
In diesem Abschnitt wird die Methode zum Einstellen der Bodenmontage oder der Portal-Installation mit dem PC-Programmiergerät WINCAPSII erläutert. Weitere Informationen hierzu, siehe Abschnitt 2 in Kapitel 2 des EINSTELLUNGSHANDBUCHES und in Abschnitt 8.6.1.2 in Kapitel 8 des WINCAPSII HANDBUCHES.

Wählen Sie “Optionen” aus dem Werkzeugmenü des Arm Manager aus, um das Optionsfenster anzuzeigen.
Klicken Sie das Register “Benutzereinstellungen” im Optionsfenster an, um das Fenster “Benutzereinstellungen” anzuzeigen.



Wenn das Feld “Wert einstellen” für [Boden oder Portal] in der Anzeige [Bedingung verwenden (Parameternummer:)] doppelt angeklickt wird, kann der Parameterwert geändert werden.

Nachdem Sie die Parameterwerte eingestellt haben, übertragen Sie die Daten an die Robotersteuerung. Schalten Sie zunächst die Stromversorgung des Motors über die Taste MOTOR am Programmiergerät aus. Stellen Sie den Arm Manager auf den Verbindungsstatus ein und klicken Sie **Senden** im Arm Manager an, um die Umgebungstabelle für die Übertragung anzuzeigen. Führen Sie anschließend die Übertragung aus.
Weitere Informationen zur Übertragung, siehe Abschnitt 8.2.5 “Projekt übertragen” in Kapitel 8 des WINCAPSII HANDBUCHES.



Anmerkung (1):	**Klicken Sie < “Empfangen” in der Umgebungstabelle für die Übertragung an, um die Daten aus dem Arm Manager des Controllers zu empfangen. Der externe Lastbedingungswert und der externe Modus kann in diesem Fall empfangen werden. Die in einem Programm geänderten internen Lastbedingungswerte und der interne Modus können jedoch nicht an WINCAPSII gesendet werden.**
Anmerkung (2):	**Geben Sie den Nutzlastschwerpunkt in die Werkzeug-0-Koordinaten ein (siehe Abb. 4-9). Die Einheit sollte mm sein. Der Ursprung des Werkzeug-0-Koordinatensystems fällt zusammen mit dem Flanschmittelpunkt der 6. Achse; die Y-Komponente weist vom Flanschzentrum auf die Bohrung ϕ6H7 (die Richtung des Orientierungsvektors). Die Z-Komponente läuft durch das Flanschzentrum und ist an der Senkrechten zu der Flanschfläche ausgerichtet (die Richtung des Anfahrvektors), und die X-Komponente ist die X-Achse des rechten Koordinatensystems, wobei die Y-Achse der Orientierungsvektor und die Z-Achse der Anfahrvektor sind (siehe Abb. 4-10).**

4.7.4 Einstellen der Initialisierung der optimalen Lastkapazität [V1.4 oder neuer]

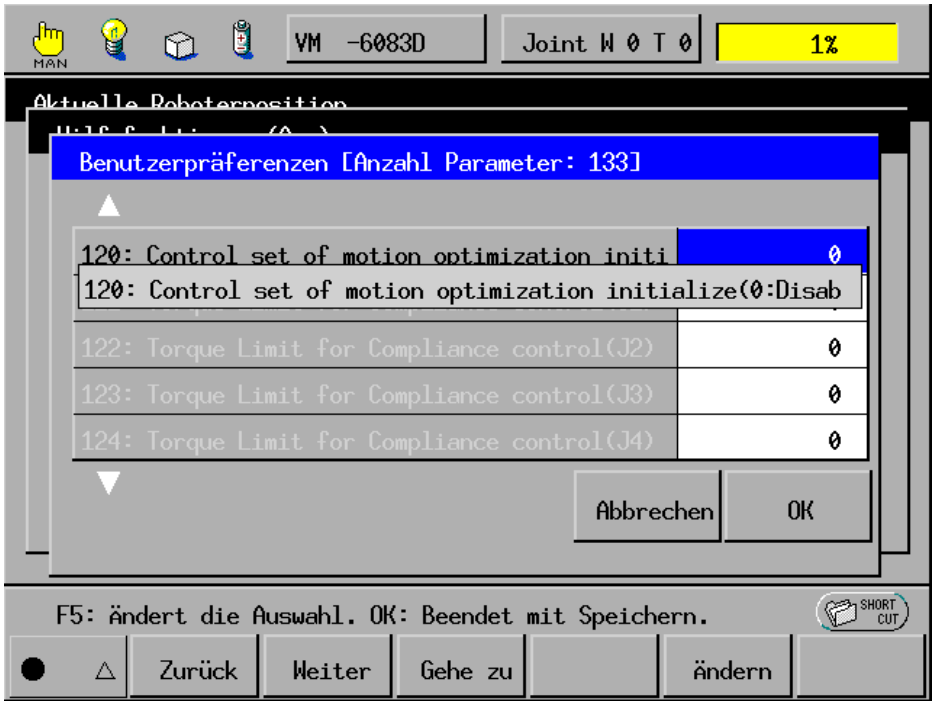
In diesem Abschnitt wird beschrieben, wie der optimale Modus für die Initialisierung der Lastkapazität auf Modus 0 eingestellt oder die aktuelle Einstellung nach dem Einschalten des Controllers beibehalten wird.

Eingestellter Wert	Beschreibung
0	Initialisiert den Einstellungsmodus für die optimale Lastkapazität auf den Modus 0, nachdem der Controller eingeschaltet wurde.
1	Initialisiert den Einstellungsmodus für die optimale Lastkapazität nach dem Einschalten des Controllers nicht (die aktuelle Einstellung wird beibehalten).

■ Einstellen mit dem Programmiergerät

Bedienungsablauf: Hauptanzeige -[F2 Arm] [F6 Hilfs.]-[F7 Konfig.]

Die Anzeige [Benutzereinstellung (Anzahl Parameter:)] erscheint, nachdem Sie mit dem Programmiergerät den oben beschriebenen Bedienungsablauf durchgeführt haben. Auf dieser Anzeige sehen Sie die aktuellen internen Lastbedingungswerte und den internen Modus.



Wählen Sie in der Anzeige [Benutzereinstellung (Anzahl Parameter:)] die Option [Initialisierung für optimale Lastkapazität einstellen] aus und drücken Sie [F5 Änderung einstellen]. Die Anzeige [Parameteränderung] erscheint, und Sie können einzelne Parameterwerte ändern.

0: Deaktiviert→ Führt eine Initialisierung durch, nachdem der Controller eingeschaltet wurde.

1: Aktiviert→ Führt nach dem Einschalten des Controllers keine Initialisierung durch.
(behält die aktuellen Werte bei)

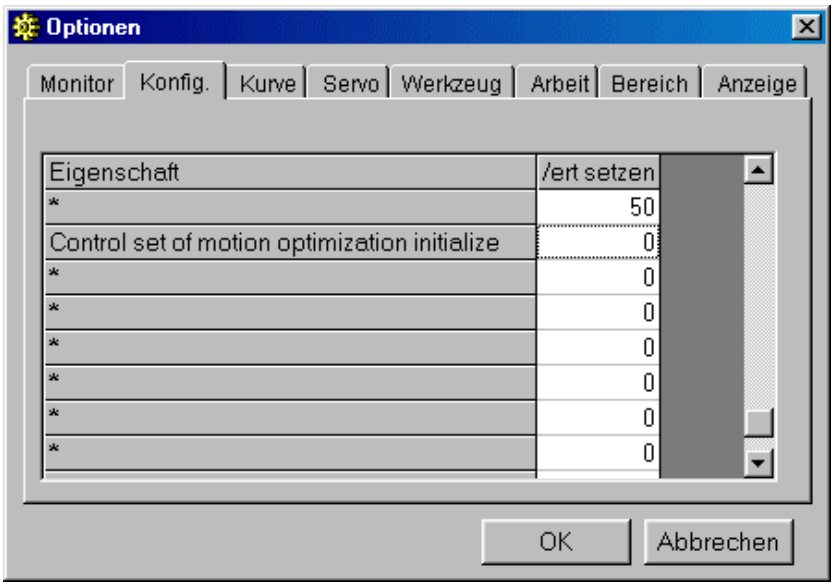
Sie können für die Initialisierung der optimalen Lastkapazität nur 0 oder 1 angeben; andernfalls wird ein Fehler “6003 Gültiger Bereich überschritten” angezeigt.

Anmerkung:	**Sie sollten die mit dem Programmiergerät angegebenen Einstellungen für die Initialisierung der optimalen Lastkapazität an WINCAPSII senden. Hinweise zum Übertragen der Einstellungen finden Sie im Abschnitt 4.7.3 Einstellen mit WINCAPS, Anmerkung (2).**

■ Einstellen mit WINCAPSII

In diesem Abschnitt wird beschrieben, wie Sie WINCAPSII, ein PC-Programmiersystem, zum Einstellen der Initialisierung der optimalen Lastkapazität verwenden können.

Wählen Sie [Einstellen] im Menü [Extras] des Arm Manager aus, um das Fenster [Einstellen aufzurufen].
Klicken Sie das Register [Benutzereinstellungen] im Fenster [Einstellen] an, um die Anzeige [Benutzereinstellungen] aufzurufen.



Klicken Sie ein Einstellungselement für die Initialisierung der optimalen Lastkapazität doppelt an, um die Parameterwerte in dieser Anzeige [Benutzereinstellungen (Anzahl Parameter:)] zu ändern.

Nachdem Sie die Einstellung der Parameter abgeschlossen haben, können Sie die Daten an die Robotersteuerung übertragen. Schalten Sie zunächst die Stromversorgung des Motors über die Taste MOTOR am Programmiergerät aus. Stellen Sie die Verbindung zum Arm Manger her, klicken Sie [Übertragen] an, um das Dialogfenster [Übertragen] anzuzeigen, und führen Sie die Übertragung aus.
Weitere Informationen zur Übertragung siehe Abschnitt 8.2.5 “Übertragen” in Kapitel 8 des “WINCAPS II HANDBUCHES”.

Sie können für die Initialisierung der optimalen Lastkapazität nur 0 oder 1 angeben; andernfalls wird ein Fehler “6003 Gültiger Bereich überschritten” angezeigt.

Anmerkung:	**Wenn Sie [←Empfangen] im Dialogfenster [Übertragen] anklicken, werden Daten im Arm Manager des Controllers empfangen. Die an den Einstellungen der Initialisierung der optimalen Lastkapazität vorgenommenen Änderungen werden nicht an WINCAPSII gesendet.**

.

4.8 Sicherheitsfunktionen

4.8.1 ndTc (Anweisung) [V1.2 oder neuer]

Funktion Stellt die Dauer der WM-Zeit ein.

Syntax ndTc(<TClength>)

Erläuterung Diese Anweisung ist funktional gleichbedeutend mit dem TC-Befehl der konventionellen Sprache.

Die Dauer der WM-Zeit kann in einem Bereich von 0 bis 600 Sekunden angegeben werden. Der werkseitige Standardwert lautet 60 Sekunden.

Die Spezifikation 0 ergibt die gleiche Funktion wie TC OFF in der konventionellen Sprache.

Makrodefinition Erfordert eine Datei <pacman.h>.

Verwandte Begriffe ndTs

4.8.2 ndTs (Anweisung) [V1.2 oder euer]

Funktion Legt die Dauer der WM-Zeit fest und die langsame Geschwindigkeit.

Syntax ndTs(<TSlength>,<Slowspeed>)

Erläuterung Diese Anweisung ist funktional gleichbedeutend mit dem TS-Befehl der konventionellen Sprache.

Die Dauer der TS-Zeit kann in einem Bereich von 3 bis 30 Sekunden angegeben werden. Der werkseitige Standardwert lautet 5 Sekunden.

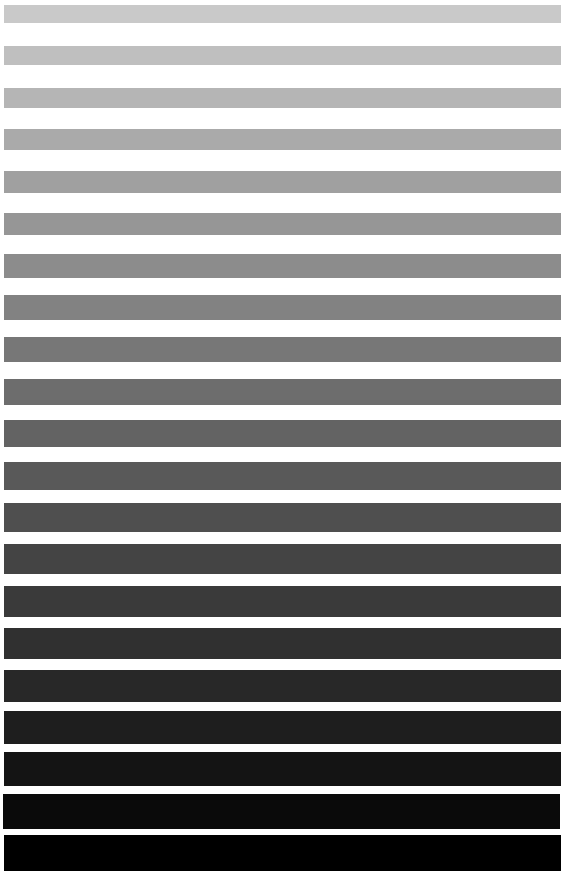
Die langsame Geschwindigkeit kann in einem Bereich von 1 bis 10% angegeben werden. Der werkseitige Standardwert lautet 10%.

Makrodefinition Erfordert eine Datei <pacman.h>.

Verwandte Begriffe ndTc

Kapitel 5

Bildverarbeitungs-steuerung



Dieses Kapitel erläutert die Begriffe zur Bildverarbeitung, die zum Erstellen von Programmen erforderlich sind.

5.1 Bildverarbeitungssteuerung

Dieser Abschnitt erläutert die Befehle zur Verwendung der µBildverarbeitungskarte, die optional im Roboter-Controller eingebaut ist.
Sie sollten dieses Kapitel lesen, wenn Sie mit dem Bildverarbeitungsgerät ein Programm erstellen oder eingeben.

5.1.1 Begriffe der Bildverarbeitungssteuerung

5.1.1.1 Pixel

Die Bildverarbeitungskarte behandelt Elementdaten als eine Gruppe von Punkten. Ein Punkt wird als Bildelement oder Pixel bezeichnet.
Die Anzahl der von der µBildverarbeitungskarte verarbeiteten Pixel kann auf zwei verschiedenen Anzeigen in Pixel ausgedrückt werden. Der Speicher des Prozeßfensters beträgt 512 mal 480 Pixel; der Bildschirmspeicher für das Zeichnungsfenster beträgt 624 mal 480 Pixel.

5.1.1.2 Helligkeit

Jedes von der µBildverarbeitungskarte verarbeitete Pixel von Abbilddaten hat einen Wert, der die Helligkeit angibt. Dieser Wert wird als “Helligkeit” bezeichnet. Der Helligkeitswert hat 256 Stufen von 0 bis 255. Je näher der Wert bei Null liegt, desto dunkler ist das Bild. Umkehrt ist das Bild umso heller, je näher der Wert bei 255 liegt.

5.1.1.3 Fenster

Definieren Sie ein Fenster wie in Abb. 5-1 dargestellt, um den Bildverarbeitungsbereich der µBildverarbeitungskarte darzustellen. Mit dem Fenster wird für jede Fensternummer in der µBildverarbeitungskarte die Größe gespeichert. Das Fenster kann auf zwei Arten geändert werden. Die erste Art ist der Vision Manager, die zweite das Benutzerprogramm. Beachten Sie, daß Fensterinformationen, die über ein Benutzerprogramm geändert werden, beim Ausschalten des Systems verlorengehen.

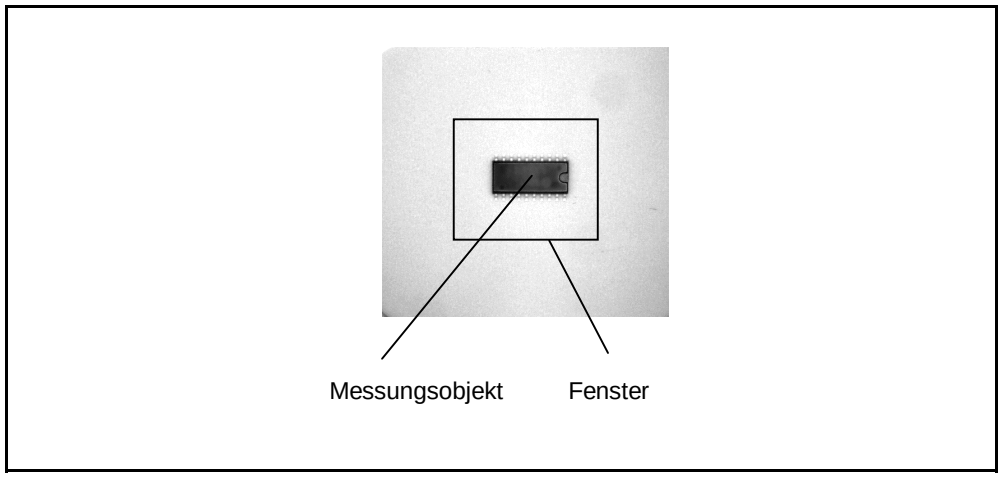


Abb. 5-1 Bildverarbeitungsfenster

5.1.1.4 Bereich, Schwerpunkt und Winkel der großen Achse

[1] Bereich

Die Karte binärisiert die Abbilddaten aus der Kamera und zählt jedes Pixel aus weiß (1) und schwarz (0) im angegebenen Bereich des Fensterbereichs. Die Bildverarbeitungskarte zählt den Bereich durch eine Echtzeit-Binärisierung, ohne den Helligkeitswert der einzelnen Pixel in dem Fenster zu ändern. Sie brauchen daher keine Abbilddaten vorab zu binärisieren.

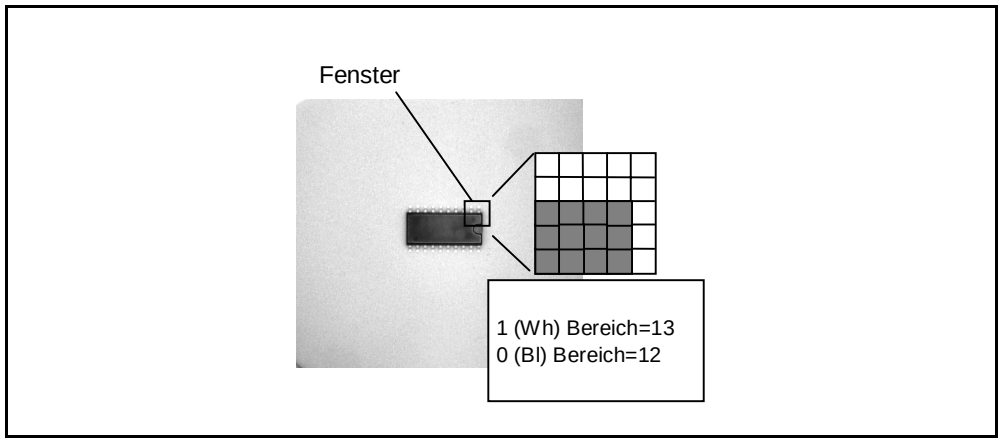


Abb. 5-2 Bereich

[2] Schwerpunkt

Die von der Kamera erfaßten Daten zeigen das Objekt als Ebene an. Der Punkt, an dem das Gewicht des Objekts ausbalanciert ist, wird als Schwerpunkt bezeichnet.
Die µBildverarbeitungskarte ermittelt den Schwerpunkt aus den Pixeln von weiß (1) und schwarz (0) in dem angegebenen Bereich des Fensters. Der Schwerpunkt wird mit den Koordinatenwerten X und Y angegeben.

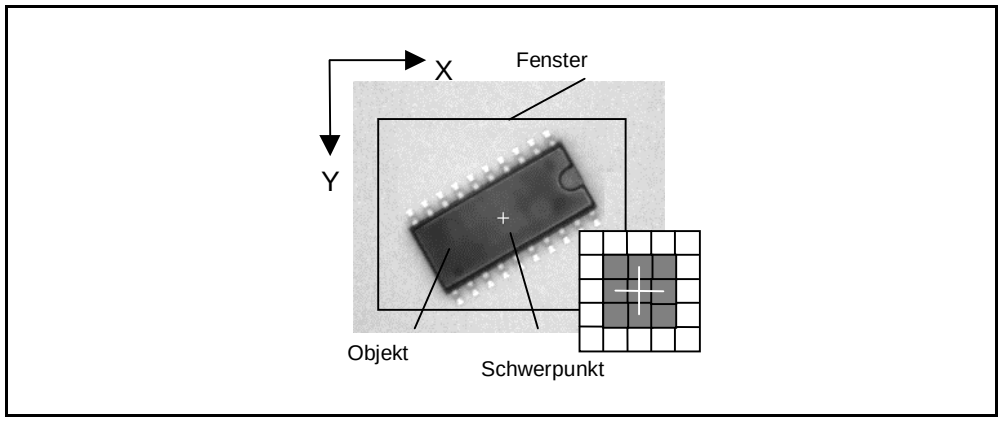


Abb. 5-3 Schwerpunkt

[3] Winkel der großen Achse

Die von der Kamera erfaßten Daten zeigen das Objekt als Ebene an. Wenn das Objekt auf dieser Ebene gedreht wird, wird die Achse in Längsrichtung, die leicht gedreht werden kann, als große Achse der Hauptachse bezeichnet. Die Achse vertikal zu dieser Achse wird als kleine Achse der Hauptachse bezeichnet.
Die µBildverarbeitungskarte definiert einen Winkel (Θ) zwischen der horizontalen Achse (X-Achse) und der großen Achse als Hauptwinkel. Der Winkel der Hauptachse zu einem Objekt in weiß (1) oder schwarz (0) wird im angegebenen Bereich des Fensters nach der Binärisierung der Abbilddaten ermittelt.

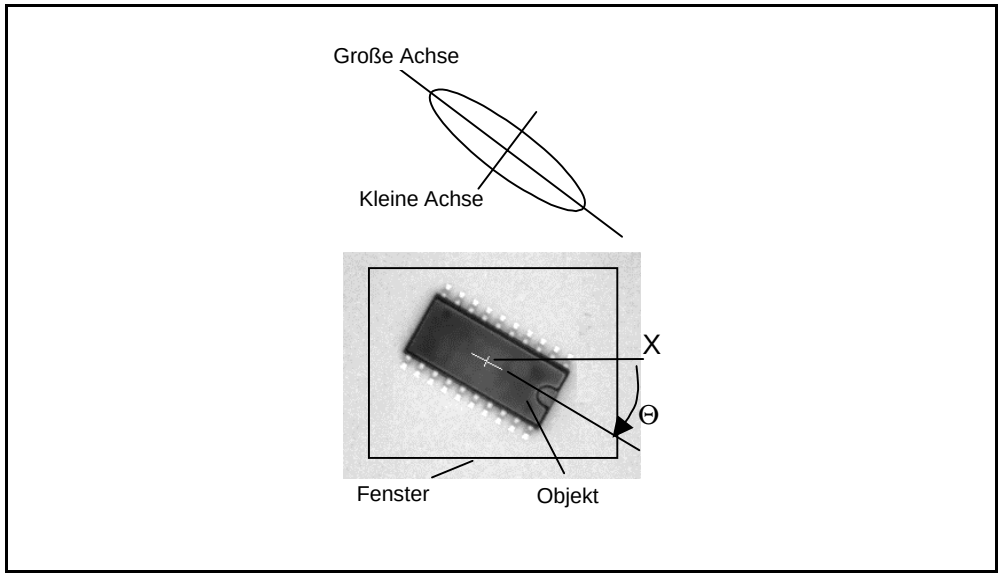


Abb. 5-4 Winkel der Hauptachse

Anmerkung:	**Der Winkel der Hauptachse kann für ein Quadrat oder einen Kreis nicht angegeben werden.**

5.1.1.5 Binärisierung

[1] Binärisierung

Abbilddaten, die in der µBildverarbeitungskarte von der Karte erfaßt wurden, haben 256 Helligkeitsstufen pro Pixel. Die Binärisierung schreibt die Helligkeit jedes Pixels erneut als weiß (1) und schwarz (0) mit einem Schwellenwert als Begrenzung. Dieser Schwellenwert wird als Binärisierungsstufe bezeichnet. Die µBildverarbeitungskarte kennzeichnet die Binärisierungsstufe mit zwei Werten, einem unteren Binärlimit und einem oberen Binärlimit. Die Stufen für die Werte zwischen der unteren Binärbegrenzung und der oberen Binärbegrenzung werden in weiß (1) binärisiert; die anderen Stufen in schwarz (0).

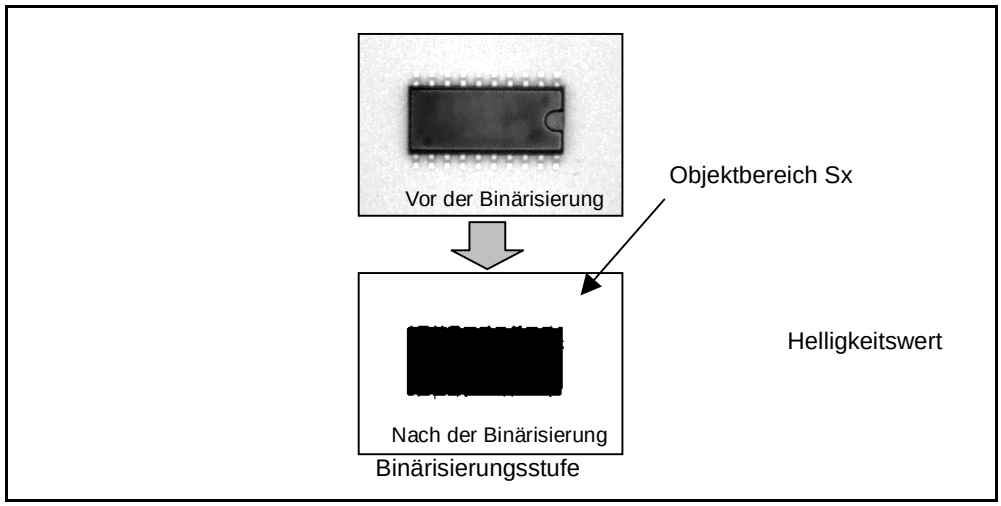


Abb. 5-5 Binärisierung

[2] Histogramm

Ein Histogramm zählt die Häufigkeit des Auftretens von Helligkeitswerten in einem angegebenen Bereich eines Fensters der mit der Kamera erfaßten Abbilddaten. Das Zeichnen des Histogramms als Grafik erleichtert das Verständnis für die Verteilung der Helligkeitswerte; die Binärisierungsstufe kann somit leicht ermittelt werden. Die µBildverarbeitungskarte bietet eine automatische Funktion zur Entscheidung der Binärisierungsstufe. Diese Funktion verwendet Histogramme zum Entscheiden der Binärisierungsstufe. Ein Histogramm von Abbilddaten vor der Binärisierung ist in Abb. 5-6 dargestellt. Sie sehen 2 Bilder, die in ein Objekt und einen Hintergrund getrennt sind. Das Ergebnis der Binärisierung mit unterschiedlichen Binärisierungsstufen ist in Abb. 5-7 dargestellt. Durch die Anpassung der unteren und oberen Binärbegrenzung können Sie andere Binärisierungsergebnisse erzielen.

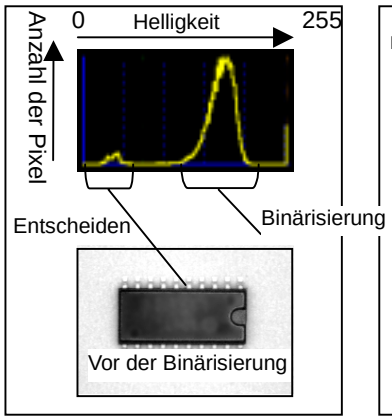


Abb. 5-6 Vor der Binärisierung

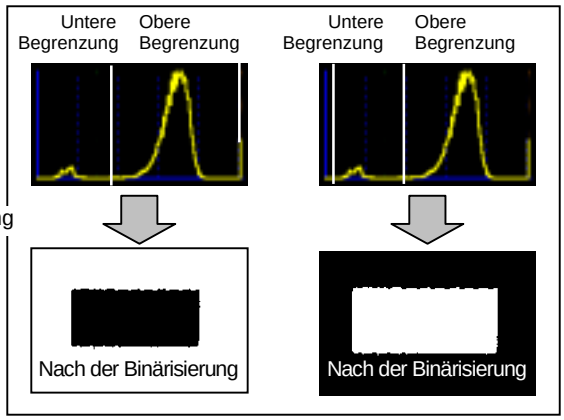


Abb. 5-7 Nach der Binärisierung

[3] Erkennung der Binärisierungsstufe

Modusmethode

Wenn das Histogramm eines Abbildes eine Verteilung mit doppeltem Buckel (zwei Spitzen für die Objektform und den Hintergrund) bildet, wird die Methode zur Einstellung der Binärisierungsstufe (t) auf den Talabschnitt (t) zwischen den beiden Buckeln als Modusmethode bezeichnet.
Da die Bildverarbeitungskarte Spitze und Tal im Histogramm in der Richtung vom Helligkeitswert von 0 zu 255 erkennt (siehe Abb. 5-8), wird Punkt t als Tal erkannt; die Karte definiert diesen Wert als Binärisierungsstufe.

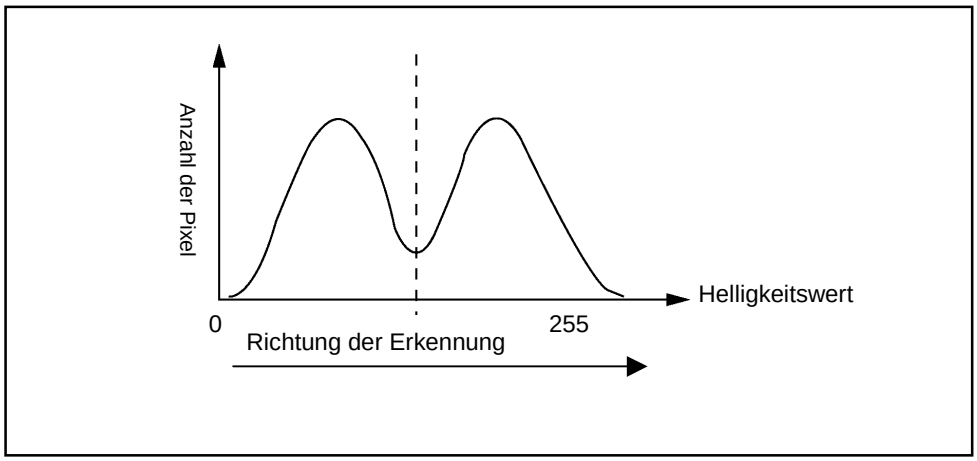


Abb. 5-8 Binärisierungsstufe entsprechend Modusmethode

Unterscheidungsanalysemethode

In einem Histogramm eines Abbildes unterscheiden Sie den Helligkeitswert nach schwarzen und weißen Abschnitten mit der Binärisierungsstufe K. Die Methode zur Bestimmung der Binärisierungsstufe K durch Verwendung der statistischen Methode zur klaren Unterscheidung der beiden Abschnitte wird als Unterscheidungsanalysemethode bezeichnet.
Wenn ein Bild kein Histogramm mit einer doppelten Buckelverteilung ergibt, ist die oben beschriebene Modusmethode nicht geeignet. Sie können den optimalen Schwellenwert stattdessen mit der Unterscheidungsanalysemethode bestimmen.

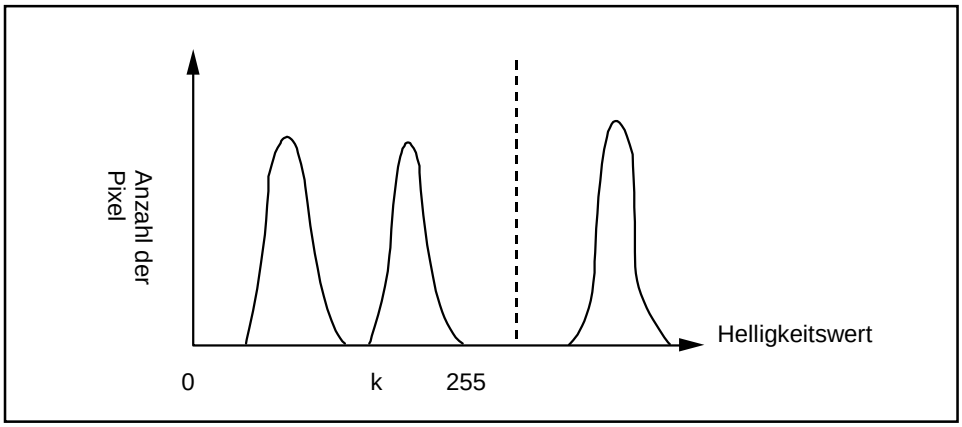


Abb. 5-9 Binärisierungsstufe durch Unterscheidungsanalysemethode

P-Kachelungsmethode

Diese Methode erkennt die Binärisierungsstufe, bei der ein Objektbereich mit Sx zusammenfällt, durch Verwendung eines Histogrammes, wenn die weißen und schwarzen Bereiche des Objekts bereits als Sx bekannt sind.
Wenn das Objekt weiß ist, erhält man den Bereich durch Hinzufügen von Pixeln aus einem höheren Helligkeitswert des Histogrammes. Erreicht das berechnete Ergebnis Sx auf der Helligkeitsstufe, wird der Helligkeitswert auf der Binärisierungsstufe eingestellt.
Wenn das Objekt schwarz ist, erhält man den Bereich durch Hinzufügen von Pixeln aus einem niedrigeren Helligkeitswert des Histogramms. Die Binärisierungsstufe erhält man auf die gleiche Weise wie bei einem weißen Objekt.
Diese Methode ist wirksam, wenn der Objektbereich vorher bekannt ist.

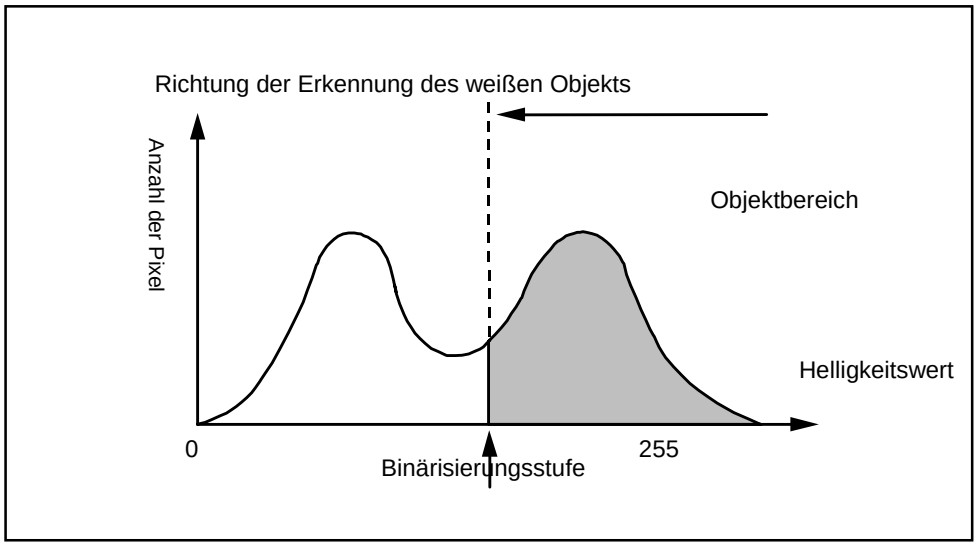


Abb. 5-10 Binärisierungsstufe entsprechend P-Kachelungsmethode

5.1.1.6 Helligkeitsintegralwert

Die Summe aller Helligkeitswerte der Pixel in dem Bereich, der mit dem Fenster für das abgerufene Bild von der Kamera angegeben ist, wird als Helligkeitsintegral bezeichnet. Der Wert der Summe wird als Helligkeits-integralwert bezeichnet.

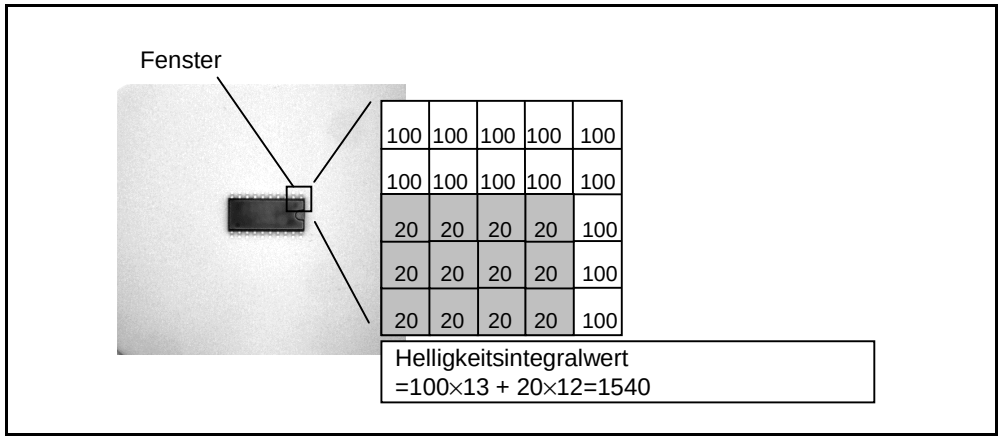


Abb. 5-11 Helligkeitsintegralwert

5.1.1.7 Kante

Ein Punkt, an dem sich das Objekt in einem gekennzeichneten Fenster von dunkel nach hell (schwarz nach weiß) oder hell nach dunkel (weiß nach schwarz) ändert, also der Punkt des Helligkeitswechsels, wird als Kante bezeichnet.
Die von der µBildverarbeitungskarte erkannte Kantenposition ist der Helligkeitswert des Bereicheswertes, der den Wert der angegebenen Stufe erfüllt. Sie können zur Angabe des Stufenwertes einen absoluten Wert und einen differentiellen Wert verwenden. Verwenden Sie diese Angaben entsprechend dem Status des Objekts. Für eine absolute Angabe ermittelt das System eine Stufe, bei der der Helligkeitswert oder Bereichswert durch die angegebene Position verläuft. Bei der Angabe differentieller Werte werden jedoch Positionen erkannt, an denen der geänderte Helligkeitswert oder Bereichswert größer ist als der angegebene Wert.

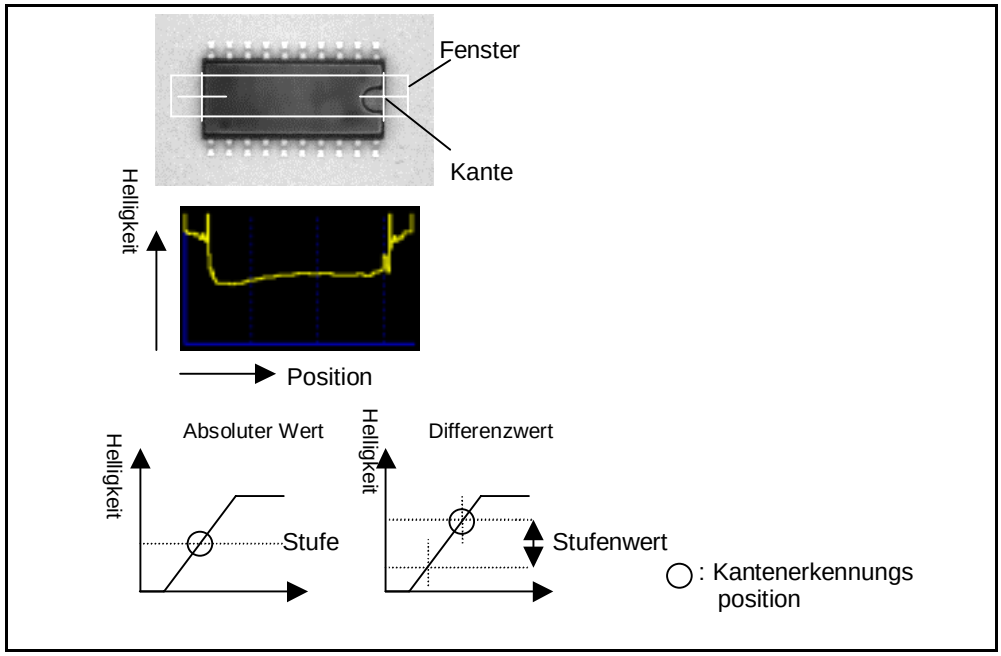


Abb. 5-12 Kante

5.1.1.8 Bezeichnung

Bezeichnung ist der Prozeß der Binärisierung der von der Kamera erhaltenen Abbilddaten und das Anfügen einer Sequenznummer zu den Verbindungsbereichen von weißen (1) oder schwarzen (0) Pixel.

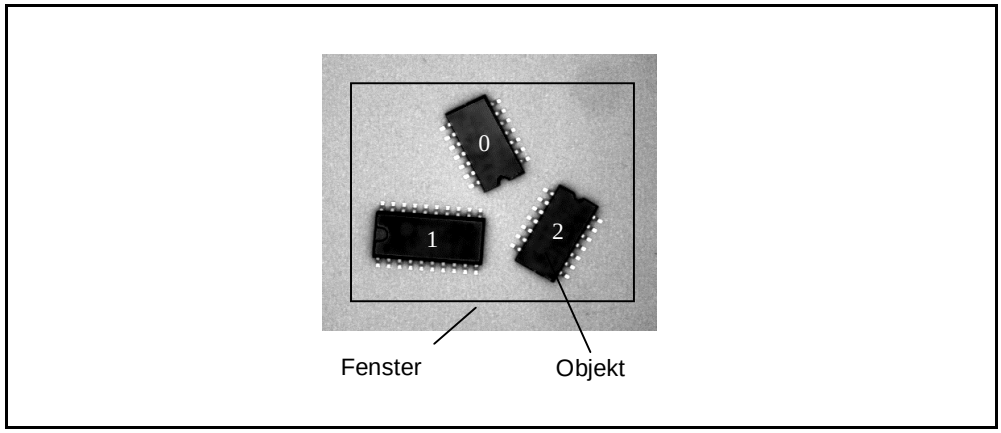


Abb. 5-13 Bezeichnung

Mit Bezeichnungen können Sie mehrere Objekte in dem angegebenen Bereich des Fensters separat und einzeln bearbeiten.
Nach der Ausführung der Bezeichnung kann die µBildverarbeitungskarte den Bereich, den Schwerpunkt, den Winkel der Hauptachse, die Rundungsform und die Peripherielänge als individuelle Objektmerkmale ermitteln. Die Rundungsform ist ein umbeschreibendes Polygon der Objekte.

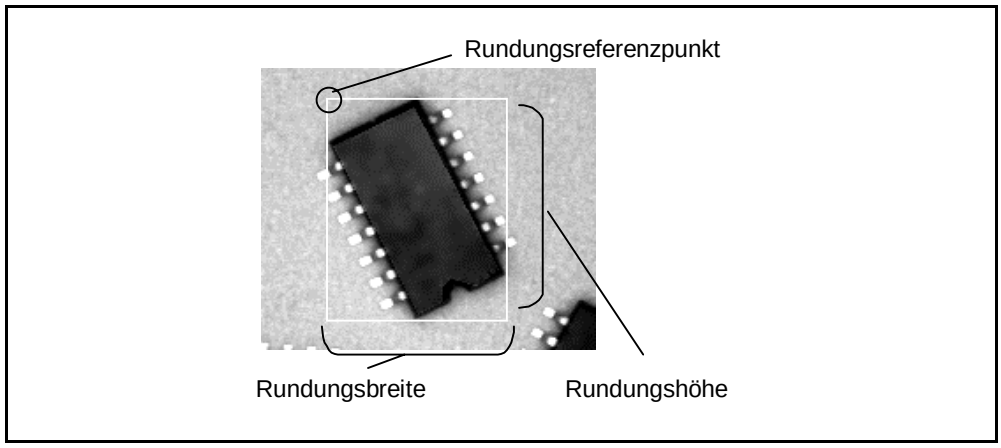


Abb. 5-14 Rundungsform

Die Peripherielänge ist die Summe der Anzahl der Pixel, die die äußere Form des Objekts bilden.

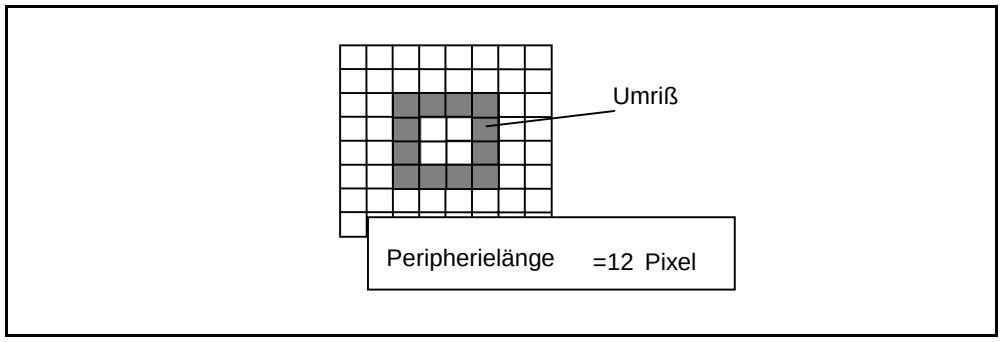


Abb. 5-15 Peripherielänge

5.1.1.9 Suchen

Suchen durchläuft die Standardabbilddaten (Suchmodell), die zuvor in dem Suchbereich (innerhalb des Fensterbereichs) des Messungsobjektbildes gespeichert wurden und findet die Position, die mit dem Suchmodell übereinstimmt.

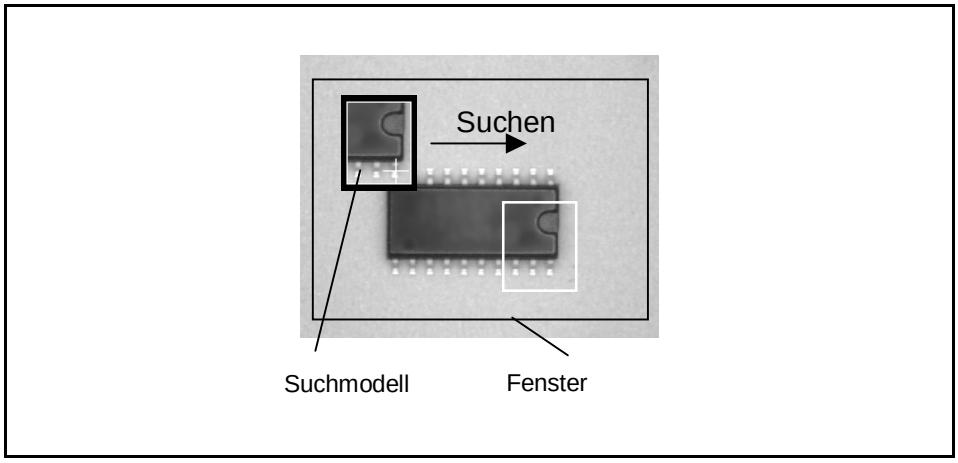


Abb. 5-16 Suchen

In der µBildverarbeitungskarte werden Standardabbilddaten als ein Suchmodell bezeichnet.
Es umfaßt Abbilddaten und die Referenzkoordinaten (OX, OY).

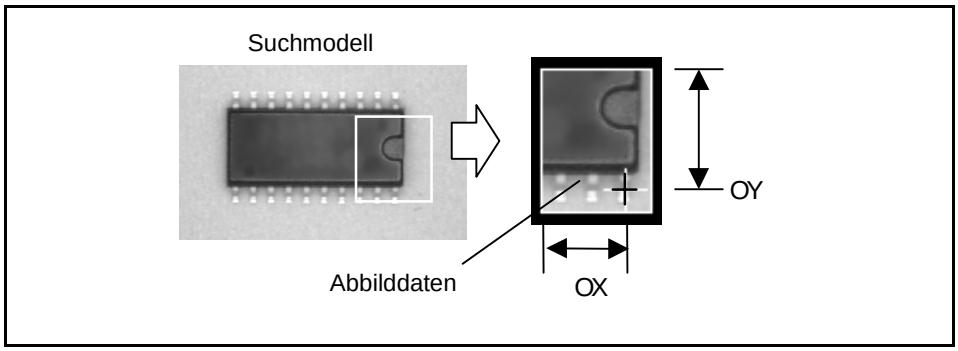


Abb. 5-17 Suchmodell

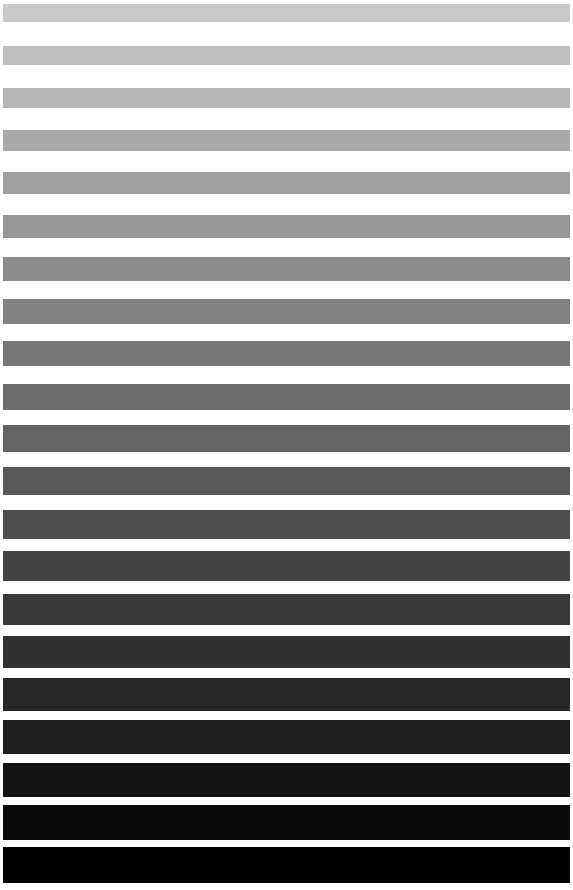
Der Grad, in dem ein angegebenes Suchmodell dem Messungsobjektbild entspricht, wird als Korrespondenzwinkel bezeichnet. Wenn ein Korrespondenzwert größer als der angegebene erhalten wird, kann das System die Koordinaten der Korrespondenzposition des Suchmodells und des Messungsobjektbilds abrufen. Die Erkennungsgenauigkeit der Koordinaten beträgt 1 (die Mindesteinheit für Pixel). Sie können Ergebnisse mit einer Genauigkeit von weniger als 1 Pixel erhalten, indem Sie Subpixel verwenden. Wenn Sie Messungen mit Subpixeln ausführen, dauert die Messung länger als mit Pixeln.

TEIL 2

BEFEHLSREFERENZ

Kapitel 6

Anleitung zur Befehlsreferenz



Dieses Kapitel enthält Befehlsbeschreibungen und eine Befehlsliste für den PAC-Roboter. Verwenden Sie die Befehlsliste, um Informationen zu den einzelnen Befehlen schnell zu finden.

6.1 Erläuterungen zum Format der Befehlsbeschreibungen

Kapitel 9 und die folgenden Kapitel enthalten Beschreibungen aller PAC-Befehle. In diesem Kapitel wird das Format der Befehlsbeschreibungen erläutert.

9.4 HOME Koordinaten

HOME (1) (Anweisung) (2) [Entspricht SLIM]

(3) **2.4.1. Funktion**

Deklariert eine zufällige Koordinatenposition als Ausgangsposition.

(4) **2.4.1.2 Format**

HOME <Positionstyp>

(5) **2.4.1.3 Beschreibung**

Diese Anweisung deklariert zufällige, mit <Positionstyp> angegebene Koordinaten als Ausgangsposition.

Definieren der Ausgangsposition für jedes Programm.

(6) **2.4.1.4 Verwandte Begriffe**

GOHOME

(7) **2.4.1.5 Beispiel**

```
DIM lp1 As Point
HOME(350,0,450,0,0,180)  'deklariert die Koordinaten von
                         '(350,0,450,'0,0,180) als Ausgangsposition.
HOME lp1                 'deklariert die lp1-Koordinaten als
                         'Ausgangsposition.
```

Erläuterungen zum Format der Befehlsbeschreibungen

(1) Befehlsname: Gibt zunächst den Namen des Befehls an.

(2) Befehlstyp: Gibt den Typ des Befehls (Anweisung, Funktion, Systemvariable, integrierte Konstante) nach dem Befehlsnamen an. Wenn der Befehl dem SLIM-Standard entspricht, ist außerdem [Entspricht SLIM] angegeben.

(3) Funktion: Erläutert die Funktion des Befehls.

(4) Format: Beschreibt die Syntax des Beschreibungsformats. Wenn der Befehl ein Argument enthält, ist die Beschreibung des Arguments in “<” und “>” eingeschlossen. Falls es mehrere Argumente gibt, wird “,” als Trennzeichen verwendet.

Elemente, die ignoriert werden können, sind in eckigen Klammern eingeschlossen (“[“ und “]”).

“[,<b>]” in “[” und “]” im Format “[<a>[,<b>]]” bedeutet, daß wenn <a> ignoriert wird, “[,<b>]” ebenfalls ignoriert werden muß.

Wenn in einem Satz von Elementen nur ein einziges Element erforderlich ist, wird der Elementsatz in Klammern “{” und “}” angegeben, und die Argumente werden mit dem Zeichen “|” gegeneinander abgegrenzt.

(5) Erläuterung: Beschreibt die Verwendung des Befehls.

(6) Verwandte Begriffe: Zeigt weitere zugehörige Begriffe oder Erläuterungen.

(7) Beispiel: Zeigt eine Beschreibung zur Verwendung des Befehls an.

6.2 Befehlsliste

6.2.1 Befehle in alphabetischer Reihenfolge

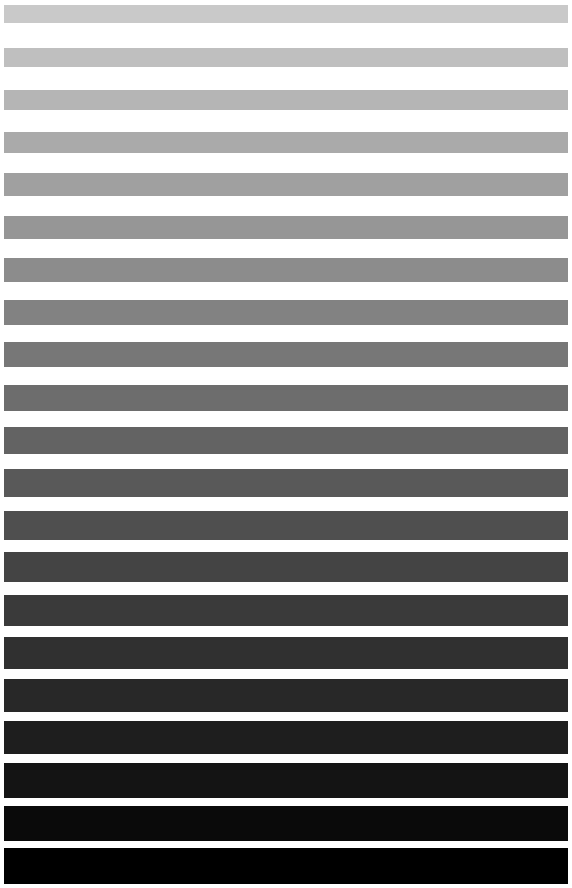
Siehe die alphabetische Befehlsliste im Anschluß an das Inhaltsverzeichnis.

6.2.2 Befehle nach Funktionen geordnet

Siehe die nach Funktionen geordnete Befehlsliste im Anschluß an das Inhaltsverzeichnis.

Kapitel 7

Konfigurations-elemente der PAC-Sprache



Dieses Kapitel erläutert die Elemente, die die PAC-Sprache konfigurieren.

7.1 Die neue Robotersprache PAC

Eine Programmiersprache, mit der Bewegung und Arbeit eines Roboters beschrieben wird, nennt man eine Robotersprache.
Die Robotersprache für DENSO Roboter wird als PAC (Programming Language for Assembly Cell) bezeichnet. PAC wurde neu entwickelt, um die Effizienz bei Entwicklung und Wartung von Steuerungsprogrammen für Roboter gegenüber herkömmlichen Sprachen zu verbessern. Die wichtigsten Funktionen sind unten beschrieben.

- Sie ist aufwärtskompatibel mit der Industrie-Robotersprache SLIM nach JIS.
- Sie ist leicht lesbar, weil es sich um eine strukturierte Programmiersprache handelt; dies erleichtert auch die Entwicklung und Wartung.
- Es können nicht nur Roboterprogramme beschrieben werden, sondern auch die Steuerung des Bildverarbeitungsgeräts ist mit PAC universell.
- Dank der Multitasking-Funktion ist die Programmverarbeitung sehr effektiv.
- Als Ergebnis der Funktion zur Verarbeitungsunterbrechung kann eine Ausnahmeverarbeitung, beispielsweise beim Auftreten eines Fehlers, sehr effizient beschrieben werden.

7.2 Beziehung zwischen der PAC-Robotersprache und konventionellen Sprachen

Üblicherweise gibt es viele Probleme bei Robotersprachen, wie z.B. eine Inkompatibilität zwischen verschiedenen Robotern und Herstellern. JIS hat die Industrieroboter-Programmiersprache SLIM eingeführt, um Robotersprachen zu standardisieren und damit die Mühe beim Verwenden verschiedener Sprachen für verschiedene Roboter zu vermeiden.

SLIM wurde mit dem Ziel standardisiert (JIS B 8439), hauptsächlich Montagearbeiten zu beschreiben. Datentypen und Roboterbewegungsanweisungen für einen bestimmten Roboter werden in BASIC geschrieben.

Ein Vorteil von SLIM ist daher, daß auch Personen, die an Robotersprachen vor SLIM gewöhnt sind, die SLIM Sprache vergleichsweise leicht erlernen können.

Die für DENSO Roboter verwendete PAC-Robotersprache basiert auf der strukturierten BASIC-Sprache, einer aufwärtskompatiblen BASIC-Sprache. Dadurch erhält PAC die Vorteile von SLIM, hat aber zusätzlich verschiedene andere Funktionen, wie eine strukturierte Sprache und Multitasking-Funktion. Da die Spezifikationen für Roboter-Anweisungen dieselben sind wie bei SLIM, ist PAC auch aufwärtskompatibel zur SLIM Sprache. PAC kann hohen Anforderungen durch Nutzung der Vorteile von SLIM so gerecht werden, daß es auch von Personen, die an Robotersprachen vor SLIM gewöhnt sind, leicht beherrscht werden kann.

7.3 Sprachelemente

Die folgenden Elemente werden zur Konstruktion der PAC-Sprache verwendet.

- Kennung ・・・ Der Name, der ein Konstruktionselement kennzeichnet.
- Variable・・・・ Zum vorübergehenden Speichern von Daten.
- Konstante ・・ Daten mit einem konstanten Wert.
- Operator ・・・ Ein Symbol zur Berechnung von zwei Werten.
- Ausdruck ・・ Eine Kombination von Konstruktionselementen zum Ermitteln eines Wertes.
- Befehl ・・・・・ Eine in der PAC-Sprache integrierte Anweisung zur Ausführung von Prozessen.

Es gibt nur wenige Typen von Variablen und Konstanten.
Dieses Kapitel erläutert die einzelnen Konstruktionselemente und Datentypen.
Für Informationen zu Befehlen siehe Teil 2 “Befehlsreferenz.”

7.4 Name

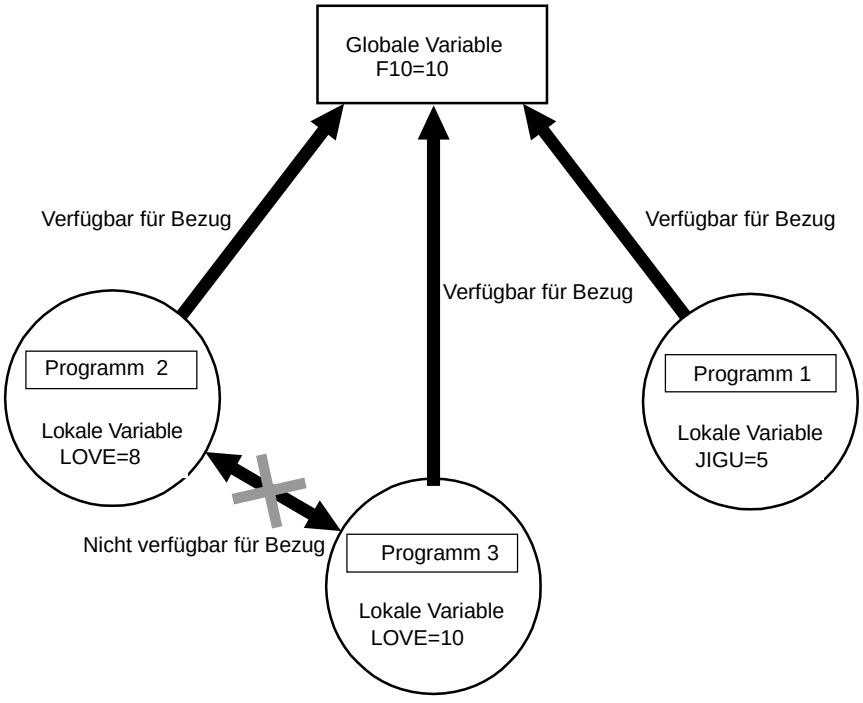
Die PAC-Sprache hat Regeln zur Identifizierung verschiedener Elemente in einem Programm. In diesem Kapitel werden diese Regeln erklärt. Namen von Befehlen, Variablen, Funktionen, Kennungen und Programmen richten sich nach folgenden Regeln.

- Ein Name muß mit einem Zeichen (ein-Byte Alphabet, keine Unterscheidung zwischen Groß- und Kleinbuchstaben) oder einem geregelten Symbol beginnen.
- Buchstaben, Ziffern und Unterstreichungszeichen können für Namen verwendet werden.
- Das erste Zeichen eines Namens muß ein alphabetisches Zeichen sein.
- Punkte, Schrägstriche, umgekehrte Schrägstriche, Leerzeichen, Doppelpunkte, Semikolons, einfache und doppelte Anführungszeichen sowie Sterne dürfen nicht verwendet werden.
- Zeichen wie +, -, *, /, (,), die als Operatoren verwendet werden, sind für Namen nicht zulässig.
- Zur Unterscheidung zwischen Namen und anderen Worten setzen Sie ein Leerzeichen zwischen den Namen und die anderen Worte.
- Die maximale Anzahl zulässiger Zeichen für einen Namen beträgt 64.

7.5 Kennung

7.5.1 Variable

Eine Variable wird verwendet, um in einem Programm genutzte Daten temporär zu speichern. Es gibt globale Variablen, lokale Variablen und Systemvariablen.
Eine globale Variable kann von allen Programmen (Tasks) gemeinsam verwendet werden.
Eine lokale Variable gilt nur für ein Programm. Selbst wenn ein anderes Programm, das gleichzeitig ausgeführt wird, eine Variable mit dem selbem Namen hat, arbeitet diese nur mit dem Programm zusammen, zu dem sie gehört und beeinflußt nicht die anderen Programme.



Unerwartete Vorkommnisse können vermieden werden, da die lokale Variable die Variablen anderer Programme nicht stört.

Globale Variable und Lokale Variable

[1] Globale Variable

Der Name einer globalen Variable wird mit einem Buchstaben des Alphabets (I, F, D, S, V, P, J, T, IO) angegeben, der den Typ angibt, und einem ganzzahligen Ausdruck hinter dem Buchstaben. Nur eine E/A-Variable hat zwei Buchstaben (IO).
Zum Beispiel bezeichnen F0001, F1 und F[1] alle dieselbe Real-Variable mit einfacher Genauigkeit.
Da Variablennamen vom System reserviert sind, können sie ohne Deklaration benutzt werden. Die folgenden Typen können für globale Variablen verwendet werden.

- Typ I: Ganzzahliger Typ (Wertebereich: -2147483648 ~ + 2147483647)
  Beispiel) I0001, I1, I[1]
- Typ F: Real-Typ mit einfacher Genauigkeit
  (-3,402823E + 38 ~ 3,402823E + 38)
  Beispiel) F0001, F1, F[1]
- Typ D: Real-Typ mit doppelter Genauigkeit
  (-1,79769313486231D + 308 ~1,79769313486231D + 308)
  Beispiel) D0001, D1, D[1]
- Typ S: Zeichenfolge (maximal 247 Zeichen)
  Beispiel) S0001, S1, S[1]
- Typ V: Vektortyp (X, Y, Z)
  Beispiel) V0001, V1, V[1]
- Typ P: Positionstyp (X, Y, Z, RX, RY, RZ, FIG) (6 Achsen)
  Beispiel) P0001, P1, P[1]
- Typ J: Gelenk-Typ (J1, J2, J3, J4, J5, J6) (6 Achsen)
  Beispiel) J0001, J1, J[1]
- Typ T: Homogene Transformation
  (Px, Py, Pz, Ox, Oy, Oz, Ax, Ay, Az, FIG)
  Beispiel) T0001, T1, T[1]
- Typ E/A: E/A-Typ
  Beispiel) IO0001, IO1, IO[1]

ANMERKUNG:	Die Typen V, P, J und T sind mit Bildverarbeitungsgeräten µVision-21 nicht verfügbar.

Globale Variable Indirekter Bezug

Wenn eine globale Variable bezeichnet wird, wird die Variablennummer mit einem Ausdruck versehen. Dies nennt man einen indirekten Bezug.
Wenn ein indirekter Bezug ausgeführt wird, wird die in [ ] eingeschlossene Variablennummer ausgedrückt.

Beispiel:

```
I1 = I[5*3]     'Weist I1 den Wert von I15 zu.

F1 = F[I1]      'Weist F1 die Variable vom Typ F mit dem Wert I1 zu.

D1 = D[I1+1]    'Weist D1 die Variable vom Typ D mit dem Wert I1 plus 1 zu.

S1 = S[I7]      'Weist S1 die Variable vom Typ S mit dem Wert I7 zu.

V1 = V[I1]      'Weist V1 die Variable vom Typ V mit dem Wert I1 zu.

MOVE L, P[I5]   'Bewegt zu einer Position mit Variable vom Typ P mit dem Wert I5.

J1 = J[I5*3]    'Weist I5 die Variable vom Typ J mit einem Wert zu, der das 3-fache 'von
                I1 ist.

T1 = T[I1*3]    'Weist T1 die Variable vom Typ T mit einem Wert zu, der das 3-fache 'von
                I1 ist.

SET IO[I1*5]    'Setzt einen Anschluß vom Typ Bit, der das 5-fache von I1 ist, auf ON.
```

[2] Lokale Variable

Die folgenden Variablentypen können für lokale Variablen in der selben Weise verwendet werden wie globale Variablen.

- Typ I: Ganzzahliger Typ (Wertebereich: -2147483648 ~ +2147483647)
- Typ F: Real-Typ mit einfacher Genauigkeit (-3,402823E + 38 ~ 3,402823E + 38)
- Typ D: Real-Typ mit doppelter Genauigkeit (-1,79769313486231D + 308 ~ 1,79769313486231D + 308)
- Typ S: Zeichenfolge (maximal 247 Zeichen)
- Typ V: Vektor (X, Y, Z)
- Typ P: Position (X, Y, Z, RX, RY, RZ, FIG) (6 Achsen)
- Typ J: Gelenk (J1, J2, J3, J4, J5, J6) (6 Achsen)
- Typ T: Homogene Transformation (Px, Py, Pz, Ox, Oy, Oz, Ax, Ay, Az, FIG)
- Typ E/A: E/A-Typ

Lokale Variablen können verwendet werden, nachdem mit Befehlen zur Typdeklaration die Deklaration ausgeführt wurde.
Die Typdeklaration kann auch mit den Typdeklarationszeichen für lokale Variablen vom Typ Numerischer Wert oder Zeichenfolge ausgeführt werden. Die Befehle und Zeichen zur Typdeklaration sind in Abschnitt 8.6 “Deklarationsanweisung” beschrieben.

Anmerkung (1): **Wenn eine Variable ohne Typdeklaration verwendet wird, funktioniert sie wie eine Variable des Real-Typs mit einfacher Genauigkeit. Jedoch kann eine ohne Typdeklaration verwendete Variable zu einem Programmfehler führen. Daher sollte, wenn möglich, die Typdeklaration ausgeführt werden.**

Anmerkung (2): **In der Standardeinstellung des PC-Programmiergeräts WINCAPSII ist “Explizite Typdeklaration immer erforderlich” eingestellt. Wenn dies eingestellt ist und die Typdeklaration ignoriert wird, tritt beim Kompilieren ein Fehler auf. Wenn keine speziellen Gründe dies verhindern, nutzen Sie das System mit dieser Einstellung.**

Anmerkung (3): **Falls im Programm auf andere Variablen als die vom Typ E/A verwiesen wird, muß der lokalen Variablen ein bestimmter Wert zugewiesen werden (Initialisierung der Variablen), bevor auf sie zugegriffen wird. Wenn auf eine Variable ohne Initialisierung der lokalen Variablen verwiesen wird, tritt bei der Ausführung ein Fehler auf. Variablen können durch Beschreiben einer Zuweisungsanweisung für die Variable im Programm im voraus initialisiert werden, oder durch Verwenden von DEFINT A = 1, zum Beispiel, um sie im Variablendeklarationsabschnitt des Programmes für Variablen, die keine Array-Variablen sind, zu beschreiben.**

Anmerkung (4): **Wenn Sie das Programm erneut starten, wenn ein Programm mit lokalen Variablen mit CALL aufgerufen und ausgeführt wird bzw. sich im Warte- oder Unterbrechungsstatus befindet, sollte das System zwei Tasks haben: Eine Task wird vom ersten Aufruf generiert, die zweite beim Neustart. Obwohl dies verschiedene Tasks sind, kann das selbe Programm die Unabhängigkeit der lokalen Variablen nicht sichern und der Roboter kann daher unerwartete oder falsche Aktionen ausführen.**
Um dies zu vermeiden, starten Sie das Programm nicht selbst, wenn das Programm bereits vom CALL Befehl aufgerufen wird.

Anmerkung (5): **Die Typen V, P, J und T sind mit Bildverarbeitungsgeräten µVision-21 nicht verfügbar.**

[3] Systemvariable

Eine Systemvariable wird zum Prüfen des Systemstatus verwendet. Da die Variablennamen vom System reservierte Wörter verwenden, muß der Variablenname nicht deklariert werden. Es gibt folgende Systemvariablen.

CURJNT CURPOS CURTRN CURFIG CURACC CURDEC CURJDEC CURJSPD CURSPD DESTJNT DESTPOS DESTTRN DATE$ TIME$ TIMER ERL ERR

7.5.2 Funktion

Eine Funktion wird verwendet, um das Operationsergebnis der im voraus bestimmten Operationsmethode für den angegebenen Wert (Argument) zu erhalten. Es gibt einige Funktionen, die je nach Funktionstyp kein Argument haben.
In der PAC-Sprache gibt es Funktionen zur Bearbeitung von Stellungs- und Vektordaten, von Daten Numerischer Werte, von Zeichenfolgen, und eine benutzerdefinierte Funktion.
Weitere Informationen zu Funktionsbefehlen, siehe Kapitel 15 “Befehlsreferenz”.

7.5.3 Kennung

Eine Kennung drückt die Anweisungsposition in einem Programm aus. Die Zielposition einer Programmverzweigung kann mit einer Kennung bezeichnet werden.
Wenn für die Kennung ein eindeutiger Name vergeben wird, kann das Programm leicht verstanden werden.

7.5.4 Programm

Ein Programm kann durch einen Programmnamen bezeichnet werden und andere Programme aufrufen.
Deklarieren Sie einen Programmnamen zu Beginn eines Programmes mit dem PROGRAM Befehl.
Weitere Informationen zu Programmnamen, siehe Abschnitt 8.2 “Programmname und Deklaration” in Kapitel 8 dieses Handbuches.

Programmbeispiel: Programmaufruf mit einem Programmnamen

Programm auf der Programmaufrufseite

```
PROGRAM PRO1
    IF IO[138] = ON THEN
        CALL MOTION
    ELSE
        DELAY 200
    END IF
END
```

“MOTION”-Programm auf der aufgerufenen Seite.

```
PROGRAM MOTION
    TAKEARM
    SPEED 100
    MOVE P, P1
    DELAY 200
    MOVE P, P2
END
```

7.6 Datentyp

Die PAC-Sprache arbeitet mit folgenden Datentypen: Zeichenfolge, Numerischer Wert, Position, Vektor und E/A. Diese Datentypen werden unten beschrieben.

7.6.1 Zeichenfolge

Daten vom Typ Zeichenfolge (Zeichenf (String)) werden auch Daten vom Typ S genannt.
Sie können maximal 247 Zeichen beinhalten.

7.6.2 Numerischer Wert

Die folgenden 3 Datentypen gehören zum Typ Numerischer Wert.

Typ I: Ganzzahliger Typ (Wertebereich: -2147483648 ~ +2147483647)
Typ F: Real-Typ mit einfacher Genauigkeit
(-3,402823E + 38 ~ 3,402823E + 38)
Typ D: Real-Typ mit doppelter Genauigkeit
(-1,79769313486231D + 308 ~ 1,79769313486231D + 308)

7.6.3 Vektor

Der Vektor-Datentyp wird auch Typ V bezeichnet.
Er umfaßt drei Parameter vom Real-Typ mit einfacher Genauigkeit der Komponenten X, Y und Z.

Typ V: Vektor (X, Y, Z)

7.6.4 Stellung

Die folgenden 3 Datentypen gehören zum Typ Stellung.

Typ P: Position (X, Y, Z, RX, RY, RZ, FIG)
Typ J: Gelenk (J1, J2, J3, J4, J5, J6)
Typ T: Homogene Transformation
(Px, Py, Pz, Ox, Oy, Oz, Ax, Ay, Az, FIG)

7.6.5 E/A (EIN/AUS)

Daten vom Typ E/A haben einen E/A-Anschlußstatus (EIN oder AUS (ON oder OFF)) als Wert.

7.7 Datentypumwandlung

Die Änderung des Datentyps in verschiedene andere Datentypen ist ebenso möglich.

7.7.1 Numerischer Wert

Daten vom Typ Numerischer Wert können nach den folgenden Regeln umgewandelt werden.

- Wenn Daten eines numerischen Wertes einer Variablen eines anderen numerischen Datentyps zugeordnet werden, wird der numerische Wert umgewandelt, so daß er dem Typ der Variablen entspricht.
- Wenn eine Operation mit einem anderen Typ eines numerischen Wertes ausgeführt wird, erfolgt diese Operation derart, daß der Typ mit der höheren Genauigkeit verwendet wird.
- In einer logischen Operation werden numerische Werte in Ganzzahlen umgewandelt, und die Operation wird ausgeführt. Das Ergebnis ist eine ganze Zahl.
- Bei der Umwandlung eines Real-Typs in eine Ganzzahl wird das Ergebnis auf die maximale Ganzzahl, die das Ergebnis nicht überschreitet, abgerundet.
- Bei der Zuordnung eines Real-Typs mit doppelter Genauigkeit zu einem Real-Typ mit einfacher Genauigkeit wird das Ergebnis auf 7 signifikante Stellen abgerundet.

7.7.2 Zeichen und Numerischer Wert

Die PAC-Sprache enthält die folgenden Befehle zur Umwandlung von Zeichen, Zeichenfolgen, Zeichencodes und numerischen Werten.

Umwandlungsbefehle für Zeichen und numerische Werte

Nach Umwandlung / Vor Umwandlung	Umgewandelt in Zeichencode	Umgewandelt in Zeichenfolge	Umgewandelt in numerischen Wert	Umgewandelt in Zeichenfolge in Binärdarstellung	Umgewandelt in Zeichenfolge in hexadezimaler Darstellung
Zeichencode		CHR$	-	-	-
Zeichen	ASC		VAL$	-	-
Numerischer Wert	-	STR$		BIN$	HEX$

7.7.3 Stellungstyp-Daten

Die Typumwandlung von Stellungstyp-Daten kann mit den folgenden Funktionen ausgeführt werden. Die Werkzeug- und Arbeitskoordinatensysteme finden sich beim Ausführen der Typumwandlung in den Stellungstyp-Daten wieder.

Umwandlungsbefehle für Daten des Typs Stellung

Nach Umwandlung / Vor Umwandlung	Umgewandelt in Typ P	Umgewandelt in Typ J	Umgewandelt in Typ
Typ P		P2J	P2T
Typ J	J2P		J2T
Typ T	T2P	T2J	

```
Beispiel:  J0 = P2J (P0)
           T0 = P2T (P0)
           P0 = J2P (J0)
           T0 = J2T (J0)
           P0 = T2P (T0)
           J0 = T2J (T0)
```

7.8 Konstante

Eine Konstante ist ein Ausdruck mit einem festen Wert.
Konstanten in der PAC-Sprache sind wie folgt klassifiziert.

1	Daten vom Typ Numerischer Wert	Typ I:	Konstante vom Typ Ganze Zahl
		Typ F:	Konstante vom Typ Real mit einfacher Genauigkeit
		Typ D:	Konstante vom Typ Real mit doppelter Genauigkeit
2	Zeichenfolge-daten	Typ S:	Zeichenfolgekonstante (maximal 247 Zeichen)
3	Daten vom Typ Vektor	Typ V:	Konstante vom Typ Vektor (X,Y,Z)
4	Daten vom Typ Stellung	Typ P:	Konstante vom Typ Position (X,Y,Z,RX,RY,RZ,FIG) (bei 6 Achsen)
		Typ J:	Konstante vom Typ Gelenk (J1,J2,J3,J4,J5,J6) (bei 6 Achsen)
		Typ T:	Konstante vom Typ Homogene Transformation (Px,Py,Pz,Ox,Oy,Oz,Ax,Ay,Az,FIG) (Jedes Element des Typs V, P, J und T ist ein Real-Typ mit einfacher Genauigkeit.)

Konstanten jedes Typs werden im folgenden erläutert.

7.8.1 Konstante vom Typ Numerischer Wert

[1] Konstante vom Typ Ganze Zahl

Diese Konstante ist eine ganze Zahl zwischen -2147483648 und +2147483647.
Es gibt 3 Arten von Notationen für Konstanten vom Typ Ganze Zahl: dezimal, binär und hexadezimal.

Dezimalformat

Dies ist eine Konstante vom Typ Ganze Zahl, die mit Dezimalzahlen ausgedrückt wird.
Wenn Sie eine Abschlußstellung "%" vom Typ Ganze Zahl zu einem Real-Typ innerhalb eines Bereiches von -2147483648 bis +2147483647 addieren, wird sie als Konstante vom Typ Ganze Zahl betrachtet, wobei die Ziffern nach dem Dezimalzeichen abgerundet werden.

```
Beispiel: 32767
          -125
          +10
          3256.21% → Da diese Abschlußstellung den TypGanze
                     Zahl hat, wird sie als 3256
                     betrachtet.
```

Binärformat

Dies ist eine Konstante vom Typ Ganze Zahl, die mit binären Zahlen ausgedrückt wird.
Sie wird als Anordnung der binären Ziffern 0 und 1 ausgedrückt und am Anfang des numerischen Wertes mit "&B" gekennzeichnet.
Wenn der Wertebereich des numerischen Wertes einer Konstante vom Typ Ganze Zahl im binären Format ausgedrückt wird, beträgt er
&B0~&B11111111111111111111111111111111
(bei 32 Bit-Wertebereich).

```
Beispiel: &B110
          &B0011
```

Hexadezimalformat

Dies ist eine Konstante vom Typ Ganze Zahl, die mit hexadezimalen Zahlen ausgedrückt wird.
Sie wird als Anordnung von Ziffern von 0 bis F ausgedrückt und am Anfang des numerischen Wertes mit “&H” gekennzeichnet.
Wenn der Wertebereich des Numerischen Wertes einer Konstante vom Typ Ganze Zahl im hexadezimalen Format ausgedrückt wird, beträgt er &H0~&HFFFFFFFF (bei 32 Bit-Wertebereich).

```
Beispiel: &H100
          &H3D5A
```

[2] Konstante vom Typ Real mit einfacher Genauigkeit

Dies ist eine Konstante vom Typ Real mit einer signifikanten Genauigkeit von bis zu 7 Stellen. Der Wertebereich ist -3,402823E + 38 ~ 3,402823E + 38.
Die folgenden 3 Notationsformate sind für eine Konstante vom Typ Real mit einfacher Genauigkeit verfügbar.

- Eine Zahl mit einer Abschlußziffer “!” vom Typ Real mit einfacher Genauigkeit
- Exponentialformat mit Verwendung von E
- Reale Zahl mit 7 oder weniger Ziffern ohne die Angabe oben

```
Beispiel: 1256,3
          35,78!
          -9,345E-06
```

[3] Konstante vom Typ Real mit doppelter Genauigkeit

Dies ist eine Konstante vom Typ Real mit einer signifikanten Genauigkeit von bis zu 15 Stellen.
Der Wertebereich ist -1,79769313486231D + 308 ~ 1,79769313486231D + 308.
Die folgenden 3 Notationsformate sind für eine Konstante vom Typ Real mit doppelter Genauigkeit verfügbar.

- Eine Zahl mit einer Abschlußziffer “#” vom Typ Real mit doppelter Genauigkeit
- Exponentialformat mit Verwendung von D
- Reale Zahl mit mehr als 7 Stellen oder weniger als 15 Stellen ohne die obige Angabe

```
Beispiel: 1256,325468
          35,78#
          -9,345D-06
```

Anmerkung: Aufgrund des internen Ausdrucks und der signifikanten Stellenzahl von Konstanten des Typs Real mit einfacher und doppelter Genauigkeit kann ein Fehler auftreten.

Beispiel
F1=10,0025
D1=10,0025
In diesem Fall wird die Zahl intern als 1,0002499580383…E+1 ausgedrückt. Mit dem Typ Real mit einfacher Genauigkeit eingegeben, dies wird zu 10,00250 durch Runden der nächsten Ziffer der signifikanten Stelle (8. Stelle).
Wenn dies mit dem Typ Real mit doppelter Genauigkeit eingegeben wird, wird es zu einem Wert von 10,002499580383… durch Abrunden der nächsten Ziffer der signifikanten Stelle (16. Stelle).

7.8.2 Konstante vom Typ Zeichenfolge

Die Konstante vom Typ Zeichenfolge ist eine Konstante, mit der eine Zeichenfolge ausgedrückt wird.
Eine Zeichenfolge wird ausgedrückt, indem sie in doppelte Anführungszeichen gesetzt wird.
Die Länge einer Zeichenfolge muß 247 Zeichen oder weniger betragen.

```
Beispiel: "PAC"
```

7.8.3 Konstante vom Typ Vektor

Eine Konstante vom Typ Vektor ist eine Konstante in Vektordarstellung mit den Komponenten X, Y und Z.
Jedes Element von X, Y und Z wird mit einem Typ Real einfacher Genauigkeit ausgedrückt.

```
Beispiel: Die Konstante vom Typ Vektor wird der
          Variablen vom Typ Vektor V1 zugewiesen.
          V[1] = (1,0,0)
```

7.8.4 Konstante vom Typ Stellung

[1] Konstante vom Typ Position

Eine Konstante vom Typ Position besteht aus der Position (X, Y, Z), Haltung (RX, RY, RZ) und Armstellung (FIG) und hat 7 Real-Parameter mit einfacher Genauigkeit. Die Position X, Y und Z wird mit der Einheit mm ausgedrückt, die Positionierung (RX, RY, RZ) in Einheiten des Rotationswinkels (°) und die Armstellung (FIG) mit Zahlen von 0 bis 31 (nicht definiert entspricht -1).

Anmerkung:	**Wenn ein Wert außerhalb des Bereiches von 0 bis 31 für die Armstellung (FIG) eingegeben wird, wird der eingegebene Wert zwar angezeigt, jedoch intern als nicht definiert (-1) behandelt.**

Die Haltung zeigt den Drehwinkel der X-, Y- und Z-Achse, und jede rechtsläufige Richtung wird als Vorwärtsdrehung betrachtet. Der Bereich ist auf -180° ~ +180° begrenzt. Da die Haltung von der Drehreihenfolge der Achsen abhängt, wird die Drehreihenfolge nur von den X-, Y- und Z-Achsen geregelt.

```
Beispiel: Die Konstante vom Typ Position wird der
          Variablen vom Typ Position P1 zugewiesen.
          P[1] = (100,200,300,10,20,30,0)
          'X=100,Y=200,Z=300,RX=10,RY=20,RZ=30,FIG=0
```

Die Elementkomponenten einer Positionsvariable können innerhalb eines Programmes ignoriert werden. Falls sie ignoriert werden, wird den Variablen Y, Z, RX, RY, RZ "0" und der Variablen FIG "-1 (nicht definiert)" zugewiesen. Sie können die Komponente X nicht ignorieren.

```
Beispiel:
        P[0] = (100,200,300)
und
        P[0] = (100,200,300,0,0,0,-1)
haben dieselbe Bedeutung.
```

[2] Konstante vom Typ Gelenk

Eine Konstante vom Typ Gelenk besteht aus Werten zu jeder Achse von der ersten bis zur sechsten. Jeder Achsenwert wird durch einen Typ Real einfacher Genauigkeit ausgedrückt. Die Einheit ist Grad.

```
Beispiel: Die Konstante vom Typ Gelenk wird der
          Variable vom Typ Achse J1 zugewiesen.
          J[0] = (10,20,30,40,50,60) '1~6 Achsen:
          10°,20°,30°,40°,50°,60°
```

Die Armkomponenten einer Variable vom Typ Gelenk können innerhalb eines Programmes ignoriert werden. Werden sie ignoriert, weist das System “0” zu. Die Komponente J1 können Sie jedoch nicht ignorieren.

```
Beispiel:
         J[0] = (10,20)
und
         J[0] = (10,20,0,0,0,0)
haben dieselbe Bedeutung.
```

[3] Konstante vom Typ Homogene Transformation

Eine Konstante vom Typ Homogene Transformation ist eine Konstante vom Typ Stellung, die durch die 3 Vektoren für Position, Orientierung und Annäherung und durch die Armstellung ausgedrückt wird.
Die 3 Vektoren werden jeweils durch 3 Real-Parameter einfacher Genauigkeit ausgedrückt. Die Konstante vom Typ Homogene Transformation wird daher durch 10 Real-Parameter einfacher Genauigkeit ausgedrückt.
Die Einheit für die 3 Positionsvektoren ist mm.

Anmerkung:	**Der Orientierungsvektor und der Anfahrvektor sind beides Einheitsvektoren der Größe 1 und stehen senkrecht aufeinander. Falls ein Vektor bezeichnet wird, der obige Bedingungen nicht erfüllt und mit einem Bewegungsbefehl verwendet wird, setzt das System die Priorität auf den Anfahrvektor und führt die Bewegung nach einem automatischen Kreuzen des Orientierungsvektors aus.**

```
Beispiel: Zuweisen der Konstante vom Typ Homogene
          Transformation zur Variablen vom Typ Homogene
          Transformation T1.
          T[1] = (10,20,30,1,0,0,0,1,0,4)
```

7.9 Ausdruck und Operator

Ein Ausdruck wird zur Rückgabe eines Wertes verwendet. Es gibt Ausdrücke, die einen unabhängigen Wert haben, und Ausdrücke, die aus mehreren, durch Operatoren verbundenen Elementen bestehen. Alle Datentypwerte in der PAC-Sprache werden mit Ausdrücken dargestellt.
Wenn die in den folgenden Abschnitten beschriebenen Operatoren verwendet werden, kann die Operation im Ausdruck ausgeführt werden.

7.9.1 Zuweisungs-Operator

Verwenden Sie den Zuweisungsoperator [=] in der Zuweisungsanweisung zum Zuordnen von Variablen. Ein Wert auf der rechten Seite des Zuweisungs-Operators wird der linken Seite zugewiesen.
Verwenden Sie in der Zuweisungsanweisung die folgenden Befehle entsprechend dem Datentyp.

LET, LETA, LETF, LETJ, LETO, LETP, LETR, LETRX, LETRY, LETRZ, LETX, LETY, LETZ

Da nur der LET Befehl beim Zuweisen einer Variable mit numerischem Wert ignoriert werden kann, haben die folgenden zwei Ausdrücke dieselbe Bedeutung.

```
Beispiel: LET I1 = 5          'Ordnet I1 den Wert 5 zu.
          I1 = 5              'Ordnet I1 den Wert 5 zu.
```

7.9.2 Arithmetischer Operator

Verwenden Sie die in Tabelle 1-3 dargestellten Operatoren bei arithmetischen Operationen.
Eine Operation wird mit der in der Tabelle dargestellten Priorität ausgeführt.

Tabelle 1-3 Arithmetische Operatoren

Arithmetischer Operator	Beschreibung der Operation	Priorität der Operation
^	Exponential- (Exponentierungs-) Operation	Hoch ↕
-	Negatives Vorzeichen	↕
*, /	Multiplikation, Division	↕
MOD	Rest	↕
+, -	Addition, Subtraktion	Niedrig

Anmerkung (1): Bei der Division durch 0 tritt ein Fehler in der Ausführung auf.

Anmerkung (2): Wenn ein Overflow bei der Addition oder Multiplikation ganzer Zahlen auftritt, werden die Overflow-Zahlen ignoriert. Beachten Sie, daß hierbei kein Fehler auftritt.

Beispiel: I1 = 2147483647 + 1 '2147483647 ist der maximal erlaubte Wert von Variablen des Typs Ganze Zahl.
Wenn obiger Ausdruck ausgeführt wird, ist das Ergebnis -2147483648.
-2147483648 ist der minimal erlaubte Wert von Variablen des Typs Ganze Zahl.

Anmerkung (3):	**Ein Fehler tritt auf, wenn ein Overflow bei der Addition oder Multiplikation von Variablen des Real-Typs auftritt, oder wenn versucht wird, einen Wert zuzuweisen, der durch die ganzzahlige Variable des Real-Typs nicht aufgezeichnet werden kann.**
Anmerkung (4):	**Bei der Subtraktion ganzer Zahlen wird das Ergebnis auf die maximale ganze Zahl gerundet, die das Ergebnis nicht übersteigt.**
Anmerkung (5):	**Das Vorzeichen für den Rest einer ganzen Zahl wird entsprechend der folgenden Tabelle bestimmt.**

Arithmetische Operatoren

Divisor / **Dividend**	**+**	**0**	**-**
+	+	Fehler	+
0	0	Fehler	0
-	-	Fehler	+

7.9.3 Relationaler Operator

Ein relationaler Operator wird verwendet, um zwei numerische Werte zu vergleichen. Das Ergebnis erhalten Sie als Booleschen Wert (Wahr “1” und Falsch “0”); es wird beispielsweise in einem bedingten Ausdruck in der Flußsteuerungsanweisung verwendet.

Relationale Operatoren

Relationaler Operator	Beschreibung der Operation
=	Gleich
=.	Annähernd gleich (Näherungsvergleich)
<>	Nicht gleich
<	Kleiner
>	Größer
<=	Kleiner oder gleich
>=	Größer oder gleich

Bemerkung:	**Zur Genauigkeit des Operators für den Näherungsvergleich (=.), siehe “Näherungsvergleich Genauigkeit” der Programmtabelle in [Projektsatz] von [Datei] im [PAC Manager].**

7.9.4 Logischer Operator

Der logische Operator führt Bit-Operationen aus.
Eine Operation wird ausgeführt, nachdem Werte, die nicht vom Typ Ganze Zahl sind, in ganze Zahlen umgewandelt wurden.

Anmerkung:	**Wenn ein Wert außerhalb des erlaubten Wertebereichs als Typ Ganze Zahl bezeichnet wird, tritt ein Fehler bei der Ausführung auf.**

Logische Operatoren

Logischer Operator	Beschreibung der Operation
NOT	Negation
AND	Logisches Produkt
OR	Logische Addition
XOR	Exklusive logische Addition

```
Beispiel: Bit-Operation
          I1 = &B1100  XOR  &B0101
          Das Ergebnis dieses Beispiels ist &B1001.
```

7.9.5 Zeichenfolge-Operator

Eine Zeichenfolge kann mit einem Zeichenfolge-Operator “+” verknüpft werden.

```
Beispiel: A$ = “ABC” + “DEF” 'A$ wird zu “ABCDEF”.
```

7.9.6 Vektor-Operation

Verwenden Sie bei einer Vektor-Operation die in der folgenden Tabelle angegebenen Operatoren.
Eine Operation wird entsprechend der in der Tabelle angegebenen Priorität augeführt.

Vektor-Operatoren

Vektor-Operator	Beschreibung der Operation	Priorität der Operation
., *, x	Inneres Produkt, Skalar, Äußeres Produkt	Hoch
+, -	Addition, Subtraktion	Niedrig

```
Beispiel 1:  Berechnung des inneren Produkts von V1 und V2.
             F1 = V1 . V2

Beispiel 2:  Berechnung von zwei Mal V1.
             V2 = V1 * 2

Beispiel 3:  Berechnung des äußeren Produkts von V1 und V2.
             V3 = V1 x V2
```

Anmerkung: Da das Vorzeichen des äußeren Produktoperators der Buchstabe "x" ist, stellen Sie sicher, daß Sie ein Leerzeichen vor und nach dem Vorzeichen einfügen.

7.9.7 Positions-Operation

Dies ist die Operation, die für Daten vom Typ Position ausgeführt wird.
Für die Operation von einer parallelen Bewegungs- und Rotationsabweichung kann nur "+" verwendet werden.

Beispiel 1: Berechnung von Punkt P1, der (dx, dy, dz) von der Abweichung des Referenzpunktes P0 im Roboter-Koordinatensystem entfernt ist.

P1 = P0 + (dx, dy, dz) oder P1 = P0 + (dx, dy, dz, 0, 0, 0)

Beispiel 2: Berechnung von Punkt P2, der (df, dg, dh) von der aktuellen Position im Werkzeugkoordinatensystem entfernt ist.
P2 = * + (df, dg, dh)H oder P2 = * + (df, dg, dh, 0, 0, 0)H

Beispiel 3: Berechnung von Punkt P3, der abgerufen wird, so daß die Armstellung am Referenzpunkt um Rx (Grad) um die X-Achse, um Ry (Grad) um die Y-Achse und um Rz (Grad) um die Z-Achse im Roboter-Koordinatensystem gedreht wird.
P3 = P0 + (0, 0, 0, Rx, Ry, Rz)

Beispiel 4: Berechnung von Punkt P4, der (df, dg, dh) von der aktuellen Position im Werkzeugkoordinatensystem entfernt ist und um Rx (Grad) um den Normalvektor, um Ry (Grad) um den Orientierungsvektor und um Rz (Grad) um den Anfahrvektor gedreht wird.
P4 = * + (df, dg, dh, Rx, Ry, Rz)H

Anmerkung: Eine Variable vom Typ Position kann nicht für Abweichungen verwendet werden.

7.9.8 Gelenk-Operation

Dies ist die Operation, die für Daten vom Typ Gelenk ausgeführt wird.
Für Operationen von einer parallelen Abweichung kann nur “+” verwendet werden.

```
Beispiel:  Berechnung von Punkt J1, der (d1, d2, d3, d4,
           d5, d6) von der Abweichung des Referenzpunktes
           J0 im Roboter-Koordinatensystem entfernt ist.
           J1 = J0 + (d1, d2, d3, d4, d5, d6)
```

Anmerkung: Eine Variable vom Typ Gelenk kann nicht für Abweichungen verwendet werden.

7.9.9 Homogene Transformations-Array-Operation

Dies ist die Operation, die für Homogene Transformations-Arrays ausgeführt wird.
Verwenden Sie für das Produkt aus einem Homogenen Transformations-Array den Operator [*].

```
Beispiel:  Berechnung von T3 als Produkt von T1 und T2.
           T3 = T1 * T2
Die Armstellung von T3 wird dabei auf die von T2 eingestellt.
```

7.10 Einheiten für die PAC-Sprache

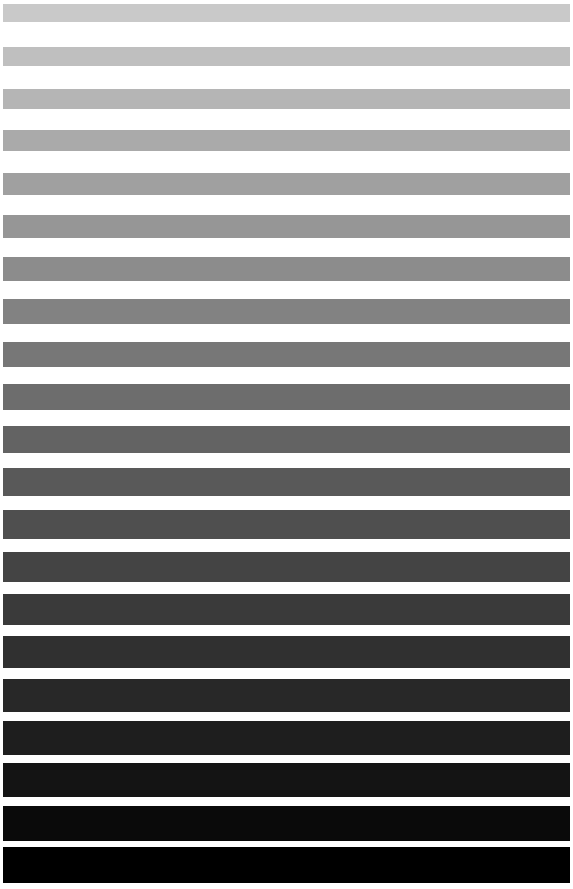
Die folgende Tabelle zeigt die Einheiten für den Ausdruck der physikalischen Werte in der PAC-Sprache.

Einheiten für den Ausdruck der physikalischen Werte in der PAC-Sprache

Physikalischer Wert	Einheit
Länge	Millimeter (mm)
Winkel	Grad (DEGREE)
Zeit	Millisekunden (msec.)
Geschwindigkeit	% (Prozentsatz der Maximalgeschwindigkeit)
Beschleunigung	% (Prozentsatz der Maximalbeschleunigung)
Verzögerung	% (Prozentsatz der Maximalverzögerung)

Kapitel 8

Syntax der PAC-Sprache



Dieses Kapitel enthält eine Erläuterung der Regeln zum Schreiben von Programmen mit der PAC-Sprache.

8.1 Anweisung und Zeile

Ein Programm in der PAC-Sprache besteht aus mehreren Zeilen.
Eine Anweisung kann in einer beliebigen Zeile stehen.
Die Länge einer Zeile kann bis zu 255 Byte betragen.
Eine Anweisung ist die kleinste Einheit zur Beschreibung eines Prozesses in der PAC-Sprache und umfaßt einen Befehl.
Ein Befehl umfaßt einen Befehlsnamen und die Informationen (Parameter) für den Befehl.

8.2 Programmname und Deklaration

Deklarieren Sie die für die Programmausführung erforderlichen Elemente, wie Programmname und Variablen, vor der Ausführung.
Besonders wenn ein Programmname deklariert wird, muß er in der ersten gültigen Programmzeile deklariert werden. Diese Anweisung wird PROGRAM-Deklarationsanweisung genannt.
Die PROGRAM-Deklarationsanweisung kann ignoriert werden. Wenn sie ignoriert wird, wird der Programm-Dateiname als Programmname verwendet. Wenn zum Beispiel der Programm-Dateiname PRO1.PAC ist und kein anderer Name mit einer PROGRAM-Deklarationsanweisung angegeben wird, wird der Programmname auf PRO1 gesetzt.
Wenn die PROGRAM-Deklarationsanweisung ignoriert wird, kann bei einem Programmaufruf kein Argument übergeben werden. Darüber hinaus ist ein editiertes Programm schwer zu verstehen. Ignorieren Sie daher keine PROGRAM-Deklarationsanweisung ohne besonderen Grund.
Weitere Informationen zum Aufrufen eines Programmes siehe Abschnitt 2.1 “Aufrufen eines Programmes und eines Unterprogrammes” in Kapitel 2 dieses Handbuches.

Anmerkung (1):	**Programme, die vom Programmiergerät aus ausgeführt werden können, können keine Argumente beinhalten. Programme, die mit einem externen Gerät aufgerufen werden können, müssen das Programmnamensformat “PRO <Nummer>” verwenden. Programme, die nicht vom Programmiergerät oder von einem externen Gerät ausgeführt werden können, können von anderen Programmen aufgerufen werden.**
Anmerkung (2):	**Wenn ein Programm mit einem Programmnamen im Format “PRO<Nummer>” aufgerufen wird, kann kein Argument übergeben werden.**
Anmerkung (3):	**Wenn “PRO” für die ersten 3 Buchstaben eines Programmnamens verwendet wird, stellen Sie sicher, daß Sie auch eine Nummer hinzufügen.** **Da <Nummer> als Zahl interpretiert wird, werden PRO01 oder PRO001 wie PRO1 verarbeitet.**

8.3 Kennung

Eine Kennung kann verwendet werden, um auf ein Verzweigungsziel und die Position einer Anweisung in einem Programm hinzuweisen.
Die folgenden Regeln gelten für die Verwendung einer Kennung.

- Der Name einer Kennung beginnt mit einem Stern (*).
- Das zweite Zeichen im Namen einer Kennung muß ein Buchstabe sein.
- Für das dritte und die weiteren Zeichen in einem Kennungsnamen kann eine beliebige Kombination von Buchstaben und Ziffern verwendet werden.
- Das letzte Zeichen im Namen einer Kennung muß ein Doppelpunkt (:) sein.
- Ein reserviertes Wort kann nicht als Kennungsname verwendet werden.
- Eine Kennung, auf die verwiesen werden soll, muß am Anfang der Zeile stehen.
- Wenn der Name einer Kennung, auf die verwiesen werden soll, mehrfach angegeben ist, tritt ein Fehler auf.
- Der Bereich, innerhalb dessen auf eine Kennung verwiesen werden kann, ist auf das Programm begrenzt, in dem die Kennung steht.

```
PROGRAM WITH_LABEL
    IF IO138=1  THEN*ACTION:  ELSE  *NOACTION:
    *ACTION:        SPEED 100
                    MOVE P, P1
                    DELAY 200
                    MOVE P, P2
    *NOACTION:      DELAY 200
    END IF
END
```

Programmbeispiel mit Kennung

8.4 Zeichensatz

Die Zeichen, die in der PAC-Sprache verwendet werden können, sind Buchstaben aus dem Alphabet, Ziffern und Symbole. Bei Buchstaben aus dem Alphabet wird nicht zwischen Groß- und Kleinbuchstaben unterschieden.
Die folgenden Symbole werden zusätzlich zu den arithmetischen Operatoren (+, -, *, /) verwendet.

- Komma (,)
  Trennt die einzelnen Parameter.
- Semikolon (;)
  Trennt die einzelnen Parameter in einem Argument eines Befehls.
- Einfaches Anführungszeichen (')
  Dient als Ersatz für den REM Befehl.
- Doppeltes Anführungszeichen (")
  Bezeichnet Daten vom Typ Zeichenfolge, indem es vor und hinter einer beliebigen Zeichenfolge steht.
- Sternchen (*)
  Steht vor einem Kennungsnamen.
  Dies zeigt die aktuelle Position des Schnittstellenkoordinatensystems an.
- Leerzeichen
  Wird vor und hinter einem Anweisungsnamen benötigt.

Anmerkung:	**Sie können die chinesischen Zeichencodes (Umschaltung-JIS) in Kommentaren der REM-Zeilen und in Zeichenfolgekonstanten verwenden.**

8.5 Reserviertes Wort

Ein Wort, das bei der Verarbeitung in der PAC-Sprache eine feste Verwendung hat, wie ein Befehlsname oder Operator, wird Reserviertes Wort genannt.
Zur Verwendung reservierter Wörter müssen Sonderzeichen (Leerzeichen, ", # und :) vor und/oder hinter jedem Wort eingefügt werden, um die Sonderzeichen zu kennzeichnen.
Reservierte Worte in der PAC-Sprache sind in der "Liste der Reservierten Worte" in Anhang-5 aufgeführt.
Ein Reserviertes Wort kann nicht selbst als Variablenname oder Kennung verwendet werden, aber ein Name, der ein Reserviertes Wort enthält, kann verwendet werden. "HOME" ist beispielsweise ein reserviertes Wort. Sie können jedoch einen Namen wie "HOME1" für eine Variable oder eine Kennung verwenden.

8.6 Deklarationsanweisung

Deklarieren Sie Definitionen in einer Deklarationsanweisung, bevor Sie Variablen, Konstanten, Funktionen usw. verwenden, die bezeichnete Namen oder Typen in einem Programm benötigen.
Die folgenden 3 Grundtypen von Anweisungen sind Deklarationsanweisungen.

- Typdeklaration : deklariert den Typ einer Variablen auf Konstante.
- Funktions-/Programmdeklaration : deklariert eine Funktion oder ein Programm.
- Benutzerkoordinatensystem-Deklaration: deklariert ein Benutzerkoordinatensystem.

8.6.1 Typdeklaration

[1] Typdeklarationszeichen (Abschlußziffer)

Feste Abschlußziffern können für die Deklaration eines Variablentyps verwendet werden. Die Abschlußziffern sind die folgenden.

Typdeklarationszeichen (Abschlußziffern)

Typ	Abschlußziffer	Beispiel
Abschlußziffer für Typ Ganze Zahl	%	A%
Abschlußziffer für Typ Real mit einfacher Genauigkeit	!	A!
Abschlußziffer für Typ Real mit doppelter Genauigkeit	#	A#
Abschlußziffer für Zeichenfolge	$	A$

Anmerkung: **Die Variablentypen in einem Programm werden wie der zuerst deklarierte Typ verarbeitet. Selbst wenn die Abschlußziffer eines anderen als des deklarierten Typs im Programm verwendet wird, wird der neue Typ nicht identifiziert.**
A, A%, A!, A# und A$ werden als identische Variablen erkannt und als der erste verwendete Typ registriert. Daher tritt ein Fehler auf, wenn die Abschlußziffer eines anderen als des zuerst verwendeten Typs verwendet wird.

[2] Typdeklarationsanweisung

Die folgenden Typdeklarationsbefehle können zur Deklaration des Variablentyps verwendet werden.

Typdeklarationsanweisungen

Typ	Befehl	Beispiel
Ganze Zahl	DEFINT	DEFINT AA, AB
Real mit einfacher Genauigkeit	DEFSNG	DEFSNG BA, BB
Real mit doppelter Genauigkeit	DEFDBL	DEFDBL CA, CB
Zeichenfolge	DEFSTR	DEFSTR DA, DB
Vektor	DEFVEC	DEFVEC EA, EB
Position	DEFPOS	DEFPOS FA, FB
Gelenk	DEFJNT	DEFJNT GA, GB
Homogene Transformation	DEFTRN	DEFTRN HA, HB

In diesen Befehlen kann eine Variable zusammen mit der Typdeklaration initialisiert werden.
In einer Deklarationsanweisung, die diese Befehle verwendet, kann keine Abschlußziffer einem Variablennamen hinzugefügt werden.

Beispiel:

```
DEFINT AA = 1          'Weist 1 zu AA als Typ Ganze Zahl zu.
DEFSNG BB (10)         'Setzt BB auf einen Typ Real mit 'einfacher
                       Genauigkeit, dessen 'Elementeanzahl 10
                       ist.
```

Ungültiges Beispiel:

```
DEFINT AB%             'Ein Fehler tritt auf, da eine
                       'Abschlußziffer verwendet wurde.
```

[3] Array-Deklaration

Dies ist eine Deklarationsanweisung für ein Array. Ein Array kann für alle Typen außer E/A-Variablen durch Hinzufügen einer Abschlußziffer zu einem Variablennamen erstellt werden. Ein Array kann jedoch nicht zusammen mit der Deklaration initialisiert werden.
Der Index eines Arrays muß 0 oder mehr sein. Das Array kann bis zu 3 Dimensionen haben.
Die obere Begrenzung für die Gesamtzahl von Elementen in einem Array ist 32767.

Array-Deklaration

Typ	Befehl	Beispiel
Array-Deklaration	DIM	DIM AA (10,10)

Beispiel einer Array-Deklaration:

```
DIM CC (3,3,3)  AS INTEGER 'Setzt CC auf ein 3-dimensionales Array
                           'des Typs Ganze Zahl.
```

Das obige Beispiel kann wie folgt unter Verwendung eines DEFINT Befehls ausgedrückt werden.

```
DEFINT CC (3,3,3)          'Setzt CC auf ein 3-dimensionales Array
                           'des Typs Ganze Zahl.
```

Anmerkung: Unterschied zwischen DEF???-Anweisung und DIM-Anweisung

Eine DIM-Anweisung kann für mehr als nur ein Array verwendet werden.

Zum Beispiel kann für

DEFINT CC,

geschrieben werden:

DIM CC AS INTEGER

Da sie nicht gleichzeitig mit der Typdeklaration initialisiert werden kann, ist eine DIM-Anweisung universeller verwendbar als eine DEF-Anweisung. Zum Beispiel kann der folgende Ausdruck nicht mit einer DIM-Anweisung erstellt werden.

DEFINT CC = 1 'Deklariert CC als Typ Ganze Zahl und setzt den Anfangswert auf 1.

Wenn es nicht zusammen mit der Deklaration initialisiert ist, können alle Typen von Arrays in derselben Weise verarbeitet werden. Daher wird die Verwendung einer DIM-Anweisung empfohlen.

[4] E/A-Variablen-Deklaration

Ein Variablenname entspricht einem bestimmten E/A-Anschluß.

E/A-Variablen-Deklaration

Typ	Befehl	Beispiel
E/A-Variablen-Deklaration	DEFIO	DEFIO PORT = BYTE,104

8.6.2 Funktions-/Programmdeklaration

Folgende Befehle werden zur Deklaration einer Funktion oder eines Programmnamens verwendet.

Funktions/Programmdeklaration

Typ	Befehl	Beispiel
Funktionsdeklaration	DEF FN	DEF FN AREA (R) = PI * R * R
Programmdeklaration	PROGRAM	PROGRAM PRO1

8.6.3 Benutzerkoordinatensystem-Deklaration

Folgende Befehle werden zur Deklaration eines Benutzerkoordinatensystems verwendet.

Benutzerkoordinatensystem-Deklaration

Typ	Befehl	Beispiel
Sicherheitsbereichsdeklaration	AREA	AREA 1, P0, V0, 15, 6
Werkzeugkoordinatensystemdeklaration	TOOL	TOOL 1, P0
Arbeitskoordinatensystemdeklaration	WORK	WORK 1, P0

8.7 Zuweisungsanweisung

Eine Zuweisungsanweisung setzt eine beliebige Variable auf einen Wert.
Es gibt 4 Zuweisungsanweisungen für numerische Werte: Zuweisungsanweisung, Zuweisungsanweisung für Zeichenfolge, Zuweisungsanweisung für Vektor und Zuweisungsanweisung für Stellung.

Anmerkung:	**Der LET Befehl kann bei allen Zuweisungsanweisungen ignoriert werden.**

8.7.1 Zuweisungsanweisung für numerischen Wert

Eine Zuweisungsanweisung für numerischen Wert weist einer Variablen mit numerischem Wert einen Wert zu.

Beispiel:
```
LET I[1] = 10      'Weist 10 zu I[1] zu.
D[2] = 3,14        'Weist 3,14 zu D[2] zu.
```

8.7.2 Zuweisungsanweisung für Zeichenfolge

Eine Zuweisungsanweisung für Zeichenfolge weist eine Variable vom Typ Zeichenfolge zu.

Beispiel:
```
S[2] = "DENSO"  'Weist "DENSO" zu S[2] zu.
LET SS$ = S[2]  'Weist den Wert von S[2] zu SS$ zu.
```

8.7.3 Zuweisungsanweisung für Vektor

Eine Zuweisungsanweisung für Vektor weist eine Variable des Typs Vektor zu. Die folgenden 4 Befehle können verwendet werden.

Zuweisungsanweisungen für Vektor

Typ	Befehl	Beispiel
Zuweisung zu Vektor	LET	LET V[2] = (1,2,3)
Zuweisung zur X-Komponente	LETX	LETX V[2] = F1
Zuweisung zur Y-Komponente	LETY	LETY V[2] = F1
Zuweisung zur Z-Komponente	LETZ	LETZ V[2] = F1

8.7.4 Zuweisungsanweisung für Stellung

Es gibt 4 Typen von Zuweisungsanweisungen für Stellung: Zuweisungsanweisung für Position, Zuweisungsanweisung für Gelenk, Zuweisungsanweisung für Homogene Transformation und Zuweisungsanweisung für Ausgangsposition.

[1] Zuweisungsanweisung für Ausgangsposition

Eine Zuweisungsanweisung für Position weist eine Variable des Typs Position zu. Die folgenden Befehle können verwendet werden.

Zuweisungsanweisungen für Position

Typ	Befehl	Beispiel
Zuweisung zu Position	LET	LET P[3] = (10, 10, 10, 0, 0, 0)
Zuweisung zum Positionsvektor	LETP	LETP P3 = V1
Zuweisung zum Drehungsvektor	LETR	LETR P3 = V2
Zuweisung zur X-Komponente	LETX	LETX P[3] = F1
Zuweisung zur Y-Komponente	LETY	LETY P[3] = F1
Zuweisung zur Z-Komponente	LETZ	LETZ P[3] = F1
Zuweisung zur RX-Komponente	LETRX	LETRX P[3] = F1
Zuweisung zur RY-Komponente	LETRY	LETRY P[3] = F1
Zuweisung zur RZ-Komponente	LETRZ	LETRZ P[3] = F1
Zuweisung zur Armstellungskomponente	LETF	LETF P[3] = I1

[2] Zuweisungsanweisung für Gelenk

Eine Zuweisungsanweisung für Gelenk weist eine Variable des Typs Gelenk zu. Die folgenden Befehle können verwendet werden.

Zuweisungsanweisungen für Gelenk

Typ	Befehl	Beispiel
Zuweisung zu Gelenk	LET	LET J[4] = (1, 2, 3, 4, 5, 6) 6 Achsen
Zuweisung zur Achsenkomponente	LETJ	LETJ 1, J[4] = F1

[3] Zuweisungsanweisung für Homogene Transformation

Eine Zuweisungsanweisung für Homogene Transformation weist einer Variablen vom Typ Homogene Transformation einen Wert zu. Die folgenden Befehle können verwendet werden.

Zuweisungsanweisungen für Homogene Transformation

Typ	Befehl	Beispiel
Zuweisungsanweisung zu Homogene Transformation	LET	LET T[x]= (1, 2, 3, 4, 5, 6, 7, 8, 9, 4)
Zuweisung zum Positionsvektor	LETP	LETP T[x] = V1
Zuweisung zum Orientierungsvektor	LETO	LETO T[x] = V[1]
Zuweisung zum Anfahrvektor	LETA	LETA T[x] = V2
Zuweisung zur Armstellungskomponente	LETF	LETF T[x] = I1

[4] Zuweisungsanweisung zur Ausgangsposition

Eine Zuweisungsanweisung für Ausgangsposition setzt eine beliebige Position als Ausgangsposition.

Zuweisungsanweisungen für Ausgangsposition

Typ	Befehl	Beispiel
Zuweisungsanweisung für Ausgangsposition	HOME	HOME * oder HOME CURPOS Setzt die aktuelle Position als Ausgangsposition.

8.8 Flußsteuerungsanweisung

Verwenden Sie eine Flußsteuerungsanweisung zur Steuerung der Ausführungsreihenfolge jeder Anweisung in einem Programm.
Verwenden Sie eine Kennung für eine Programmflußsteuerungsanweisung zur Angabe ihrer Position in einem Programm.
Die Flußsteuerung kann grob in 5 Anweisungen eingeteilt werden: Unbedingte Verzweigung, Bedingte Verzweigung, Auswahl, Wiederholung und Aufruf eines definierten Prozesses.

8.8.1 Unbedingte Verzweigung

Verwenden Sie dies, um die Programmausführung an eine beliebige Position zu übertragen. Geben Sie nach dem GOTO-Befehl eine Kennung für die Anweisung ein, die Sie als nächste ausführen möchten.

Beispiel: GOTO *LABEL1

8.8.2 Bedingte Verzweigung

Wenn eine IF ~ THEN ~ ELSE-Anweisung oder IF ~ ENDIF-Anweisung verwendet wird, bestimmt ein Verzweigungsziel, ob die angegebene Bedingung erfüllt ist.
Wenn der Wert des relationalen Ausdrucks hinter IF wahr ist (TRUE (1)), wird der Prozeß hinter THEN ausgeführt. Andernfalls wird der Prozeß hinter ELSE ausgeführt.

8.8.3 Auswahl

Je nach Wert des angegebenen Ausdrucks wird der auszuführende Prozeß gewählt. Es gibt 3 Befehle.

Bei SELECT CASE-Anweisungen setzen Sie einen arithmetischen Ausdruck hinter CASE auf der SELECT Zeile. Der Prozeß wird von der Zeile von CASE, deren Wert diesem arithmetischen Ausdruck genügt, bis zur nächsten CASE-Zeile oder zur END SELECT-Zeile ausgeführt. Falls diese CASE-Zeilen dem Ausdruck nicht genügen, wird der Prozeß von der Zeile hinter CASE ELSE bis zur END SELECT-Zeile ausgeführt.

Geben Sie in ON ~ GOSUB-Anweisungen einen arithmetischen Ausdruck nach ON ein. Der Prozeß wird fortgesetzt bis zu einem Unterprogramm entsprechend dem Wert des arithmetischen Ausdrucks.

Geben Sie in ON ~ GOTO-Anweisungen einen arithmetischen Ausdruck nach ON ein. Der Prozeß wird fortgesetzt bis zu einer Kennung entsprechend dem Wert des arithmetischen Ausdrucks.

8.8.4 Wiederholung

Dies steuert die Wiederholung entsprechend einer angegebenen Bedingung. Es gibt hierfür 4 Befehle.

Bei FOR ~ NEXT-Anweisungen setzen Sie eine Wiederholungsbedingung hinter die FOR-Zeile. Der Prozeß zwischen FOR und der entsprechenden NEXT-Zeile wird wiederholt, bis diese Bedingung erfüllt wird.

Bei DO ~ LOOP-Anweisungen setzen Sie einen relationalen Ausdruck hinter WHILE oder UNTIL. Während dies erfüllt ist (bei WHILE) oder bis dieses erfüllt ist (bei UNTIL), wird der Prozeß zwischen der DO-Zeile und der LOOP-Zeile wiederholt.

Bei WHILE ~ WEND-Anweisungen setzen Sie einen relationalen Ausdruck hinter WHILE. Während dies erfüllt ist, wird der Prozeß zwischen der WHILE-Zeile und der WEND-Zeile wiederholt. Dies hat dieselbe Funktion wie die DO WHILE ~ LOOP-Anweisung.

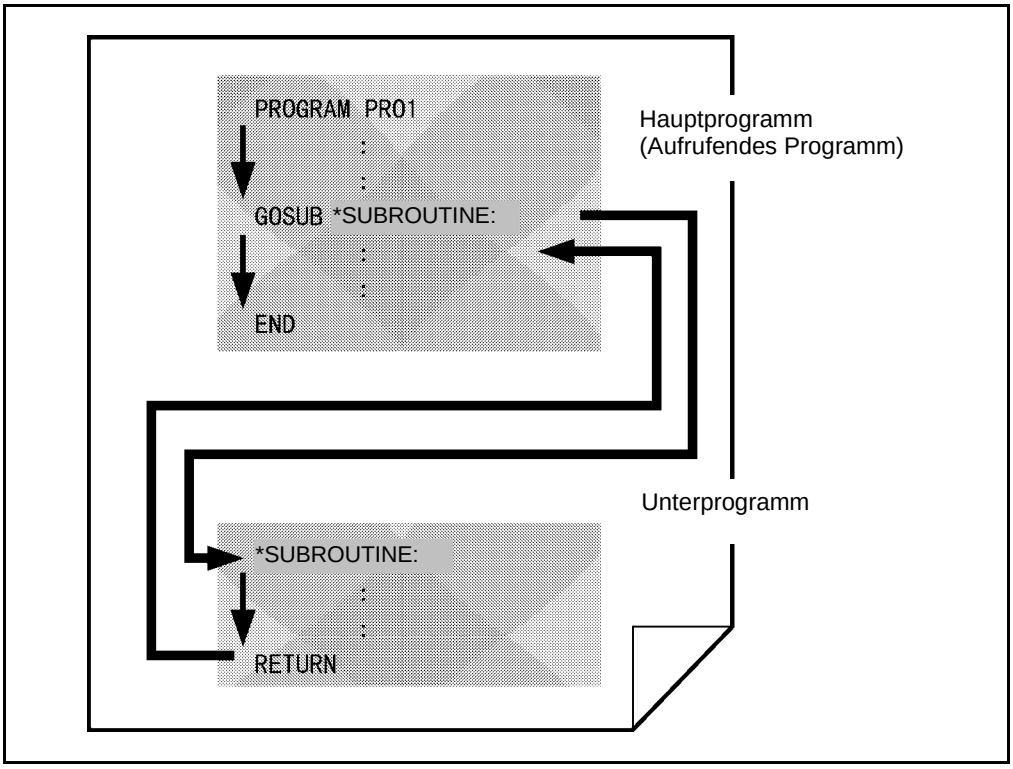
Bei REPEAT ~ UNTIL-Anweisungen setzen Sie einen relationalen Ausdruck hinter UNTIL. Bis dieser erfüllt ist, wird der Prozeß zwischen der REPEAT-Zeile und der UNTIL-Zeile wiederholt. Dies hat dieselbe Funktion wie die DO ~ LOOP UNTIL-Anweisung.

8.8.5 Aufruf eines definierten Prozesses

Wenn ein Teil eines Programmes, das eine bestimmte Bewegung wiederholt, abgetrennt wird, kann dieser Teil bei Bedarf separat aufgerufen werden. Dies nennt man Aufruf eines definierten Prozesses.
Um einen solchen Prozeß aufzurufen, gibt es zwei Möglichkeiten: Unterprogrammaufruf und Programmaufruf.

[1] Unterprogramm

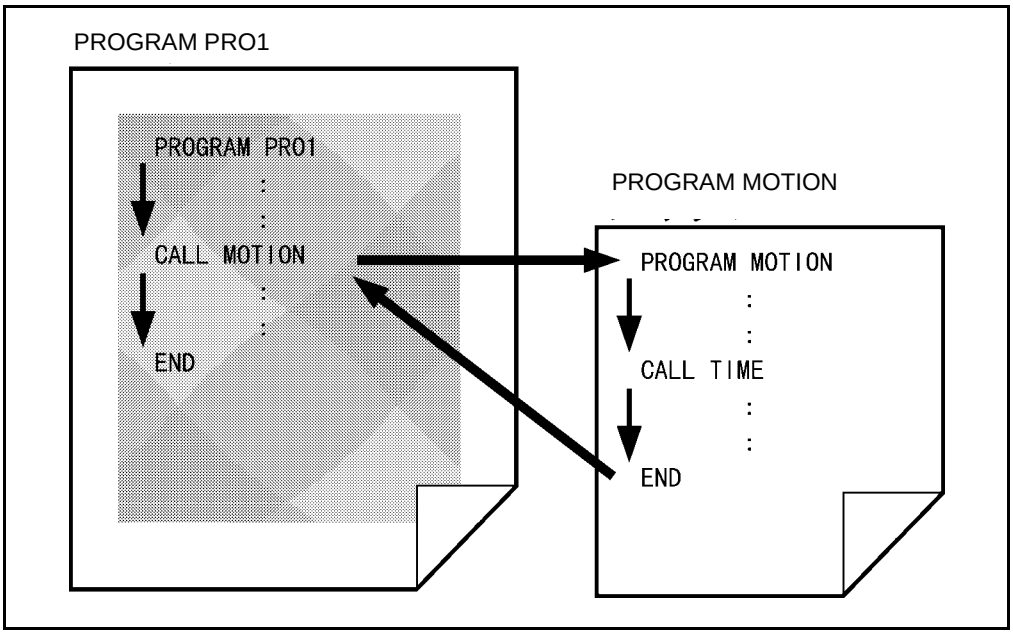
Wenn ein Unterprogramm aufgerufen werden soll, geben Sie ein Ziel mit einer Kennung in einer GOSUB-Anweisung an.
Das Unterprogramm muß in derselben Datei geschrieben sein wie das aufrufende Programm. Die letzte Zeile der Unterprogrammes muß die Anweisung RETURN enthalten. Nachdem eine Reihe von Prozessen durchgeführt sind, wird die RETURN-Anweisung ausgeführt und die Steuerung geht zurück zu der nächsten Zeile im Programm, das das Unterprogramm aufgerufen hat.



Struktur zum Aufruf eines Unterprogrammes

[2] Programm

Wenn ein Programm aufgerufen werden soll, geben Sie den Programmnamen in einer CALL-Anweisung an und führen Sie diese aus. Ein rekursiver Aufruf kann ebenfalls ausgeführt werden.
Informationen hierzu, siehe Abschnitt 2.1 “Aufrufen eines Programmes und eines Unterprogrammes” in Kapitel 2 dieses Handbuches.



Struktur zum Aufruf eines Programmes

8.9 Robotersteuerungsanweisung

Robotersteuerungsanweisungen können grob unterteilt werden in Steuerungsanweisungen für Bewegung, Roboterarmsteuerung, Steuerungsanweisungen für Stop, Steuerungsanweisungen für Geschwindigkeit, Steuerungsanweisungen für Zeit und Anweisung für Koordinatentransformation.

8.9.1 Steuerungsanweisung für Bewegung

Eine Anweisung zur Steuerung der Roboterbewegung wird Steuerungsanweisung für Bewegung genannt. Es gibt Roboterarmsteuerungsanweisungen, Handsteuerungsanweisungen und Motorsteuerungsanweisungen.

[1] Roboterarmsteuerung

Es gibt vier Typen von Steuerungsanweisungen für Roboterarmbewegungen: Anweisungen für Typ P, Typ J, Typ T und für andere Daten als P/J/T.

Zielstellung: Daten vom Typ P, J, T

Bewegungssteuerungsbefehle (Die Zielstellung sind Daten vom Typ P, J und T.)

Bewegungsart	Befehl
Allgemeine Bewegung	MOVE
Annäherungsbewegung	APPROACH
Wegfahrbewegung	DEPART

Zielstellung: Andere Daten als Typ P, J, T

Bewegungssteuerungsbefehle (Die Zielstellung sind andere Daten als Typ P, J und T.)

Bewegungsart	Befehl
Drehbewegung	ROTATE
Drehbewegung in Annäherungsrichtung	ROTATEH
Translationsbewegung	DRAW
Relative Bewegung jeder Achse	DRIVE
Absolute Bewegung jeder Achse	DRIVEA
Bewegung zur Ausgangsposition	GOHOME

8.9.2 Steuerungsanweisung für Stop

Die Steuerungsanweisung für Stop stoppt oder beendet die Programmausführung.

Steuerbefehle für Stop

Bewegungsart	Befehl	Anmerkung
Ausführung zeitweise stoppen	HOLD	Stop nach einem Schritt.
Ausführung stoppen	HALT	Stoppt sofort. SCHIEBT die eigene Task auf.
Ausführung beenden	STOP	LÖSCHT die eigene Task.
Bewegungsabbruch	INTERRUPT	Stoppt sofort wegen einer Unterbrechung.

8.9.3 Steuerungsanweisung für Geschwindigkeit

Eine Steuerungsanweisung für Geschwindigkeit kann verwendet werden, um die Bewegungsgeschwindigkeit und die Beschleunigung und Verzögerung des Armes einzustellen.

Steuerbefehle für Geschwindigkeit

Bewegungsart	Befehl
Lineargeschwindigkeit	SPEED
Punkt-zu-Punkt-Geschwindigkeit	JSPEED
Linearbeschleunigung	ACCEL
Punkt-zu-Punkt-Beschleunigung/Verzögerung	JACCEL
Linearverzögerung	DECEL
Punkt-zu-Punkt-Verzögerung	JDECEL

8.9.4 Steuerungsanweisung für Zeit

Eine Steuerungsanweisung für Zeit führt die Bewegungssteuerung eines Roboters entsprechend der abgelaufenen Zeit aus.

Steuerbefehle für Zeit

Bewegungsart	Befehl
Stop zur vorgegebenen Zeit	DELAY
Bedingter Stop	WAIT

8.9.5 Anweisung für Koordinatentransformation

Eine Anweisung für Koordinatentransformation ändert das Koordinatensystem.

Befehle für Koordinatentransformation

Bewegungsart	Befehl
Änderung des Werkzeugkoordinatensystems	CHANGETOOL
Änderung des Arbeitskoordinatensystems	CHANGEWORK
Aktiviert den Bereich für die Interferenzprüfung	SETAREA
Deaktiviert den Bereich für die Interferenzprüfung	RESETAREA

8.10 Steuerungsanweisung für die Ein/Ausgabe

Es gibt 3 Typen von Steuerungsanweisungen für die Ein-/Ausgabe: DI/DO-Steuerungsanweisung, RS232C-Steuerungsanweisung und Steuerungsanweisung für Programmiergerät.

8.10.1 DI/DO-Steuerungsanweisung

Eine DI/DO-Steuerungsanweisung steuert die Ein-/Ausgabe am E/A-Anschluß.

Steuerbefehle für DI/DO

Bewegungsart	Befehl
Daten lesen	IN
Daten schreiben	OUT
E/A-Anschluß ON	SET
E/A-Anschluß OFF	RESET
Gleichzeitige Verarbeitung von anderen als Bewegungsbefehlen	IOBLOCK

8.10.2 RS232C-Steuerungsanweisung

Eine RS232C-Steuerungsanweisung steuert die Ein-/Ausgabe am RS232C-Anschluß.

RS232C-Steuerbefehle

Bewegungsart	Befehl
Daten drucken	PRINT
Daten lesen	INPUT
Daten schreiben	WRITE
Puffer löschen	FLUSH

8.10.3 Steuerungsanweisung für Programmiergerät

Eine Steuerungsanweisung für das Programmiergerät legt die Eingänge und Ausgänge am Programmiergerät fest.

Steuerbefehle für Programmiergerät

Bewegungsart	Befehl
Ausgabe auf Debug-Bildschirm	PRINTDBG
Definition der Bedienfeldtasten	PRINTLBL
Ausgabe auf Meldungsbildschirm	PRINTMSG
Signalton auslösen	BUZZER

8.11 Multitasking-Steuerungsanweisung

Multitasking-Steuerungsanweisungen beinhalten eine Task-Steuerungsanweisung und eine Semaphor-Steuerungsanweisung.

8.11.1 Task-Steuerungsanweisung

Eine Task-Steuerungsanweisung steuert Tasks mit Ausnahme der Tasks, die eine eigene Task-Steuerungsanweisung enthalten.

Steuerbefehle für Tasks

Bewegungsart	Befehl
Task erstellen und ausführen	RUN
Task abbrechen	SUSPEND
Task löschen	KILL
Task schützen	DEFEND

8.11.2 Semaphor-Steuerungsanweisung

Eine Semaphor-Steuerungsanweisung führt Semaphor-bezogene Steuerungen aus.

Semaphor-Steuerbefehle

Bewegungsart	Befehl
Semaphor erstellen	CREATESEM
Semaphor löschen	DELETESEM
Semaphor freigeben	GIVESEM
Semaphor erhalten	TAKESEM
Taskfreigabe wartet auf Semaphor	FLUSHSEM

8.11.3 Spezielle Semaphor-Steueranweisungen

Spezielle Semaphor-Steueranweisungen

Bewegungsart	Befehl
Arm-Semaphor freigeben	GIVEARM
Arm-Semaphor abrufen	TAKEARM
Bildverarbeitungs-Semaphor freigeben	GIVEVIS
Bildverarbeitungs-Semaphor abrufen	TAKEVIS

8.12 Steuerung von Zeit und Datum

Steuerungsanweisungen für Uhrzeit und Datum rufen die aktuelle Uhrzeit und das Datum, die abgelaufene Zeit und die Steuerungsunterbrechung aufgrund der Zeit ab.

Steuerbefehle für Zeit und Datum

Bewegungsart	Befehl
Ruft das aktuelle Datum ab	DATE$
Ruft die aktuelle Uhrzeit ab	TIME$
Ruft die abgelaufene Zeit ab	TIMER

8.13 Fehlerprüfung

Eine Fehlersteuerungsanweisung steuert die Unterbrechung aufgrund eines Fehlers.

Steuerbefehle für Fehlerprüfung

Bewegungsart	Befehl
Definition der Fehlerunterbrechung	ON ERROR GOTO
Fehlercode	ERR
Fehlerzeile	ERL
Fehlerbehebung	RESUME

8.14 Systeminformationen

Mit den folgenden Befehlen erhalten Sie Informationen zum System.

Befehle zu Systeminformationen

Bewegungsart	Befehl
Ruft den Wert für die Umgebungseinstellung des Systems ab	GETENV
Einstellungswert der Systemumgebung zuweisen	LETENV
Gibt Informationen zur ROM-Version aus	VER$
Gibt den Programmstatus aus	STATUS

8.15 Präprozessor

Eine Präprozessor-Anweisung steuert das Ersetzen von Zeichenfolgen oder das Einbinden einer Datei, wenn ein Programm in Ausführungsform umgewandelt (kompiliert) wird.

Präprozessorbefehl

Bewegungsart	Befehl
Ersetzen einer Zeichenfolge durch eine Konstante und einen Makronamen	#define
Abbrechen einer #define-Anweisung	#undef
Datei einbinden	#include
Auftreten eines Pseudo-Fehlers	#error
Bezeichnet Programm-Optimierung	#pragma optimize

8.16 Aufrufen mit einem Wert und einem Bezug

Es kann erwünscht sein, Daten an ein Programm zu übergeben, wenn ein anderes Programm vom aktuellen Programm aufgerufen wird. In diesem Fall können die Daten durch Hinzufügen einer Argumentliste an den aufzurufenden Programmnamen übergeben werden. “Aufruf mit einem Wert” übergibt einen Wert direkt, während “Aufruf mit Bezug” Variablen mit einer Methode zur Übergabe von Argumenten übergibt.
Die Befehle zum Aufruf eines Programmes sind CALL und RUN. Für den CALL-Befehl können beide Methoden verwendet werden; für den RUN-Befehl ist jedoch nur “Aufruf mit einem Wert” zulässig.
Der Typ des an ein Programm übergebenen Arguments muß zu dem im aufzurufenden Programm beschriebenen Argument passen. Daher muß bei Übergabe einer Konstanten ein Typdeklarationszeichen hinzugefügt werden. Und im aufzurufenden Programm muß dem Argumentnamen ein Typdeklarationszeichen hinzugefügt werden, da das Argument als lokale Variable deklariert ist.

8.16.1 Aufruf mit einem Wert

Wenn Konstanten übergeben werden, verwenden Ausdrücke und Zeichenfolgen stets Werte. Wenn eine Variable in Klammern gesetzt ist, wird sie als Ausdruck betrachtet. Daher kann sogar eine Variable mit einem Wert übergeben werden.

Beispiel: Methode, ein Programm PROGRAM SUB1 (AA#) aufzurufen.

- Wenn eine Variable als Wert übergeben wird CALL SUB1 ((D1))
- Wenn eine Konstante übergeben wird CALL SUB1 (10#)
- Wenn ein Ausdruck übergeben wird CALL SUB1 (D1 + D2)

Anmerkung 1: **Wenn eine Variable als Wert übergeben wird, ist der Aufruf mit dem Wert des gesamten Array nicht möglich. Wenn Sie einen solchen Aufruf schreiben, wird er als "Aufruf mit Bezug" verarbeitet.**

Anmerkung 2: **Beachten Sie bei der Übergabe eines Ausdrucks den Typ, den das Ergebnis des Ausdrucks hat. Dieser Typ muß der gleiche sein wie der Typ des Arguments des aufzurufenden Programmes.**

8.16.2 Aufruf mit Bezug

Eine lokale Variable kann als ein Argument übergeben werden.
Zur Bezeichnung eines ganzen Arrays als Argument setzen Sie den Array-Namen in Klammern.

Wenn der Inhalt einer mit Aufruf mit Bezug übergebenen Variablen im aufzurufenden Programm geändert wird, ist die Änderung wirksam, auch wenn der Ablauf zu dem aufrufenden Programm zurückkehrt.

Beispiel 1: Wenn Sie das Programm PROGRAM SUB1 (AA#) aufrufen

- Wenn eine lokale Variable übergeben wird 1 (wenn DD mit DEFDBL deklariert wurde)
  CALL SUB1 (DD#)
- Wenn eine lokale Variable übergeben wird 2 (wenn DA mit DEFDBL deklariert wurde)
  CALL SUB1 (DA)

Anmerkung:	**Ein Wert muß im voraus einer lokalen Variable zugewiesen werden. Im folgenden Fall verwenden Sie einen Wert zur Übergabe.** **CALL SUB1 (D1)**

Beispiel 2: Wenn das Programm PROGRAM SUB2 (BB% (10)) aufgerufen wird

- Wenn der gesamte Array aufgerufen wird (wenn AB% (10) mit DIM deklariert wurde)
  CALL SUB2 (AB% ())

8.17 Bildverarbeitungssteuerung

8.17.1 Bildeingabe/Ausgabe

Die folgenden Befehle steuern Kamerabilder und Bilddaten im Speicher.

Steuerbefehle zur Bildein-/Ausgabe

Bewegungsart	Befehl
Speichert Kamerabilder im Speicher	CAMIN
Funktionseinstellung für Speichern von Kamerabildern	CAMMODE
Einstellung Eingangs-Level für Kamerabilder	CAMLEVEL
Zeigt Kamerabilder auf dem Monitor	VISCAMOUT
Zeigt Speicherbilder auf dem Monitor	VISPLNOUT
Zeigt Zeichenbildschirm-Informationen auf dem Monitor	VISOVERLAY
Einstellung der Nachschlagetabellendaten	VISDEFTABLE
Bezug der Nachschlagetabellendaten	VISREFTABLE

8.17.2 Fenstereinstellung

Die folgenden Befehle werden verwendet, um Fenster (Bereich zum Ausführen der Bildverarbeitung) zu editieren.

Befehle zur Fenstereinstellung

Bewegungsart	Befehl
Einstellen der Fensterinformation	WINDMAKE
Löschen der Fensterinformation	WINDCLR
Kopieren der Fensterinformation	WINDCOPY
Verweisen der Fensterinformation	WIMDREF
Fenster zeichnen	WINDDISP

8.17.3 Zeichnen

Die folgenden Befehle werden zur Steuerung der Zeichenbewegung im Speicher (Prozeßbildschirm) und Überlagerungsspeicher (Zeichenbildschirm) verwendet.

Zeichnungsbefehle

Bewegungsart	Befehl
Zeichnungszielbildschirm angeben	VISSCREEN
Helligkeit der Zeichnung angeben	VISBRIGHT
Bildschirm löschen	VISCLS
Punkt zeichnen	VISPUTP
Linie zeichnen durch Angabe von Länge und Winkel	VISLINE
Linie zeichnen durch Verbindung zweier Punkte	VISPTP
Rechteck zeichnen	VISRECT
Kreis zeichnen	VISCIRCLE
Ellipse zeichnen	VISELLIPSE
Sektor zeichnen	VISSECT
Kreuzsymbol zeichnen	VISCROSS
Zeichendarstellungsposition angeben	VISLOC
Zeicheneinstellung für Zeichnung	VISDEFCHAR
Zeichen anzeigen	VISPRINT

8.17.4 Bildverarbeitung

Die folgenden Befehle werden zur Verarbeitung von Bilddaten verwendet.

Befehle zur Bildverarbeitung

Bewegungsart	Befehl
Verarbeitungsobjektbildschirm angeben	VISWORKPLN
Angegebene Koordinatenhelligkeit erhalten	VISGETP
Histogramm Messung	VISHIST
Histogramm-Ergebnisverweis	VISREFHIST
Berechnung der Binärisierungsstufe	VISLEVEL
Binärisierungsprozeß	VISBINA
Binärisierungsanzeige	VISBINAR
Filterprozeß	VISFILTER
Operation zwischen Bildern	VISMASK
Bildschirmkopie	VISCOPY
Messung von Bereich, Schwerpunkt und Winkel der Hauptachse	VISMEASURE
Projektionsdatenmessung	VISPROJ
Kantenmessung	VISEDGE

8.17.5 Codeerkennung

Der folgende Befehl führt Lesen des QR-Codes aus.

Codeerkennungsbefehl

Bewegungsart	Befehl
QR-Code lesen	VISREADQR

8.17.6 Bezeichung

Die folgenden Befehle werden zur Bezeichnungsverarbeitung verwendet.

Bezeichnungsbefehle

Bewegungsart	Befehl
Bezeichnung ausführen	BLOB
Funktionsmessung für Kennungsnummer	BLOBMEASURE
Bezeichnungsnummer abrufen	BLOBLABEL
Bezeichnungsbild kopieren	BLOBCOPY

8.17.7 Suchfunktion

Die folgenden Befehle werden zur Ausführung der Registrierung und Suche von Bildmodellen verwendet.

Befehle zur Suchfunktion

Bewegungsart	Befehl
Registrierung des Suchmodells	SHDEFMODEL
Information zu registriertem Modell erhalten	SHREFMODEL
Registriertes Modell kopieren	SHCOPYMODEL
Registriertes Modell löschen	SHCLRMODEL
Registriertes Modell anzeigen	SHDISPMODEL
Modellsuche	SHMODEL
Einstellung Eckensuchbedingungen	SHDEFCORNER
Ecken suchen	SHCORNER
Einstellung Kreissuchbedingungen	SHDEFCIRCLE
Kreis suchen	SHCIRCLE

8.17.8 Abrufen von Ergebnissen

Die folgenden Befehle werden zum Abrufen von Informationen zu den Ergebnissen nach der Bildverarbeitung verwendet.

Befehle zum Abrufen von Ergebnissen

Bewegungsart	Befehl
Bildverarbeitungsergebnis abrufen	VISGETNUM
Abrufen der Erkennungsergebnissteuerung	VISGETSTR
Abrufen der X-Koordinate des Bildverarbeitungsergebnisses	VISPOSX
Abrufen der Y-Koordinate des Bildverarbeitungsergebnisses	VISPOSY
Prozeßergebnisstatus abrufen	VISSTATUS

8.17.9 Kalibrierung der Bildverarbeitung

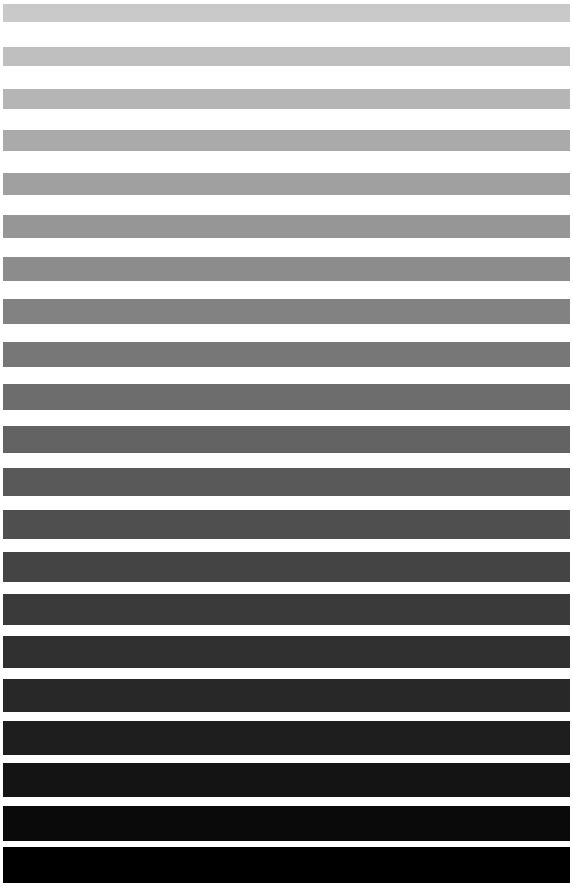
Der folgende Befehl wird zum Abruf der Kalibrierungsdaten (für die Koordinatentransformation zwischen Bildverarbeitung und Roboter) verwendet.

Befehle zur Kalibrierung der Bildverarbeitung

Bewegungsart	Befehl
Koordinatentransformationsdaten erhalten	VISREFCAL

Kapitel 9

Deklarations-anweisungen



Wenn Variablen oder Funktionen in einem Programm verwendet werden, müssen sie mit einer Deklarationsanweisung definiert werden. Hierzu werden die in diesem Kapitel beschriebenen Deklarationsanweisungen verwendet.
Systemvariablen und integrierte Funktionen können in einem Programm jedoch ohne Deklaration verwendet werden.

9.1 Programmname

PROGRAM (Anweisung)

Funktion

Deklariert einen Programmnamen.

Format

PROGRAM <Programmname> [(<Argument>[,<Argument>...])]

Erklärung

Diese Anweisung deklariert die in <Programmname> angegebene Zeichenfolge als Programmname.
Deklarieren Sie einen Programmnamen in der ersten Zeile des Programmes. Wenn keine Deklaration eines Programmnamens in der ersten Zeile angegeben ist, wird der Dateiname als Programmname verwendet.
Hat der Programmname das Format PRO <Nummer>, kann kein Argument verwendet werden.
Das erste Zeichen eines Programmnamens muß ein alphabetisches Zeichen sein. Für einen Programmnamen können maximal 64 Zeichen verwendet werden.
Für <Argument> werden die von der aufrufenden Seite gesendeten Daten verwendet. Wenn eine Variable als Argument mit einer CALL-Anweisung gesendet wird und der Wert der mit <Argument> in der PROGRAM-Anweisung angegebenen Variablen geändert wird, ändert sich auch der Wert der in <Argument> angegebenen Variablen auf der aufrufenden Seite.
Der <Argument> typ muß der gleiche sein wie der Typ in der CALL-Anweisung. Der Typ wird wie folgt mit einer Abschlußziffer oder einem AS-Ausdruck ausgedrückt.

```
PROGRAM    SUB0 (aa%, bb!)
PROGRAM    SUB0 (aa AS INTEGER, bb AS SINGLE)
```

Wenn eine Array-Variable in <Argument> verwendet wird, geben Sie den maximalen Wert der in der DIM-Anweisung definierten Array-Indexnummer sowie die Anzahl der Indexnummern ein.
Es können maximal 32 <Argument>e verwendet werden.
Über das Bedienfeld und über externe Geräte können nur Programme mit dem Format PRO <Nummer> gestartet werden.

Verwandte Begriffe

CALL

Beispiel

```
PROGRAM PRO1                    'Deklariert PRO1 als Programmname.
PROGRAM SUB2 ( la, lb%, lc# )   'Deklariert SUB2 mit den Argumenten la, lb%,
                                'und lc# als Programm.
PROGRAM SUB3 ( lb%( 12, 5 ) )   'Deklariert SUB3 als Programm und empfängt Argumente
                                'mit einem zweidimensionalen Array lb%. In diesem
                                'Beispiel lauten die Array-
                                'Indexnummern 12 und 5.
PROGRAM SUB0 ( la%, lb! )
PROGRAM SUB0 ( la As Integer, lb As Single )
```

9.2 Koordinaten des Interferenzbereichs

AREA (Anweisung)

Funktion

Deklariert den Bereich, in dem eine Interferenzprüfung ausgeführt wird.

Format

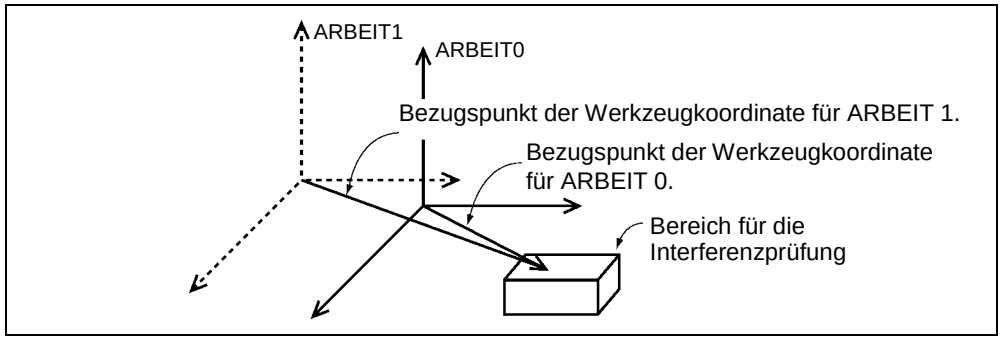
AREA <Bereichsnummer>, <Position>, <Vektor>, <E/A-Nummer>, <Variable vom Typ Position zum Speichern der Interferenzposition>

Erklärung

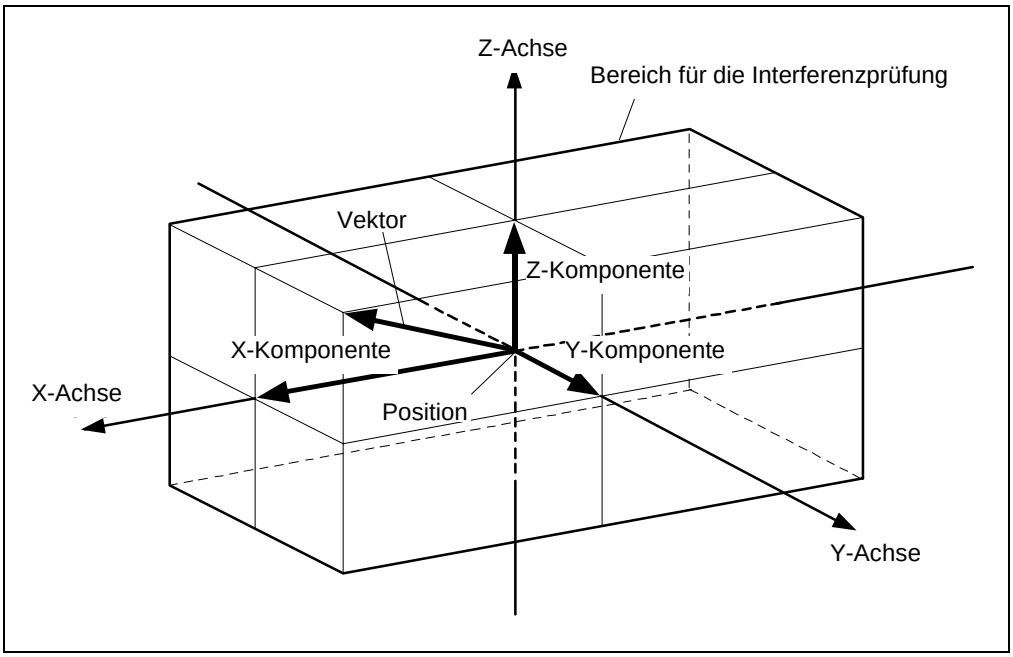
Diese Anweisung deklariert einen Bereich für die Interferenzprüfung.
Mit <Bereichsnummer> können 8 Bereiche mit den Nummern 0 bis 7 deklariert werden.
<Position> ist die Mittelpunktposition und der Winkel eines Bereiches für die Interferenzprüfung.
<Vektor> kennzeichnet die Bereichszone für die Interferenzprüfung.
Die Seitenlänge eines Bereiches wird zweimal jede Komponente von <Vektor>.
Geben Sie die E/A-Nummer, die beim Auftreten einer Interferenz im Prüfbereich gesetzt wird, als <E/A-Nummer> an. Der Status von E/A wird beibehalten, bis RESETAREA ausgeführt wird oder E/A zurückgesetzt wird.
Geben Sie die Anzahl der Positionstypen an, wobei die Interferenzposition in der <Nummer der Positionstypvariablen zum Speichern der Interferenzposition> gespeichert wird.
Bei einer Interferenzprüfung vergleicht das System die Ausgangsposition des Werkzeugkoordinatensystems mit dem Quader, der im Bereich für die Interferenzprüfung definiert wurde. Wenn das System eine Ausgangsposition einer Werkzeugkoordinate erkennt, die innerhalb des Bereiches für die Interferenzprüfung liegt, definiert das System dies als eine Interferenz.
Eine Variablennummer des Typs Interferenzpositions-Speicherposition speichert eine Ausgangsposition einer Werkzeugkoordinate im Benutzerkoordinatensystem, wenn eine Interferenz erkannt wird (wenn E/A gesetzt wird). Im allgemeinen wird dies jedoch ein Punkt auf einer Ebene eines Quaders, wenn nach der Ausführung von SETAREA eine Ausgangsposition einer Werkzeugkoordinate in dem Quader vorhanden ist, und die Position wird gespeichert.

Anmerkungen

Die Position des Mittelpunkts eines Bereiches basiert immer auf ARBEIT0.
Auch wenn das Benutzerkoordinatensystem geändert wird, bleibt die Position des Bereiches für die Interferenzprüfung unverändert.



Der Interferenzbereich kann mit WINCAPSII oder mit dem Programmiergerät definiert werden.



Verwandte Begriffe

SETAREA, RESETAREA, AREAPOS, AREASIZE

Beispiel

6/4 Achsen

```
AREA 2, P50, V10, 104, 55
                    'Definiert einen Bereich an Nummer 2, definiert 'durch
                    P50 und V10.
SETAREA 2           'Macht die Bereichsprüfung von Nummer 2 gültig.
RESETAREA 2         'Macht die Bereichsprüfung von Nummer 2 ungültig.
```

6 Achsen

```
AREA 2, P50+(100, 100, 0, 10, 0, 0), V10, 104, 55
                    'Definiert einen Bereich an Nummer 2, definiert 'durch
                    P50+(100, 100, 0, 10, 0, 0) und V50.
SETAREA 2           'Macht die Bereichsprüfung von Nummer 2 gültig.
RESETAREA 2         'Macht die Bereichsprüfung von Nummer 2 ungültig.
```

4 Achsen

```
AREA 2, P50+(100, 100, 0, 10), V10, 104, 55
                    'Definiert einen Bereich an Nummer 2, definiert 'durch
                    P50+(100, 100, 0, 10) und V50.
SETAREA 2           'Macht die Bereichsprüfung von Nummer 2 gültig.
RESETAREA 2         'Macht die Bereichsprüfung von Nummer 2 ungültig.
```

9.3 Benutzerfunktion

DEF FN (Anweisung) [Entspricht SLIM]

Funktion

Deklariert eine benutzerdefinierte Funktion.

Format

DEF FN <Funktionsname>[<Abschlußziffer>] = <Konstante>
DEF FN <Funktionsname>[<Abschlußziffer>](<Argument>[,<Argument>...]) = <Arithmetischer Ausdruck>

Erklärung

Diese Anweisung deklariert einen <Funktionsnamen>, der mit FN beginnt als benutzerdefinierte Funktion.
Kennzeichnen Sie einen in <Arithmetischer Ausdruck> verwendeten Variablennamen als <Argument>.
Die <Abschlußziffer> kann ignoriert werden. Falls sie jedoch hinzugefügt wird, kann gleichzeitig der Variablentyp deklariert werden. Für Abschlußziffern kann folgendes verwendet werden.

Abschlußziffer für Typ Ganze Zahl : %
Abschlußziffer für Typ Real mit einfacher Genauigkeit: !
Abschlußziffer für Typ Real mit doppelter Genauigkeit: #
Abschlußziffer für Zeichenfolge: $

Wenn die Abschlußziffer ignoriert wird, gilt der Typ Real mit einfacher Genauigkeit.
Ein anderer Variablentyp mit dem gleichen Variablennamen kann nicht deklariert werden.

Beispiel

```
DEF FND$ = “DENSO”                        'Deklariert FND$ als benutzerdefinierte Funktion.
DEF FNLAP# (Radius) = 2 * PI * Radius     'Deklariert FNLAP# (Radius) als benutzerdefinierte
                                          'Funktion des Typs Real mit doppelter Genauigkeit.
DEF FNAREA (Radius) = PI * POW(Radius, 2)
                                          'Deklariert FNAREA (Radius) als benutzerdefinierte
                                          'Funktion des Typs Real mit einfacher Genauigkeit.
PRINT #1, FND$                            'Gibt “DENSO” über ch1 aus.
PRINT #2, HANKEI
PRINT #1, FNLAP# (HANKEI)                 'Gibt den Wert (2 * PI * HANKEI) über ch1 aus.
PRINT #2, FNAREA (HANKEI)                 'Gibt den Wert (PI * POW(HANKEI, 2)
                                          'über ch2 aus.
```

9.4 Ausgangskoordinaten

HOME (Anweisung) [Entspricht SLIM]

Funktion

Deklariert zufällige Koordinaten als Ausgangsposition.

Format

HOME <Positionstyp>

Erklärung

Deklariert zufällige, mit <Positionstyp> angegebene Koordinaten als Ausgangsposition.
Definieren Sie für jedes Programm eine Ausgangsposition.
Wenn mehrere HOME-Anweisungen deklariert sind, wird die Angabe aus der jüngsten Deklaration als Ausgangsposition verwendet.

Verwandte Begriffe

GOHOME

Beispiel

6 Achsen

```
DIM lp1 As Point
HOME (350, 0, 450, 0, 0, 180)
                'Deklariert die Koordinaten (350, 0, 450, 0, 0, 180)
                'als Ausgangsposition.
HOME lp1        'Deklariert die Koordinaten von lp1 als Ausgangsposition.
```

4 Achsen

```
DIM lp1 As Point
HOME (200, 300, 300, 45, 0)
                'Deklariert die Koordinaten (200, 300, 300, 45, 0)
                'als Ausgangsposition.
HOME lp1        'Deklariert die Koordinaten von lp1 als Ausgangsposition.
```

9.5 Werkzeugkoordinaten

TOOL (Anweisung)

Funktion

Deklariert ein Werkzeugkoordinatensystem.

Format

TOOL <Werkzeugkoordinatensystemnummer>, <Positionstyp>

Erklärung

Deklariert eine mit <Positionstyp> angegebene Position als Werkzeugkoordinatensystem, das mit <Werkzeugkoordinatensystemnummer> angegeben wurde.
Für die Werkzeugkoordinatensystemnummer kann eine Nummer von 1 bis 63 angegeben werden.

Verwandte Begriffe

CHANGETOOL, TOOLPOS

Beispiel

6 Achsen

```
DIM lp1 As Point
TOOL 1, lp1       'Deklariert lp1 als Werkzeugkoordinatensystem des
                  'Werkzeugs 'Koordinatensystemnummer 1.
TOOL 2, (100, 100, 50, 0, 90, 0)
```

4 Achsen

```
DIM lp1 As Point
TOOL 1, lp1       'Deklariert lp1 als Werkzeugkoordinatensystem des
                  'Werkzeugs 'Koordinatensystemnummer 1.
TOOL 2, (100, 100, 50, 0)
```

Anmerkungen

Die mit diesem Befehl geänderten Werte behalten den neuen Wert bei, solange der Strom eingeschaltet bleibt. Wenn die Werte auch über das Ausschalten des Systems hinaus beibehalten werden sollen, müssen Sie die Systemparameter speichern (siehe “Einstellungen” in Kapitel 5).

9.6 Arbeitskoordinaten

WORK (Anweisung)

Funktion

Deklariert ein Benutzerkoordinatensystem.

Format

WORK <Benutzerkoordinatensystemnummer>, <Positionstyp>

Erklärung

Deklariert die Benutzerkoordinaten, die der unter <Positionstyp> angegebenen Variablen des Typs Position zugeordnet wurden als das Benutzerkoordinatensystem, das mit <Benutzerkoordinatensystemnummer> angegeben wurde.
Für die Benutzerkoordinatensystemnummer kann eine Nummer von 1 bis 7 angegeben werden.

Verwandte Begriffe

CHANGEWORK, WORKPOS

Beispiel

6 Achsen

```
DEFINT li1 li2
WORK 1, P1        'Deklariert Koordinaten, die der unter Positionstyp
                  'angegebenen Variablen 2 des Typs Position zugeord 'net
                  wurden als das Benutzerkoordinatensystem, das 'mit
                  Benutzerkoordinatensystemnummer 1 angegeben 'wurde.
WORK li1, P[li2]  'Deklariert Koordinaten, die der mit li2 angegebenen
                  'Variablen des Typs Position zugeordnet wurden als 'das
                  Benutzerkoordinatensystem, das mit Benutzerko
                  'ordinatensystemnummer in li1 angegeben wurde.
WORK li1, (100, 100, 50, 0, 0, 90)
```

4 Achsen

```
DEFINT li1 li2
WORK 1, P1        'Deklariert Koordinaten, die der unter Positionstyp
                  'angegebenen Variablen 2 des Typs Position
                  'zugeordnet wurden als das 'Benutzerkoordinatensystem,
                  'das mit Benutzerkoordinatensystemnummer 1 angegeben
                  'wurde.
WORK li1, P[li2]  'Deklariert Koordinaten, die der mit li2 angegebenen
                  'Variablen des Typs Position zugeordnet wurden als 'das
                  Benutzerkoordinatensystem, das mit
                  'Benutzerkoor dinatensystemnummer in li1 angegeben
                  'wurde.
WORK li1, (100, 100, 50, 0)
```

Anmerkungen

(1) Wenn die Werte für den Arbeitsbereich des Roboters größer als die X-, Y- und Z-Elemente der Arbeitskoodinaten gesetzt werden, kann sich die angegebene Roboterposition wegen eines arithmetischen Überlaufs verschieben. Verwenden Sie Werte innerhalb des Roboter-Arbeitsbereichs für die X-, Y- und Z-Elemente der Arbeitskoordinaten.

(2) Die mit diesem Befehl geänderten Werte behalten den neuen Wert bei, solange der Strom eingeschaltet bleibt. Wenn die Werte auch über das Ausschalten des Systems hinaus beibehalten werden sollen, müssen Sie die Systemparameter speichern (siehe “Einstellungen” in Kapitel 5).

9.7 Lokale Variablen

DEFINT (Anweisung)

Funktion

Deklariert eine ganzzahlige Variable. Der gültige Bereich für die Ganzzahl reicht von -2147483648 bis 2147483647.

Format

DEFINT <Variablenname>[=<Konstante>][,<Variablenname>[=<Konstante>]...]

Erklärung

Diese Anweisung deklariert die mit <Variablenname> angegebene Variable als Variable des Typs Ganzzahl. Durch Angabe einer Konstanten nach dem <Variablenname> kann gleichzeitig mit der Deklaration eine Initialisierung ausgeführt werden.
Sie können mehrere Variablennamen gleichzeitig deklarieren, indem Sie die Namen mit "," voneinander abgrenzen.

Verwandte Begriffe

DEFDBL, DEFSNG, DEFSTR

Beispiel

```
DEFINT lix, liy, liz      'Deklariert lix, liy und liz als Variablen des Typs Ganze Zahl.
DEFINT lix = 1            'Deklariert lix als Variable des Typs Ganze Zahl und setzt
                          'den Anfangswert auf 1.
```

DEFSNG (Anweisung)

Funktion

Deklariert eine Variable vom Typ Real mit einfacher Genauigkeit. Der gültige Bereich für Real-Variablen mit einfacher Genauigkeit reicht von -3,402823E+38 bis 3,402823E+38.

Format

DEFSNG <Variablenname>[=<Konstante>][,<Variablenname>[=<Konstante>]...]

Erklärung

Diese Anweisung deklariert eine Variable, die mit <Variablenname> angegeben wurde als Variable des Typs Real mit einfacher Genauigkeit. Durch Angabe einer Konstanten nach dem <Variablenname> kann gleichzeitig mit der Deklaration eine Initialisierung ausgeführt werden.
Sie können mehrere Variablennamen gleichzeitig deklarieren, indem Sie die Namen durch Kommas (“,”) voneinander abgrenzen.

Verwandte Begriffe

DEFDBL, DEFINT, DEFSTR

Beispiel

```
DEFSNG lfx, lfy, lfz     'Deklariert lfx, lfy und lfz als Variablen des Typs Real
                         'mit einfacher Genauigkeit.
DEFSNG lfx = 1.0         'Deklariert lfx als Variablen des Typs Real mit einfacher
                         'Genauigkeit und setzt den Anfangswert auf 1.0.
```

DEFDBL (Anweisung)

Funktion

Deklariert eine Variable vom Typ Real mit doppelter Genauigkeit. Der gültige Bereich für Real-Variablen mit doppelter Genauigkeit reicht von -1,79769313486231D + 308 bis 1,79769313486231D + 308.

Format

DEFDBL <Variablenname>[=<Konstante>][,<Variablenname>[=<Konstante>]...]

Erklärung

Deklariert eine Variable, die mit <Variablenname> angegeben wurde als Variable des Typs Real mit doppelter Genauigkeit. Durch Angabe einer Konstanten nach dem <Variablenname> kann gleichzeitig mit der Deklaration eine Initialisierung ausgeführt werden.
Sie können mehrere Variablennamen gleichzeitig deklarieren, indem Sie die Namen durch Kommas (“,”) voneinander abgrenzen.

Verwandte Begriffe

DEFINT, DEFSNG, DEFSTR

Beispiel

```
DEFDBL ldx, ldy, ldz     'Deklariert ldx, ldy und ldz als Variablen des Typs Real
                         'mit doppelter Genauigkeit.
DEFDBL ldx = 1.0         'Deklariert ldx als Variable des Typs Real mit doppelter
                         'Genauigkeit und setzt den Anfangswert auf 1.0.
```

DEFSTR (Anweisung)

Funktion

Deklariert eine Variable vom Typ Zeichenfolge. Sie können maximal 247 Zeichen als Zeichenfolge eingeben.

Format

DEFSTR <Variablenname>[=<Konstante>][,<Variablenname>[=<Konstante>]...]

Erklärung

Deklariert eine mit <Variablenname> angegebene Variable als Zeichenfolge. Durch Angabe einer Konstanten nach dem <Variablenname> kann gleichzeitig mit der Deklaration eine Initialisierung ausgeführt werden.
Sie können mehrere Variablennamen gleichzeitig deklarieren, indem Sie die Namen durch Kommas (“,”) voneinander abgrenzen.

Verwandte Begriffe

DEFDBL, DEFINT, DEFSNG

Beispiel

```
DEFSTR lsx, lsy, lsz     'Deklariert lsx, lsy und lsz als Variablen des Typs
                         'Zeichenfolge.
DEFSTR lsx = “DENSO”     'Deklariert lsx als Variable des Typs Zeichenfolge und
                         'setzt den Anfangswert auf “DENSO”.
```

DEFVEC (Anweisung)

Funktion

Deklariert eine Variable vom Typ Vektor.

Format

DEFVEC <Variablenname>[=<Konstante des Typs Vektor>][,<Variablenname>[=<Konstante des Typs Vektor>]...]

Erklärung

Diese Anweisung deklariert eine mit <Variablenname> angegebene Variable als Variable des Typs Vektor. Durch Angabe einer Konstanten nach dem <Variablenname> kann gleichzeitig mit der Deklaration eine Initialisierung ausgeführt werden.
Sie können mehrere Variablennamen gleichzeitig deklarieren, indem Sie die Namen mit “,” voneinander abgrenzen.

Verwandte Begriffe

DEFJNT, DEFPOS, DEFTRN

Beispiel

```
DEFVEC lvx, lvy, lvz        'Deklariert lvx, lvy und lvz als Variablen des Typs Vektor.
DEFVEC lvx = (10, 10, 5)    'Deklariert lvx als Variable des Typs Vektor und setzt
                            'den Anfangswert auf (10, 10, 5).
```

DEFPOS (Anweisung)

Funktion

Deklariert eine Variable vom Typ Position.

Format

DEFPOS <Variablenname>[=<Konstante des Typs Position>][,<Variablenname> [=<Konstante des Typs Position>]...]

Erklärung

Deklariert eine mit <Variablenname> angegebene Variable als Variable des Typs Position. Durch Angabe einer Konstanten nach dem <Variablenname> kann gleichzeitig mit der Deklaration eine Initialisierung ausgeführt werden.
Sie können mehrere Variablennamen gleichzeitig deklarieren, indem Sie die Namen mit "," voneinander abgrenzen.

Verwandte Begriffe

DEFJNT, DEFTRN, DEFVEC

Beispiel

6 Achsen

```
DEFPOS lpx, lpy, lpz
                    'Deklariert lpx, lpy und lpz als Variablen des Typs
                    'Position.
DEFPOS lpx = (10, 10, 5, 0, 9, 0, 1)
                    'Deklariert lpx als Variable des Typs Position und 'setzt
                    den Anfangswert auf (10, 10, 5, 0, 9, 0, 1).
```

6 Achsen

```
DEFPOS lpx, lpy, lpz
                    'Deklariert lpx, lpy und lpz als Variablen des Typs
                    'Position.
DEFPOS lpx = (100, 100, 300, 45, 0)
                    'Deklariert lpx als Variable des Typs Position und 'setzt
                    den Anfangswert auf (100, 100, 300, 45, 0).
```

DEFJNT (Anweisung)

Funktion

Deklariert eine Variable vom Typ Gelenk.

Format

DEFJNT <Variablenname>[=<Konstante des Typs Gelenk>][,<Variablenname>[=<Konstante des Typs Gelenk>]...]

Erklärung

Diese Anweisung deklariert eine mit <Variablenname> angegebene Variable als Variable vom Typ Gelenk. Durch Angabe einer Konstanten nach dem <Variablenname> kann gleichzeitig mit der Deklaration eine Initialisierung ausgeführt werden.
Sie können mehrere Variablennamen gleichzeitig deklarieren, indem Sie die Namen mit "," voneinander abgrenzen.

Verwandte Begriffe

DEFPOS, DEFTRN, DEFVEC

Beispiel

6 Achsen

```
DEFJNT ljx, ljy, ljz
                    'Deklariert ljx, ljy und ljz als Variable des Typs
                    'Gelenk.
DEFJNT ljx = (10, 10, 5, 0, 9, 0)
                    'Deklariert ljx als Variable des Typs Gelenk und 'setzt
                    den Anfangswert auf (10, 10, 5, 0, 9, 0).
```

4 Achsen

```
DEFJNT ljx, ljy, ljz
                    'Deklariert ljx, ljy und ljz als Variable des Typs
                    'Gelenk.
DEFJNT ljx = (10, 10, 5, 0)
                    'Deklariert ljx als Variable des Typs Gelenk und 'setzt
                    den Anfangswert auf (10, 10, 5, 0).
```

DEFTRN (Anweisung)

Funktion

Deklariert eine Variable mit dem Typ Homogene Transformation.

Format

DEFTRN <Variablenname>[=<Konstante des Typs Homogene Transformation>][,<Variablenname>[=<Konstante des Typs Homogene Transformation>]...]

Erklärung

Diese Anweisung deklariert eine Variable, die mit <Variablenname> angegeben wurde als Variable des Typs Homogene Transformation. Durch Angabe einer Konstanten nach dem <Variablenname> kann gleichzeitig mit der Deklaration eine Initialisierung ausgeführt werden.
Sie können mehrere Variablennamen gleichzeitig deklarieren, indem Sie die Namen mit "," voneinander abgrenzen.

Verwandte Begriffe

DEFJNT, DEFPOS, DEFVEC

Beispiel

```
DEFTRN ltx, lty, ltz    'Deklariert ltx, lty und ltz als Variablen des Typs
                        'Homogene Transformation.
DEFTRN ltx = (10, 10, 5, 20, 20, 10, 30, 30, 15)
                        'Deklariert ltx als Variable des Typs Homogene Transformation
                        'und setzt den Anfangswert auf
                        '(10, 10, 5, 20, 20, 10, 30, 30, 15).
```

DEFIO (Anweisung) [Entspricht SLIM]

Funktion

Deklariert eine E/A-Variable ensprechend dem Ein-/Ausgabeanschluß.

Format

DEFIO <Variablenname> = <E/A-Variablentyp>,<Anschlußadresse>[,<Maskierungsdaten>]

Erklärung

Diese Anweisung deklariert eine mit <Variablenname> angegebene Variable als E/A-Variable.
<E/A-Variablentyp> Wählt den Typ der E/A-Variablen aus. E/A-Variablentypen sind u. a. BIT, BYTE, WORD und INTEGER. Gebeen Sie einen Bereich von 1 Bit für einen BIT-Typ an, 8 Bit für den Typ BYTE, 16 Bit für den Typ WORD und 32 Bit für den Typ INTEGER.

<Anschlußadresse>	Gibt die Anfangs-Anschlußnummer für die Eingabe/Ausgabe an.
<Maskierungsdaten>	Bei einem Eingabeanschluß werden AND (Produkt-Set) aus den Eingabedaten und die Maskierungsdaten verwendet. Bei einem Ausgabeanschluß werden AND (Produkt-Set) aus den Ausgabedaten und die Maskierungsdaten verwendet; der Ausgabestatus eines Bits, für das keine Maskierung gesetzt wurde, ändert sich jedoch nicht.

Verwandte Begriffe

IN, OUT, SET, RESET

Beispiel

```
DEFIO samp1 = BIT, 1
                    'Deklariert samp1 als E/A-Variable des Typs BIT mit dem
                    'Anfang an Anschluß 1. Der Rückgabewert von samp1 wird zu einer
                    '1-Bit-Ganzzahl mit dem Wert 1 oder 0; dieser Wert gibt den
                    'Status von Anschluß 1 an.
DEFIO samp2 = BYTE, 10, &B00010000
                    'Deklariert samp2 mit Maskierungsdaten als E/A-Variable des Typs
                    'BYTE mit dem Anfang an Anschluß 10. Der Rückgabewert von samp2
                    'wird zu einer 8-Bit-Ganzzahl mit dem Wert 0 oder 16; dieser Wert
                    'gibt den Status von Anschluß 10 an.
DEFIO samp3 = WORD, 15
                    'Deklariert samp3 als E/A-Variable des Typs WORD mit dem Anfang an
                    'Anschluß 15. Der Rückgabewert von samp3 wird zu einer 16-Bit-
                    'Ganzzahl mit dem Wert 0~&Hffff; dieser Wert gibt den Status der
                    'Anschlüsse 15 bis 30 an.
DEFIO samp4 = INTEGER, 1
                    'Deklariert samp4 als E/A-Variable des Typs INTEGER mit dem
                    'Anfang an Anschluß 1. Der Rückgabewert von samp4 wird zu einer
                    '32-Bit-Ganzzahl mit dem Wert 0~&Hffffffff; dieser Wert gibt den
                    'Status der Anschlüsse 1 bis 32 an.
```

9.8 Array

DIM (Anweisung) [Entspricht SLIM]

Funktion

Deklariert einen Array.

Format

DIM <Variablenname>[<Abschlußziffer>] [(<Anzahl Elemente>[,<Anzahl Elemente>[,<Anzahl Elemente>]])][AS<Variablentyp>][,<Variablenname> [<Abschlußziffer>]...]

Erklärung

Diese Anweisung deklariert eine mit <Variablenname> angegebene Variable als Array-Variable.
Durch Hinzufügen der <Abschlußziffer> kann das System den Variablentyp gemeinsam deklarieren.
Die folgenden Abschlußziffern stehen zur Verfügung.

Abschlußziffer für Typ Ganze Zahl : %
Abschlußziffer für Typ Real mit einfacher Genauigkeit: !
Abschlußziffer für Typ Real mit doppelter Genauigkeit: #
Abschlußziffer für Zeichenfolge: $

Wenn die Abschlußziffer weggelassen wird, betrachtet das System die Anweisung als Anweisung einer Real-Zahl mit einfacher Genauigkeit.
Geben Sie die maximale Anzahl der Elemente für den Array unter <Anzahl Elemente> an. Die maximale Anzahl muß 1 oder größer sein.
Die Gesamtzahl der Elemente darf nicht größer als 32767 sein.
Für DIM-Anweisungen können bis zu drei Dimensionen verwendet werden.
Die Indexzahl kann im Bereich von 0 bis (Anzahl Elemente – 1) liegen.
Sie können den Variablentyp durch Angabe von AS <Variablentyp> festlegen.
Beachten Sie, daß der AS-Ausdruck und die Abschlußziffer nicht gleichzeitig angegeben werden können.

Die nachfolgende Tabelle zeigt die <Variablentypen>, die im AS-Ausdruck verwendet werden können.

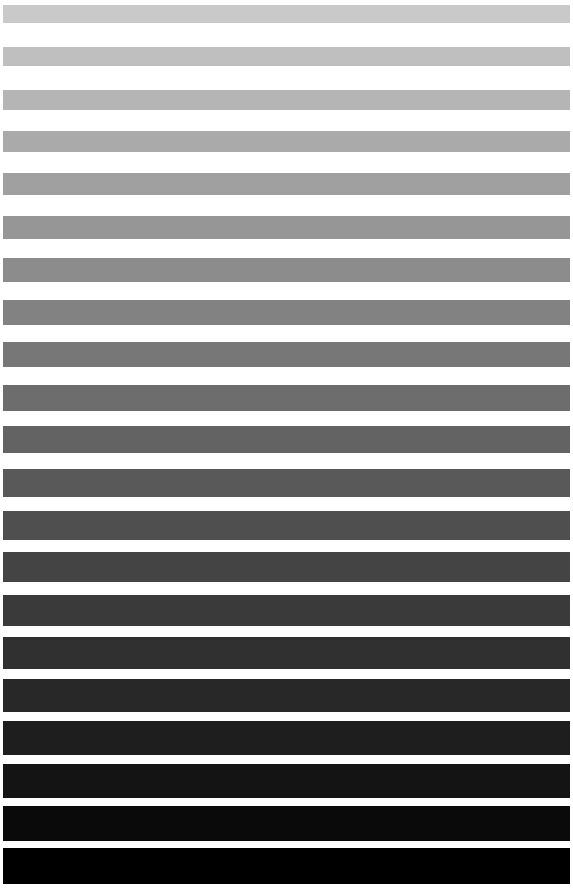
Variablentyp	Kennung	Variablentyp	Kennung
Lange ganze Zahl	INTEGER	Vektor	VECTOR
Real mit einfacher Genauigkeit	SINGLE	Position	POSITION
Real mit doppelter Genauigkeit	DOUBLE	Gelenk	JOINT
Zeichenfolge	STRING	Homogene Transformation	TRANS

Beispiel

```
DIM samp1(5)           'Deklariert samp1 als Array-Variable vom Typ Real
                       'mit einfacher Genauigkeit und der Größe (5).
DIM samp2(10, 10)      'Deklariert samp2 als Array-Variable vom Typ Real
                       'mit einfacher Genauigkeit und der Größe (10, 10).
DIM samp3(20, 5, 10)   'Deklariert samp3 als Array-Variable vom Typ Real
                       'mit einfacher Genauigkeit und der Größe (20, 5, 10).
DIM samp4% (3, 3, 3)   'Deklariert samp4 als Array-Variable vom Typ Ganze Zahl
                       'mit der Größe (3, 3, 3).
DIM samp5! (4, 3)      'Deklariert samp5 als Array-Variable vom Typ Real
                       'mit einfacher Genauigkeit und der Größe (4, 3).
DIM samp6# (3)         'Deklariert samp6 als Array-Variable vom Typ Real
                       'mit doppelter Genauigkeit und der Größe (3).
```

Kapitel 10

Zuweisungs-anweisungen



Verwenden Sie Zuweisungsanweisungen zum Zuordnen von Werten zu Variablen. Achten Sie auf die richtige Verwendung von Befehlen entsprechend dem Typ bzw. Inhalt der entsprechenden Werte.

10.1 Variablen

LET (Anweisung) [Entspricht SLIM]

Funktion

Weist einer Variablen einen Wert zu.

Format

[LET] <Variablenname> = <Arithmetischer Ausdruck>

Erklärung

Sie können [LET] ignorieren.
Normalerweise müssen der Typ der <Variablen> und der Typ des <arithmetischen Ausdrucks> gleich sein.
Wenn der Typ der Variablen und des arithmetischen Ausdrucks unterschiedlich sind, werden sie wie folgt umgewandelt.

- <Variablenname> wird umgewandelt in den Variablentyp.
- Beim Umwandeln einer Real-Variablen in eine Ganze Zahl werden die Dezimalstellen abgerundet.
- Beim Umwandeln eines Real-Typs mit doppelter Genauigkeit zu einem Real-Typ mit einfacher Genauigkeit wird das Ergebnis auf 7 signifikante Stellen abgerundet.
- Berechnungen werden entsprechend der Genauigkeit des Typs mit der höheren Genauigkeit durchgeführt.

Beispiel

6/4 Achsen

```
DEFINT li1, li2
LET li1 = li2 + 1 'Weist li1 den Wert (li2 + 1) zu.
li1 = li2 + 1     'Weist li1 den Wert (li2 + 1) zu (wie in der vorigen
                  'Anweisung).
```

6 Achsen

```
DEFPOS lp1, lp2
DEFJNT lj1, lj2
lp1, = lp2 + (10, 10, 10, 0, 0, 0)
                  'Weist lp1 den Wert (lp2 + (10,10,10,0,0,0)) zu.
lj1, = lj2 + (10, 20, 30, 40, 0, 0)
                  'Weist lj1 den Wert (lj2 + (10,20,30,40,0,0)) zu.
```

4 Achsen

```
DEFPOS lp1, lp2
DEFJNT lj1, lj2
lp1, = lp2 + (10, 10, 10, 20)
                  'Weist lp1 den Wert (lp2 + (10, 10, 10, 20) ) zu.
lj1, = lj2 + (10, 20, 30, 40)
                  'Weist lj1 den Wert (lj2 + (10, 20, 30, 40) ) zu.
```

10.2 Vektor

LETA (Anweisung)

Funktion

Weist einem Anfahrvektor des homogenen Transformationstyps einen Wert zu.

Format

LETA <Variable des Typs Homogene Transformation> = <Vektor>

Erklärung

Die Zuweisungsanweisung beginnt mit LETA, und die rechte Seite der Anweisung ist ein Vektor. Durch die Ausführung dieser Zuweisung wird der Anfahrvektor von <Variable des Typs Homogene Transformation> in <Vektor> geändert.

Verwandte Begriffe

LETO, LETP

Beispiel

```
DEFTRN lt1, lt2, lt3
DEFVEC lv1, lv2
LETA lt1 = AVEC(lt3)    'Weist einen Anfahrvektor lt3 dem Anfahrvektor lt1 zu.
LETA lt2 = lv1 x lv2    'Weist den Wert (lv1 x lv2) dem Anfahrvektor lt2 zu.
```

LETO (Anweisung)

Funktion

Weist einem Orientierungsvektor des homogenen Transformationstyps einen Wert zu.

Format

LETO <Variable des Typs Homogene Transformation> = <Vektor>

Erklärung

Die Zuweisungsanweisung beginnt mit LETO, und die rechte Seite des Ausdrucks ist ein Vektor. Durch die Ausführung dieser Zuweisung wird der Orientierungsvektor von <Variable des Typs Homogene Transformation> in <Vektor> geändert.

Verwandte Begriffe

LETA, LETP

Beispiel

```
DEFTRN lt1, lt2, lt3
DEFVEC lv1, lv2
LETO lt1 = OVEC(lt3)   'Weist den Orientierungsvektor lt3 dem Orientierungsvektor lt1 zu.
LETO lt2 = lv1 x lv2   'Weist den Wert (lv1 x lv2) dem Orientierungsvektor lt2 zu.
```

LETP (Anweisung)

Funktion

Weist einem Positionsvektor des Typs Position oder Homogene Transformation einen Wert zu.

Format

LETP {<Variable vom Typ Position>|<Variable vom Typ Homogene Transformation >} = <Vektor>

Erklärung

Die Anweisung beginnt mit LETP, und die rechte Seite des Ausdrucks ist ein Vektor. Durch die Ausführung dieser Zuweisung wird der Positionsvektor von <Variable vom Typ Position> oder <Variable vom Typ Homogene Transformation> in <Vektor> geändert.

Verwandte Begriffe

LETO, LETA

Beispiel

```
DEFTRN lt1, lt2
LETP lt1 = PVEC(lt2)   'Weist den Positionsvektor lt2 dem Positionsvektor lt1 zu.
```

10.3 Armstellung

LETF (Anweisung)

Funktion

Weist einer Armstellungskomponente des Typs Position oder Homogene Transformation einen Wert zu.

Format

LETF <Variable vom Typ Position> | <Variable vom Typ Homogene Tansformation>

Erklärung

Die Anweisung beginnt mit LETF, und die rechte Seite des Ausdrucks ist <Armstellung>. Durch diese Zuweisung wird die Armstellungskomponente der <Variablen des Typs Position> auf den Wert von <Armstellung> geändert.

Verwandte Begriffe

CURFIG, FIG, Roboter-Armstellung (Anhang 2)

Beispiel

```
DEFPOS lp1, lp2
LETF lp1 = 1          'Stellt die Armstellung lp1 auf LINKS-OBEN-KIPPEN ein.
LETF lp2 = CURFIG     'Stellt die Armstellung lp2 auf die aktuelle Haltung ein.
```

10.4 Verbindungswinkel

LETJ (Anweisung)

Funktion

Weist einem angegebenen Verbindungswinkel des Typs Gelenk einen Wert zu.

Format

LETJ <Achsennummer>, <Variable vom Typ Gelenk> = <Arithmetischer Ausdruck>

Erklärung

Die Anweisung beginnt mit LETJ <Achsennummer>, und die rechte Seite des Ausdrucks lautet <Arithmetischer Ausdruck>. Durch diese Zuweisung wird der mit <Achsennummer> von <Variable vom Typ Gelenk> angegebene Verbindungswinkel auf den Wert von <Arithmetischer Ausdruck> geändert.

Verwandte Begriffe

JOINT

Beispiel

```
DEFJNT lj1, lj2, lj3
DEFSNG lf1, lf2
LETJ 1, lj1 = JOINT(2, lj3) 'Weist den Verbindungswinkel der zweiten Achse von lj3 der 1.
                            'Achse von lj1 zu.
LETJ 3, lj2 = lf1 - lf2     'Weist den Wert von (lf1 - lf2) der 3. Achse von lj2 zu.
```

10.5 Haltung

LETR (Anweisung)

Funktion

Weist drei Rotationskomponenten des Typs Position einen Wert zu.

Format

LETR <Variable vom Typ Position> = <Vektor>

Erklärung

Die Anweisung beginnt mit LETR, und die rechte Seite des Ausdrucks ist ein Vektor. Durch diese Zuweisung wird die Armstellung der <Variablen des Typs Position> auf <Vektor> geändert.

Verwandte Begriffe

LETP, LETRX, LETRY, LETRZ

Beispiel

```
DEFPOS lp1, lp2
LETR lp1 = RVEC(lp2)   'Weist die Armstellung von lp2 der von lp1 zu.
```

10.6 Rotationskomponente

LETRX (Anweisung)

Funktion

Weist der X-Achsen-Rotationskomponente des Typs Position einen Wert zu.

Format

LETRX <Variable vom Typ Position> = <X-Achsen-Rotationswinkel>

Erklärung

Die Anweisung beginnt mit LETRX, und die rechte Seite des Ausdrucks ist ein X-Achsenrotationswinkel. Durch diese Zuweisung wird die X-Achsen-Rotationskomponente von <Variable vom Typ Position> auf den Wert von <X-Achsenrotationswinkel> geändert.

Verwandte Begriffe

LETRY, LETRZ, LETR

Beispiel

```
DEFPOS lp1, lp2, lp3
DEFSNG lf1, lf2
LETRX lp1 = POSRX(lp3)  'Weist die X-Achsen-Rotationskomponente von lp3 der von lp1 zu.
LETRX lp2 = lf1 - lf2   'Weist den Wert von (lf1 - lf2) der X-Achsenrotationskomponente
                        'von lp2 zu.
```

LETRY (Anweisung)

Funktion

Weist der Y-Achsen-Rotationskomponente des Typs Position einen Wert zu.

Format

LETRY <Variable vom Typ Position> = <Y-Achsenrotationswinkel>

Erklärung

Die Anweisung beginnt mit LETRY, und die rechte Seite des Ausdrucks ist ein Y-Achsenrotationswinkel. Durch diese Zuweisung wird die Y-Achsen-Rotationskomponente von <Variable vom Typ Position> auf den Wert von <Y-Achsenrotationswinkel> geändert.

Verwandte Begriffe

LETRX, LETRZ, LETR

Beispiel

```
DEFPOS lp1, lp2, lp3
DEFSNG lf1, lf2
LETRY lp1 = POSRY(lp3)  'Weist die Y-Achsen-Rotationskomponente von lp3 der von lp1 zu.
LETRY lp2 = lf1 - lf2   'Weist den Wert von (lf1 - lf2) der Y-Achsenrotationskomponente
                        'von lp2 zu.
```

LETRZ (Anweisung)

Funktion

Weist der Z-Achsen-Rotationskomponente des Typs Position einen Wert zu.

Format

LETRZ <Variable vom Typ Position> = <Z-Achsenrotationswinkel>

Erklärung

Die Anweisung beginnt mit LETRZ, und die rechte Seite des Ausdrucks ist ein Z-Achsenrotationswinkel. Durch diese Zuweisung wird die Z-Achsen-Rotationskomponente von <Variable vom Typ Position> auf den Wert von <Z-Achsenrotationswinkel> geändert.

Verwandte Begriffe

LETRX, LETRY, LETR

Beispiel

```
DEFPOS lp1, lp2, lp3
DEFSNG lf1, lf2
LETRZ lp1 = POSRZ(lp3)  'Weist die Z-Achsen-Rotationskomponente von lp3 der von lp1 zu.
LETRZ lp2 = lf1 - lf2   'Weist den Wert von (lf1 - lf2) der Z-Achsenrotationskomponente
                        'von lp2 zu.
```

LETT (Anweisung)

Funktion

Weist der T-Achsenkomponente des Typs Position einen Wert zu.

Format

LETT <Variable vom Typ Position> = <T-Achsenrotationswinkel>

Erklärung

Die Anweisung beginnt mit LETT, und die rechte Seite des Ausdrucks ist ein T-Achsenrotationswinkel. Durch diese Zuweisung wird die T-Achsen-Rotationskomponente von <Variable vom Typ Position> auf den Wert von <T-Achsenrotationswinkel> geändert.

Beispiel

```
DEFPOS lp1
DEFSNG lf1, lf2
LETT lp2 = lf1 - lf2    'Weist den Wert von (lf1 - lf2) der T-Achsenrotationskomponente
                        'von lp1 zu.
```

10.7 Achsenkomponente

LETX (Anweisung) [Entspricht SLIM]

Funktion

Weist der X-Achsenkomponente des Typs Vektor/Position/Homogene Transformation einen Wert zu.

Format

LETX {<Variable vom Typ Vektor>|<Variable vom Typ Position>|<Variable vom Typ Homogene Transformation>}= <X-Achsenkomponente>

Erklärung

Die Anweisung beginnt mit LETX, und die rechte Seite des Ausdrucks ist eine X-Achsenkomponente. Durch diese Zuweisung wird die X-Achsenkomponente von <Variable vom Typ Vektor>, <Variable vom Typ Position> oder <Variable vom Typ Homogene Transformation> auf den Wert von <X-Achsen-komponente> geändert.

Verwandte Begriffe

LETP, LETY, LETZ

Beispiel

```
DEFPOS lp1, lp2
DEFVEC lv1, lv2
LETX lv1 = POSX(lv2)    'Weist die X-Achsenkomponente von lv2 der von lv1 zu.
LETX lp2 = POSX(lp2)    'Weist die X-Achsenkomponente von lp2 der von lp1 zu.
```

LETY (Anweisung) [Entspricht SLIM]

Funktion

Weist der Y-Achsenkomponente des Typs Vektor/Position/Homogene Transformation einen Wert zu.

Format

LETY {<Variable vom Typ Vektor>|<Variable vom Typ Position>|<Variable vom Typ Homogene Transformation>}= <Y-Achsenkomponente>

Erklärung

Die Anweisung beginnt mit LETY, und die rechte Seite des Ausdrucks ist eine Y-Achsenkomponente. Durch diese Zuweisung wird die Y-Achsenkomponente von <Variable vom Typ Vektor>, <Variable vom Typ Position> oder <Variable vom Typ Homogene Transformation> auf den Wert von <Y-Achsen-komponente> geändert.

Verwandte Begriffe

LETP, LETX, LETZ

Beispiel

```
DEFPOS lp1, lp2
DEFVEC lv1, lv2
LETY lv1 = POSY(lv2)    'Weist die Y-Achsenkomponente von lv2 der von lv1 zu.
LETY lp2 = POSY(lp2)    'Weist die Y-Achsenkomponente von lp2 der von lp1 zu.
```

LETZ (Anweisung) [Entspricht SLIM]

Funktion

Weist der Z-Achsenkomponente des Typs Vektor/Position/Homogene Transformation einen Wert zu.

Format

LETZ {<Variable vom Typ Vektor>|<Variable vom Typ Position>|<Variable vom Typ Homogene Transformation>}= <Z-Achsenkomponente>

Erklärung

Die Anweisung beginnt mit LETZ, und die rechte Seite des Ausdrucks ist eine Z-Achsenkomponente. Durch diese Zuweisung wird die Z-Achsenkomponente von <Variable vom Typ Vektor>, <Variable vom Typ Position> oder <Variable vom Typ Homogene Transformation> auf den Wert von <Z-Achsen-komponente> geändert.

Verwandte Begriffe

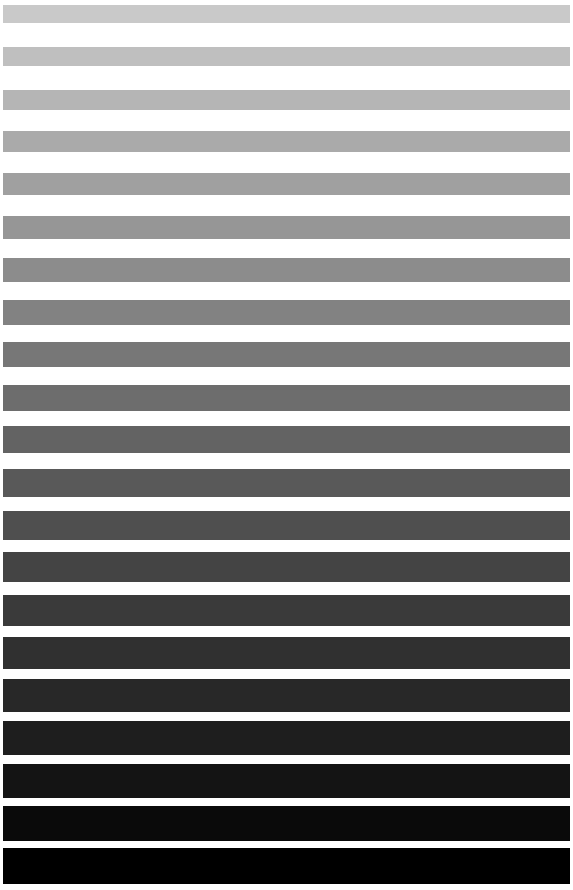
LETP, LETX, LETY

Beispiel

```
DEFPOS lp1, lp2
DEFVEC lv1, lv2
LETZ lv1 = POSZ(lv2)    'Weist die Z-Achsenkomponente von lv2 der von lv1 zu.
LETZ lp2 = POSZ(lp2)    'Weist die Z-Achsenkomponente von lp2 der von lp1 zu.
```

Kapitel 11

Flußsteuerungs-anweisungen



Eine Flußsteuerungsanweisung wird je nach Situation zum Ändern des Programmablaufs verwendet. Es gibt viele Möglichkeiten, dafür zu sorgen, daß ein Programm zum nächsten Prozeß übergeht.

11.1 Programmstop

END (Anweisung) [Entspricht SLIM]

Funktion

Deklariert das Bewegungsende über ein Programm.

Format

END

Erklärung

Die Bewegung von einem Programm wird beendet, wenn dieser Befehl ausgeführt wird.
Dies bedeutet nicht das Ende der Programmdatei.
Bei Verwendung von Unterprogrammen werden diese in manchen Fällen nach dem Hauptprogramm angeordnet. Verwenden Sie diese END-Anweisung, um das Hauptprogramm von den Unterprogrammen abzugrenzen.
Wenn keine Unterprogramme verwendet werden und keine END-Anweisung angegeben ist, wird die Systemverarbeitung fortgesetzt, als enthielte die letzte Zeile eine END-Anweisung.

Verwandte Begriffe

PROGRAM

Beispiel

```
DEFSNG lf1, lf2, lf3
PROGRAM prog1(lf1, lf2, lf3)
END                              'prog1 wird beendet.
```

STOP (Anweisung) [Entspricht SLIM]

Funktion

Beendet die Programmausführung.

Format

STOP

Erklärung

Wenn das System im Verlauf der Programmausführung eine STOP-Anweisung ausführt, endet die Programmausführung an dieser Stelle.

Verwandte Begriffe

DELAY, HALT, HOLD, STATUS

Beispiel

```
REM Führt eine Entscheidung zu einer Mehrfachbedingung aus.
SELECT CASE Index        'Der Befehl wird ausgeführt, wenn der Indexwert und die
                         'CASE-Anweisung übereinstimmen.
CASE 0                   'Wenn der Index 0 ist.
  STOP                   'Beendet ein Programm.
CASE 1                   'Wenn der Index 1 ist.
  HALT "STOP"            'Unterbricht die Programmausführung.
CASE 2                   'Wenn der Index 2 ist.
  HOLD "STOP"            'Unterbricht die Programmausführung.
CASE 3                   'Wenn der Index 3 ist.
  STOPEND                'Stoppt ein kontinuierlich ausgeführtes Programm nach einem
                         'Zyklus.
CASE 4                   'Wenn der Index 4 ist.
  ON li1 + li2 GOSUB *samp1, *samp2, *samp3
                         'Ruft ein Unterprogramm mit dem Kennungsnamen auf, das auf der
                         'gleichen Stufe wie der Wert von li1 + li2 geschrieben wurde.
CASE 5                   'Wenn der Index 5 ist.
  ON li1 + li2 GOTO *samp1, *samp2, *samp3
                         'Springt zu einer Kennung auf der gleichen Stufe wie der 'Wert
                         von li1 + li2.
CASE 6                   'Wenn der Index 6 ist.
  END                    'Deklariert das Ende der Bewegung von dem Programm.
END SELECT               'Deklariert das Ende der Entscheidungsanweisung zu einer
                         'Mehrfachbedingung.
```

STOPEND (Anweisung)

Funktion

Diese Anweisung stoppt ein kontinuierlich ausgeführtes Programm oder ein Pogramm mit einer Zyklusoption nach einem Zyklus. Wenn ein Zyklus eines Programmes mit dieser Anweisung gestartet wird, hat dies keine Auswirkung auf die Bewegung, auch wenn diese Anweisung ausgeführt wird.

Format

STOPEND

Erklärung

Wenn in einem ausgeführten Programm eine STOPEND-Anweisung gefunden wird, stoppt das System das ausgeführte Programm nach einem Zyklus.

Verwandte Begriffe

STOP, HALT, END

Beispiel

```
REM Führt eine Entscheidung zu einer Mehrfachbedingung aus.
SELECT CASE Index        'Der Befehl wird ausgeführt, wenn der Indexwert mit dem
                         'CASE-nweisungswert übereinstimmt.
  CASE 0                 'Wenn der Index 0 ist.
    STOP                 'Beendet ein Programm.
  CASE 1                 'Wenn der Index 1 ist.
    HALT "STOP"          'Unterbricht die Programmausführung.
  CASE 2                 'Wenn der Index 2 ist.
    HOLD "STOP"          'Unterbricht die Programmausführung.
  CASE 3                 'Wenn der Index 3 ist.
    STOPEND              'Stoppt ein kontinuierlich ausgeführtes Programm oder
                         'stoppt das Programm nach einem Zyklus.
  CASE 4                 'Wenn der Index 4 ist.
ON li1 + li2 GOSUB *samp1, *samp2, *samp3
                         'Ruft ein Unterprogramm mit dem Kennungsnamen auf, das
                         'auf der gleichen Stufe wie der Wert von li1 + li2
                         'geschrieben wurde.
 CASE 5                  'Wenn der Index 5 ist.
ON li1 + li2 GOTO *samp1, *samp2, *samp3
                         'Springt zu einer Kennung auf der gleichen Stufe wie der
                         'Wert von li1 + li2.
  CASE 6                 'Wenn der Index 6 ist.
    END                  'Deklariert das Ende der Bewegung von dem Programm.
END SELECT               'Deklariert das Ende der Entscheidungsanweisung zu einer
                         'Mehrfachbedingung.
```

11.2 Aufruf

CALL (Anweisung)

Funktion

Ruft ein Programm auf und führt es aus.

Format

CALL <Programmname> [(<Argument>[,<Argument>…])]

Erklärung

Eine CALL-Anweisung ruft das mit <Programmname> angegebene Programm auf und übergibt die Steuerung.
Eine END-Anweisung in einem CALL-Programm hat die gleiche Bedeutung wie eine RETURN-Anweisung in einem Unterprogramm und gibt die Steuerung an das aufrufende Programm zurück.
Geben Sie ein Argument an, um das mit <Programmname> für <Argument> angegebene Programm zu senden.
<Argument> kann durch “Aufruf mit einem Wert” (hierbei wird eine Konstante oder ein Ausdruck verwendet) oder durch “Aufruf mit Bezug” (hierbei wird eine Variable gesendet) gesendet werden. Siehe Abschnitt 2.16 “Aufruf mit einem Wert und mit Bezug” in Kapitel 1 dieses Handbuches.

1. Aufruf mit einem Wert
   Eine Konstante, ein arithmetischer Ausdruck und ein Zeichenausdruck mit einem Wert werden gesendet. Eine Variable, auch wenn sie in Klammern ( ) eingeschlossen wird, gilt als Ausdruck und kann als Wert gesendet werden.
   Für numerierte Variablen können der Typ Ganze Zahl, Real mit einfacher oder doppelter Genauigkeit oder Zeichenfolge in Klammern ( ) eingeschlossen und mit einem Wert gesendet werden.

Beispiel: Aufrufen eines Programmes von PROGRAM SUB1(AA#)

- Wenn eine Variable als Wert gesendet wird CALL SUB1((D1))
- Wenn eine Konstante gesendet wird CALL SUB1(10#)
- Wenn ein arithmetischer Ausdruck gesendet wird CALL SUB1(D1 + D2)

Anmerkung:	**Wenn ein arithmetischer Ausdruck gesendet wird, beachten Sie den Typ des Ausdruckergebnisses. Dieser Typ muß der gleiche sein wie der Typ des Arguments in dem aufgerufenen Programm.**

2. Aufruf mit Bezug
   Beim Aufruf mit Bezug kann eine Variable als Argument gesendet werden. Wenn Sie einen vollständigen Array als <Argument> angeben wollen, fügen Sie dem Array-Namen Klammern ( ) hinzu.
   Für Variablen, die durch Aufruf mit Bezug gesendet werden, ist bei einer Änderung im aufgerufenen Programm diese Änderung auch dann noch wirksam, wenn die Steuerung an das aufrufende Programm zurückgegeben wird.

Beispiel 1: Aufrufen eines Programmes von PROGRAM SUB1(AA#)

- Wenn eine lokale Variable gesendet wird 1 CALL SUB1(DD#)
- Wenn eine lokale Variable gesendet wird 2 CALL SUB1(DA)
  (DA wurde mit DEFDBL deklariert.)

Anmerkung:	**Ein Wert muß im voraus einer lokalen Variablen zugewiesen werden.**

Beispiel 2: Aufrufen des Programmes von PROGRAM SUB2(BB%(10))

- Wenn ein ganzer Array gesendet wird CALL SUB2(AB%())
  (AB%(10) wurde mit DIM deklariert.)

Anmerkung (1):	**Wenn einer Variablen kein Wert zugeordnet wurde, tritt bei der Ausführung ein Fehler auf.**
Anmerkung (2):	**Wenn der Programmname PRO <Zahl> ist, kann kein Argument gesendet werden.**
Anmerkung (3):	**Die Anzahl der Argumente auf der aufrufenden Seite muß mit der auf der aufgerufenen Seite übereinstimmen.**

Verwandte Begriffe

PROGRAM

Beispiel

```
DEFDBL ld1, ld2
CALL SUB1((ld1))              'Wenn eine Variable als Wert gesendet wird.
CALL SUB1(10#)                'Wenn eine Konstante gesendet wird.
CALL SUB1(ld1 + ld2)          'Wenn ein arithmetischer Ausdruck gesendet wird.
```

GOSUB (Anweisung) [Entspricht SLIM]

Funktion

Ruft ein Unterprogramm auf.

Format

GOSUB <Kennungsname>

Erklärung

Ruft ein über den <Kennungsnamen> angegebenes Unterprogramm auf. Weitere Unterprogramme können von einem Unterprogramm aus aufgerufen werden (verschachtelte Unterprogramme).

Verwandte Begriffe

GOTO, RETURN

Beispiel

```
DIM li1 As Integer
IF li1 = 0 THEN         'Wenn li1 0 ist.
  STOP                  'Stoppt die Programmausführung.
ELSEIF li1 = 1 THEN     'Wenn li1 1 ist.
  GOTO *samp1           'Springt zur Kennung *samp1.
  GO TO *samp2          'Springt zur Kennung *samp2.
ELSEIF li1 = 2 THEN     'Wenn li1 2 ist.
  GOSUB *samp3          'Ruft ein Unterprogramm mit dem Kennungsnamen *samp3 auf.
ELSE                    'Wenn li1 ein anderer Wert ist.
  RETURN                'Kehrt zurück zu dem Programm, das das Unterprogramm
                        'aufgerufen hat.
END IF                  'Deklariert das Ende der IF-Anweisung.
```

Anmerkungen

Verwenden Sie die RETURN-Anweisung, um die Steuerung aus dem mit GOSUB (ON-GOSUB) aufgerufenen Unterprogramm an das aufrufende Programm zurückzugeben.

ON-GOSUB (Anweisung) [Entspricht SLIM]

Funktion

Ruft bei einem bestimmten Wert eines Ausdrucks eine entsprechende Unterroutine auf.

Format

ON <Ausdruck> GOSUB <Kennungsname>[,<Kennungsname>]…

Erklärung

Diese Anweisung übergibt die Steuerung an die Stelle der Kennung, die in der gleichen Rangfolge wie der Wert von <Ausdruck> geschrieben wurde; die Programmausführung wird an dieser Stelle fortgesetzt. Die Kennungen werden von links in der Rangfolge 1, 2, … gezählt.
Wenn der Wert von <Ausdruck> eine reale Zahl ist, rundet das System die Zahl auf die höchste ganze Zahl ab, die den Wert nicht überschreitet, und verarbeitet sie.

Anmerkung: Wenn das Ergebnis eines Ausdrucks die Rangfolge überschreitet, erfolgt keine Ausführung.

Verwandte Begriffe

ON-GOTO, RETURN, SELECT CASE

Beispiel

```
REM Führt eine Entscheidung zu einer Mehrfachbedingung aus.
SELECT CASE Index    'Der Befehl wird ausgeführt, wenn der Indexwert und der Wert
                     'der CASE-Anweisung übereinstimmen.
  CASE 0             'Wenn der Index 0 ist.
    STOP             'Beendet das Programm.
  CASE 1             'Wenn der Index 1 ist.
    HALT "STOP"      'Unterbricht die Programmausführung.
  CASE 2             'Wenn der Index 2 ist.
    HOLD "STOP"      'Unterbricht die Programmausführung.
  CASE 3             'Wenn der Index 3 ist.
    STOPEND          'Stoppt ein kontinuierlich ausgeführtes Programm oder stoppt 'das
                     Programm nach einem Zyklus.
  CASE 4             'Wenn der Index 4 ist.
ON li1 + li2 GOSUB *samp1, *samp2, *samp3
                     'Ruft ein Unterprogramm mit dem Kennungsnamen auf, das auf der gleichen
                     'Stufe wie der Wert von li1 + li2 geschrieben wurde.
 CASE 5              'Wenn der Index 5 ist.
ON li1 + li2 GOTO *samp1, *samp2, *samp3
                     'Springt zu der Kennung auf der gleichen Stufe wie der Wert
                     'von li1 + li2.
  CASE 6             'Wenn der Index 6 ist.
    END              'Deklariert das Ende der Bewegung von einem Programm.
END SELECT           'Deklariert das Ende einer Entscheidungsanweisung zu einer
                     'Mehrfachbedingung.
```

Anmerkungen

Verwenden Sie die RETURN-Anweisung, um die Steuerung aus dem mit GOSUB (ON-GOSUB) aufgerufenen Unterprogramm an das aufrufende Programm zurückzugeben.

RETURN (Anweisung) [Entspricht SLIM]

Funktion

Kehrt aus einer Unterroutine zurück.

Format

RETURN

Erklärung

Diese Anweisung beendet die Ausführung einer Unterroutine, an die die Steuerung mit einer GOSUB-Anweisung übergeben wurde, und kehrt zurück zum aufrufenden Programm.

Verwandte Begriffe

GOSUB

Beispiel

```
DIM li1 As Integer
IF li1 = 0 THEN        'Wenn li1 0 ist.
  STOP                 'Stoppt die Programmausführung.
ELSEIF li1 = 1 THEN    'Wenn li1 1 ist.
  GOTO *samp1          'Springt zur Kennung *samp1.
  GO TO *samp2         'Springt zur Kennung *samp2.
ELSEIF li1 = 2 THEN    'Wenn li1 2 ist.
  GOSUB *samp3 'Ruft ein Unterprogramm mit dem Kennungsnamen *samp3 auf.
ELSE                   'Wenn li1 ein anderer Wert ist.
  RETURN               'Kehrt zurück zu dem aufrufenden Programm.
END IF                 'Deklariert das Ende der IF-Anweisung.
```

11.3 Wiederholung

DO-LOOP (Anweisung)

Funktion

Führt eine Entscheidungsiteration (Wiederholung) durch.

Format

DO [{WHILE|UNTIL}[<Bedingter Ausdruck>]]
    :
LOOP

Oder

DO
    :
LOOP [{WHILE|UNTIL}[<Bedingter Ausdruck>]]

Erklärung

DO WHILE und DO UNTIL sind Iterationen mit einer Anfangsentscheidung. LOOP WHILE und LOOP UNTIL sind Iterationen mit einer Endentscheidung. Eine WHILE-Anweisung wird wiederholt, solange ein bedingter Ausdruck erfüllt (wahr (mit Ausnahme von 0)) ist. Eine UNTIL-Anweisung wird wiederholt, bis ein bedingter Ausdruck erfüllt ist.
Wenn die rechte Seite des <bedingten Ausdrucks> ignoriert wird, ermittelt das System, ob der Wert wahr ist (außer 0).
Wenn der <bedingte Ausdruck> ignoriert wird, gilt er als wahr (außer bei 0), und die Schleife wird innerhalb einer WHILE-Anweisung zu einer Endlosschleife.
Bei der UNTIL-Anweisung wird keine Anfangsentscheidung verarbeitet, und über die Endentscheidung wird die Schleife ein Mal ausgeführt.
Es gibt auch eine WHILE-WEND-Anweisung, die dem gleichen Zweck dient wie die DO WHILE-LOOP-Schleife; es sollten jedoch nur DO WHILE-LOOP-Schleifen verwendet werden.
Es gibt auch eine REPEAT-UNTIL-Anweisung, die dem gleichen Zweck dient wie die DO-LOOP UNTIL-Schleife; es sollten jedoch nur DO-LOOP UNTIL-Schleifen verwendet werden.
WHILE und UNTIL können ignoriert werden; dies führt jedoch zu Endlosschleifen.

Verwandte Begriffe

EXIT DO, WHILE-WEND, REPEAT-UNTIL, FOR-NEXT

Beispiel

```
DEFINT li1, li2, li3, li4, li5, li6, li7, li8, li9
DO WHILE li1 > li2             'Führt eine Iteration mit Anfangsentscheidung aus.
  IF li1 = 4 THEN EXIT DO      'Beendet DO-LOOP wenn li1 = 4.
  FOR li3 = 0 TO 5             'Wiederholt den Prozeß FOR-NEXT 5 Mal.
  FOR li4 = li5 TO li6         'Wiederholt den Prozeß FOR-NEXT durch Hinzufügen
                               'von 1 zum Wert von li5 jedes Mal beim Ausführen
                               'der Wiederholung, bis li5 den Wert li6 hat.
  FOR li7 = 1 TO li8 STEP 2    'Wiederholt den Prozeß FOR-NEXT durch Hinzufügen
                               'von 2 zu einem Wert jedes Mal beim Ausführen der
                               'Wiederholung von 1, bis der Wert li8 erreicht ist.
        IF li2 = 2 THEN EXIT FOR'Beendet FOR-NEXT wenn li2 < 0 ist.
        DO WHILE li2 < li9     'Führt eine Iteration mit Anfangsentscheidung aus.
          GOSUB *samp2
          li9 = li9 + 1
        LOOP                   'Ruft eine GOSUB *samp2 Anweisung auf,
                               'bis li2 < li9 ist.
      NEXT li7                 'Wiederholung.
    NEXT                       'Wiederholung.
  NEXT                         'Wiederholung.
  li9 = 0
  DO                           'Führt eine Endentscheidungsiteration durch.
    GOSUB *samp2
    li9 = li9 + 1
  LOOP UNTIL li9 < 5           'Ruft eine GOSUB *samp2 Anweisung auf,
                               'bis li9 < 5 ist.
LOOP                           'Wiederholung.
```

EXIT DO (Anweisung)

Funktion

Erzwingt das Beenden der DO-Schleife.

Format

EXIT DO

Erklärung

Diese Anweisung erzwingt das Beenden von DO-LOOP und die Fortsetzung mit der nächsten Anweisung nach der DO-LOOP-Anweisung.

Verwandte Begriffe

DO-LOOP

Beispiel

```
DEFINT li1, li2, li3, li4, li5, li6, li7, li8, li9
DO WHILE li1 > li2              'Führt eine Iteration mit Anfangsentscheidung aus.
  IF li1 = 4 THEN EXIT DO       'Beendet DO-LOOP wenn li1 = 4.
  FOR li3 = 0 TO 5              'Wiederholt den Prozeß FOR-NEXT 5 Mal.
    FOR li4 = li5 TO li6        'Wiederholt den Prozeß FOR-NEXT durch Hinzufügen 'von
                                1 zum Wert von li5 jedes Mal bei Ausführung der
                                'Wieerholung, bis li5den Wert von li6 annimmt.
    FOR li7 = 1 TO li8 STEP 2   'Wiederholt den Prozeß FOR-NEXT durch Hinzufügen 'von
                                2 zu einem Wert jedes Mal beim Ausführen der
                                'Wiederholung von 1, bis der Wert li8 erreicht ist.
       IF li2 = 2 THEN EXIT FOR 'Beendet FOR-NEXT wenn li2 < 0 ist.
       DO WHILE li2 < li9       'Führt eine Iteration mit Anfangsentscheidung aus.
         GOSUB *samp2
         li9 = li9 + 1
       LOOP                     'Ruft eine GOSUB *samp2 Anweisung auf,
                                'bis li2 < li9 ist.
     NEXT li7                   'Wiederholung.
    NEXT                        'Wiederholung.
  NEXT                          'Wiederholung.
  li9 = 0
  DO                            'Führt eine Endentscheidungsiteration durch.
    GOSUB *samp2
    li9 = li9 + 1
  LOOP UNTIL li9 < 5            'Ruft eine GOSUB *samp2 Anweisung auf,
                                'bis li9 < 5 ist.
LOOP                            'Wiederholung.
```

FOR-NEXT (Anweisung) [Entspricht SLIM]

Funktion

Führt wiederholt eine Reihe von Anweisungen zwischen den Abschnitten FOR-NEXT aus.

Format

```
FOR <Variablenname> = <Anfangswert> TO <Endwert> [STEP <Inkrement>]
    :
NEXT [<Variablenname>]
```

Erklärung

Diese Anweisung wiederholt die Ausführung einer Reihe von Anweisungen zwischen FOR-NEXT entsprechend der Bedingung, die in der FOR-Zeile angegeben ist.
Setzen Sie den Anfangswert der mit <Variablenname> für <Anfangswert> angegebenen Variablen.

Setzen Sie den Endwert der mit <Variablenname> für <Endwert> angegebenen Variablen.

Setzen Sie den Wert für die Inkrementierung zwischen dem Anfangswert und dem Endwert für <Inkrement>. Das Inkrement 1 wird verwendet, wenn STEP ignoriert wird.

In den folgenden Fällen wird FOR-NEXT nicht ausgeführt, und die Ausführung wird mit der nächsten Anweisung nach NEXT fortgesetzt.

(1) Wenn <Inkrement> ein positiver Wert ist und <Anfangswert> größer als <Endwert>
(2) Wenn <Inkrement> ein negativer Wert ist und <Anfangswert> kleiner als <Endwert>.

<Anfangswert> wird jedoch <Variable> zugewiesen.
Sie können eine weitere FOR-NEXT-Schleife in einer FOR-NEXT-Schleife definieren (dies wird als Verschachtelung bezeichnet).
In diesem Fall muß für jeden <Variablennamen> eine andere Variable verwendet werden.
Darüber hinaus muß eine FOR-NEXT-Schleife vollständig in der anderen FOR-NEXT-Schleife enthalten sein.

Anmerkung:	**FOR und NEXT müssen paarweise angegeben werden. Wenn ein Programm in eine FOR-NEXT-Schleife springt oder aus einer solchen Schleife heraus, ist eine korrekte Verarbeitung nicht sichergestellt. Ist das Inkrement auf 0 gesetzt, wird die Schleife zu einer Endlosschleife.**

Verwandte Begriffe

DO-LOOP, EXIT FOR

Beispiel

```
DEFINT li1, li2, li3, li4, li5, li6, li7, li8, li9
DO WHILE li1 > li2              'Führt eine Iteration mit Anfangsentscheidung aus.
  IF li1 = 4 THEN EXIT DO       'Beendet DO-LOOP wenn li1 = 4.
  FOR li3 = 0 TO 5              'Wiederholt den Prozeß FOR-NEXT 5 Mal.
    FOR li4 = li5 TO li6        'Wiederholt den Prozeß FOR-NEXT durch Hinzufügen
                                'von 1 zum Wert von li5 jedes Mal beim Ausführen
                                'der Wiederholung, bis li5 den Wert von li6 annimmt.
      FOR li7 = 1 TO li8 STEP 2 'Wiederholt den Prozeß FOR-NEXT durch Hinzufügen
                                'von 2 zu einem Wert jedes Mal beim Ausführen der
                                'Wiederholung von 1, bis der Wert li8 erreicht ist.
       IF li2 = 2 THEN EXIT FOR 'Beendet FOR-NEXT wenn li2 < 0 ist.
       DO WHILE li2 < li9       'Führt eine Iteration mit Anfangsentscheidung aus.
          GOSUB *samp2
          li9 = li9 + 1
        LOOP                    'Ruft eine GOSUB *samp2 Anweisung auf,
                                'bis li2 < li9 ist.
      NEXT li7                  'Wiederholung.
    NEXT                        'Wiederholung.
  NEXT                          'Wiederholung.
  li9 = 0
  DO                            'Führt eine Endentscheidungsiteration durch.
    GOSUB *samp2
    li9 = li9 + 1
  LOOP UNTIL li9 < 5            'Ruft eine GOSUB *samp2 Anweisung auf,
                                'bis li9 < 5 ist.
LOOP                            'Wiederholung.
```

EXIT FOR (Anweisung)

Funktion

Erzwingt das Beenden des FOR-NEXT-Zweigs.

Format

EXIT FOR

Erklärung

Diese Anweisung erzwingt das Beenden von FOR-NEXT und die Fortsetzung mit der nächsten Anweisung nach der FOR-NEXT-Anweisung.

Verwandte Begriffe

FOR-NEXT

Beispiel

```
DEFINT li1, li2, li3, li4, li5, li6, li7, li8, li9
DO WHILE li1 > li2              'Führt eine Iteration mit Anfangsentscheidung aus.
  IF li1 = 4 THEN EXIT DO       'Beendet DO-LOOP wenn li1 = 4.
  FOR li3 = 0 TO 5              'Wiederholt den Prozeß FOR-NEXT 5 Mal.
    FOR li4 = li5 TO li6        'Wiederholt den Prozeß FOR-NEXT durch Hinzufügen
                                'von 1 zum 'Wert von li5, bis li5 den Wert von 'li6
                                annimmt.
      FOR li7 = 1 TO li8 STEP 2 'Wiederholt den Prozeß FOR-NEXT durch Hinzufügen
                                'von 2 zu einem Wert ab 1, bis der Wert von li8
                                'erreicht ist.
       IF li2 = 2 THEN EXIT FOR 'Beendet FOR-NEXT wenn li2 < 0 ist.
       DO WHILE li2 < li9       'Führt eine Iteration mit Anfangsentscheidung aus.
          GOSUB *samp2
          li9 = li9 + 1
        LOOP                    'Ruft eine GOSUB *samp2 Anweisung auf,
                                'bis li2 < li9 ist.
      NEXT li7                  'Wiederholung.
    NEXT                        'Wiederholung.
  NEXT                          'Wiederholung.
  li9 = 0
  DO                            'Führt eine Endentscheidungsiteration durch.
    GOSUB *samp2
    li9 = li9 + 1
  LOOP UNTIL li9 < 5            'Ruft eine GOSUB *samp2 Anweisung auf,
                                'bis li9 < 5 ist.
LOOP                            'Wiederholung.
```

REPEAT-UNTIL (Anweisung)

Funktion

Führt eine Endentscheidungsiteration (Wiederholung) durch.

Format

REPEAT
  :
UNTIL [<Bedingter Ausdruck>]

Erklärung

Diese Anweisung wiederholt Anweisungen zwischen einer REPEAT-Anweisung und einer UNTIL-Anweisung, bis <Bedingter Ausdruck> erfüllt ist.

Wenn <Bedingter Ausdruck> nicht angegeben ist, wird er als wahr betrachtet (außer bei 0), und eine Schleife wird nur ein Mal ausgeführt.
Es gibt auch eine DO-LOOP UNTIL-Anweisung, die dem gleichen Zweck dient wie die REPEAT-UNTIL-Schleife; es sollten jedoch nur DO-LOOP UNTIL-Schleifen verwendet werden.

Verwandte Begriffe

DO-LOOP, WHILE-WEND

Beispiel

```
DEFINT li1, li2, li3, li4, li5, li6, li7, li8, li9
DO WHILE li1 > li2              'Führt eine Iteration mit Anfangsentscheidung aus.
  IF li1 = 4 THEN EXIT DO       'Beendet DO-LOOP wenn li1 = 4.
  FOR li3 = 0 TO 5              'Wiederholt einen Prozeß FOR-NEXT 5 Mal.
   FOR li4 = li5 TO li6         'Wiederholt den Prozeß FOR-NEXT durch Hinzufügen
                                'von 1 zum Wert von li5 jedes Mal beim Ausführen
                                'der Wiederholung, bis li5 den Wert von li6 annimmt.
     FOR li7 = 1 TO li8 STEP 2  'Wiederholt den Prozeß FOR-NEXT durch Hinzufügen
                                'von 2 zu einem Wert jedes Mal beim Ausführen
                                'der Wiederholung von 1, bis der Wert li8 erreicht
   'ist.
      IF li2 = 2 THEN EXIT FOR  'Beendet FOR-NEXT wenn li2 < 0.
      WHILE li2 < li9           'Führt eine Iteration mit Anfangsentscheidung aus.
         GOSUB *samp2
         li9 = li9 + 1
       WEND                     'Ruft eine GOSUB *samp2 Anweisung auf, bis < li9 ist.
     NEXT li7                   'Wiederholung.
   NEXT                         'Wiederholung.
  NEXT                          'Wiederholung.
  li9 = 0
  REPEAT                        'Führt die Iteration mit Endeentscheidung aus.
    GOSUB *samp2
    li9 = li9 + 1
  UNTIL li9 < 5                 'Ruft eine GOSUB *samp2 Anweisung auf, bis li9 < 5 ist.
LOOP                            'Wiederholung.
```

WHILE-WEND (Anweisung)

Funktion

Führt eine Anfangs-Entscheidungsiteration durch.

Format

WHILE [<Bedingter Ausdruck>]
 :
WEND

Erklärung

Diese Anweisung wiederholt Anweisungen zwischen einer WHILE-Anweisung und einer WEND-Anweisung, solange <Bedingter Ausdruck> erfüllt ist.

Wenn <Bedingter Ausdruck> nicht angegeben ist, wird er als wahr betrachtet (außer bei 0), und die Schleife wird endlos ausgeführt.
Wenn eine Anweisung mit GOTO in eine WHILE-Anweisung und eine WEND-Anweisung verzweigt, führt sie normalerweise die WEND-Anweisung aus und kehrt zurück zur WHILE-Anweisung. Das Programm wird anschließend normal ausgeführt, so daß es in diese Syntax eintritt.

Es gibt auch eine DO WHILE-LOOP-Anweisung, die dem gleichen Zweck dient wie DO WHILE-WEND; es sollten jedoch nur DO WHILE-LOOP-Schleifen verwendet werden.

Verwandte Begriffe

DO-LOOP, REPEAT-UNTIL

Beispiel

```
DEFINT li1, li2, li3, li4, li5, li6, li7, li8, li9
DO WHILE li1 > li2              'Führt eine Iteration mit Anfangsentscheidung aus.
  IF li1 = 4 THEN EXIT DO       'Beendet DO-LOOP wenn li1 = 4.
  FOR li3 = 0 TO 5              'Wiederholt den Prozeß FOR-NEXT 5 Mal.
   FOR li4 = li5 TO li6         'Wiederholt den Prozeß FOR-NEXT durch Hinzufügen von 1
                                'zum Wert von li5 jedes Mal beim Ausführen
                                'der Wiederholung, bis li5
                                'den Wert von li6 annimmt.
     FOR li7 = 1 TO li8 STEP 2  'Wiederholt den Prozeß FOR-NEXT durch Hinzufügen von 2
                                'zu einem Wert jedes Mal beim Ausführen der
                                'Wiederholung von 1, bis der Wert li8 erreicht ist.
       IF li2 = 2 THEN EXIT FOR 'Beendet FOR-NEXT wenn li2 < 0 ist.
       WHILE li2 < li9          'Führt eine Iteration mit Anfangsentscheidung aus.
          GOSUB *samp2
          li9 = li9 + 1
        WEND                    'Ruft eine GOSUB *samp2 Anweisung auf,
                                'bis li2 < li9 ist.
      NEXT li7                  'Wiederholung.
    NEXT                        'Wiederholung.
  NEXT                          'Wiederholung.
  li9 = 0
  REPEAT                        'Führt eine Iteration mit Endentscheidung aus.
    GOSUB *samp2
    li9 = li9 + 1
  UNTIL li9 < 5                 'Ruft eine GOSUB *samp2 Anweisung auf,
                                'bis li9 < 5 ist.
LOOP                            'Wiederholung.
```

11.4 Bedingte Verzweigung

IF-END IF (Anweisung)

Funktion

Führt eine bedingte Entscheidung eines bedingten Ausdrucks zwischen IF-END IF aus.

Format

IF <Bedingter Ausdruck> THEN
    :
[ELSEIF <Bedingter Ausdruck> THEN]
    :
[ELSE]
    :
END IF

Erklärung

Die Ausführung eines Programmes wird über die Bedingung von <Bedingter Ausdruck> gesteuert.

Wenn <Bedingter Ausdruck> einer IF-Anweisung wahr ist (außer bei 0), werden die Anweisungen innerhalb der IF-ELSEIF-Anweisung ausgeführt. Wenn <Bedingter Ausdruck> falsch (0) ist, wird <Bedingter Ausdruck> einer ELSEIF-Anweisung entschieden. In der gleichen Weise werden ELSEIF-ELSE und ELSE-END IF ausgeführt.

Verwandte Begriffe

IF-THEN-ELSE

Beispiel

```
DIM li1 As Integer
IF li1 = 0 THEN         'Wenn li1 0 ist.
  STOP                  'Stoppt die Programmausführung.
ELSEIF li1 = 1 THEN     'Wenn li1 1 ist.
  GOTO *samp1           'Springt zur Kennung *samp1.
  GO TO *samp2          'Springt zur Kennung *samp2.
ELSEIF li1 = 2 THEN     'Wenn li1 2 ist.
  GOSUB *samp3          'Ruft ein Unterprogramm mit dem Kennungsnamen *samp3 auf.
ELSE                    'Wenn li1 ein anderer Wert ist.
  RETURN                'Kehrt zurück zu dem aufrufenden Programm.
END IF                  'Deklariert das Ende der IF-Anweisung.
```

IF-THEN-ELSE (Anweisung) [Entspricht SLIM]

Funktion

Führt eine bedingte Entscheidung eines logischen Ausdrucks aus.

Format

IF <Bedingter Ausdruck> THEN {<Anweisung>|<Kennungsname>} [ELSE {<Anweisung>|<Kennungsname>}]

Erklärung

Diese Anweisung steuert die Ausführung eines Programmes entsprechend der Bedingung von <Bedingter Ausdruck>.

Wenn <Bedingter Ausdruck> wahr ist (außer bei 0), wird der Abschnitt unter THEN ausgeführt. Wenn <Bedingter Ausdruck> falsch ist (0), wird der Abschnitt unter ELSE ausgeführt.

Verwandte Begriffe

IF-END IF

Beispiel

```
IF l0 = 0 THEN STOP ELSE GOSUB *samp1
                        'Wenn i1 0 ist, stoppt die Steuerung die Ausführung des
                        'Programmes. Hat il einen anderen Wert, ruft die Steuerung
                        'das Unterprogramm mit dem Kennungsnamen *sampl auf.
il = il + 1             'il wird hochgezählt.
END                     'Definiert das Programmende.
*sampl:                 'Definiert die Kennung des Unterprogrammes.
i0 =0                   'Ersetzt 0 für i0.
RETURN                  'Kehrt zurück zu dem aufrufenden Programm.
```

SELECT CASE (Anweisung)

Funktion

Führt eine Entscheidung zu einer Mehrfachbedingung aus.

Format

```
SELECT CASE <Ausdruck>
    CASE <Element>[,<Element>...]
        :
    [CASE ELSE]
END SELECT
```

Erklärung

Diese Anweisung führt eine Reihe von Anweisungen nach CASE aus, wenn der Wert von <Ausdruck> der Angabe <Element> in der CASE-Anweisung entspricht.

Für <Ausdruck> kann ein arithmetischer Ausdruck oder eine Zeichenfolge angegeben werden.
Für <Element> kann eine Variable, eine Konstante, ein Ausdruck oder ein bedingter Ausdruck angegeben werden.
Ein bedingter Ausdruck kann wie folgt angegeben werden.

- <Arithmetischer Ausdruck 1> TO < Arithmetischer Ausdruck 2>
  Es wird geprüft, ob das Ergebnis von <Ausdruck> gleich oder größer <Arithmetischer Ausdruck 1> ist oder gleich oder kleiner <Arithmetischer Ausdruck 2>.
- Diese Anweisung kann bei einer Zeichenfolge nicht verwendet werden.
  IS <Vergleichsoperator><Arithmetischer Ausdruck>
  Das Ergebnis von <Ausdruck> und der Wert von <Arithmetischer Ausdruck> werden verglichen.

Bei einer Zeichenfolge lautet der <Vergleichsoperator> “ = ”.
Es wird eine CASE ELSE-Anweisung ausgeführt, wenn keine der CASE-Anweisungen erfüllt ist.
Eine CASE ELSE-Anweisung muß vor einer END SELECT-Anweisung angegeben werden.

Verwandte Begriffe

IF-END IF

Beispiel

```
REM Führt eine Entscheidung zu einer Mehrfachbedingung aus.
SELECT CASE Index     'Der Befehl wird ausgeführt, wenn der Indexwert und der Wert
                      'der CASE-Anweisung übereinstimmen.
  CASE 0              'Wenn der Index 0 ist.
    STOP              'Beendet das Programm.
  CASE 1              'Wenn der Index 1 ist.
    HALT "STOP"       'Unterbricht die Programmausführung.
  CASE 2              'Wenn der Index 2 ist.
    HOLD "STOP"       'Unterbricht die Programmausführung.
  CASE 3              'Wenn der Index 3 ist.
    STOPEND           'Stoppt ein kontinuierlich ausgeführtes Programm oder stoppt 'das
                      Programm nach einem Zyklus.
  CASE 4              'Wenn der Index 4 ist.
                      ON li1 + li2 GOSUB *samp1, *samp2, *samp3
                      'Ruft ein Unterprogramm mit dem Kennungsnamen auf, das auf
                      'der gleichen Stufe wie der Wert von li1 + li2 geschrieben
                      'wurde.
  CASE 5              'Wenn der Index 5 ist.
ON li1 + li2 GOTO *samp1, *samp2, *samp3
                        'Springt zu der Kennung auf der gleichen Stufe wie der
                        'Wert von li1 + li2.
  CASE 6 TO 8           'Wenn der Index 6 bis 8 ist.
PRINTDBG "Reservierung" 'Gibt eine Meldung an das Debug-Fenster aus.
  CASE IS ≥ 9           'Wenn der Index 9 oder größer ist.
    END                 'Deklariert das Ende der Bewegung von dem Programm.
END SELECT              'Deklariert das Ende einer Entscheidungsanweisung zu
                        'einer Mehrfachbedingung.
```

11.5 Unbedingte Verzweigung

GOTO (Anweisung) [Entspricht SLIM]

Funktion

Führt eine unbedingte Programmverzweigung aus.

Format

{GOTO|GO TO}<Kennungsname>

Erklärung

Diese Anweisung springt zu der mit <Kennungsname> angegebenen Kennung und setzt dort die Ausführung fort.
Statt GOTO kann auch GO TO verwendet werden.

Verwandte Begriffe

GOSUB

Beispiel

```
DIM li1 As Integer
IF li1 = 0 THEN         'Wenn li1 0 ist.
  STOP                  'Stoppt die Programmausführung.
ELSEIF li1 = 1 THEN     'Wenn li1 1 ist.
  GOTO *samp1           'Springt zur Kennung *samp1.
  GO TO *samp2          'Springt zur Kennung *samp2.
ELSEIF li1 = 2 THEN     'Wenn li1 2 ist.
  GOSUB *samp3          'Ruft ein Unterprogramm mit dem Kennungsnamen *samp3 auf.
ELSE                    'Wenn li1 ein anderer Wert ist.
  RETURN                'Kehrt zurück zu dem aufrufenden Programm.
END IF                  'Deklariert das Ende der IF-Anweisung.
```

ON-GOTO (Anweisung) [Entspricht SLIM]

Funktion

Führt bei einem bestimmten Wert eines Ausdrucks eine unbedingte Verzweigung aus.

Format

ON <Ausdruck> GOTO <Kennungsname> [,<Kennungsname>]…

Erklärung

Diese Anweisung springt zu der Zeilennummer auf der gleichen Ebene wie der Wert von <Ausdruck> und setzt dort die Ausführung des Programmes fort. Es gilt die Rangfolge 1, 2, … von links.
Das System rundet den Wert von <Ausdruck> auf eine Ganze Zahl ab und verarbeitet sie.

Anmerkung: Wenn das Ergebnis des Ausdrucks die Rangfolge überschreitet, erfolgt keine Ausführung.

Verwandte Begriffe

ON-GOSUB

Beispiel

```
REM Führt eine Entscheidung zu einer Mehrfachbedingung aus.
SELECT CASE Index       'Der Befehl wird ausgeführt, wenn der Indexwert und der Wert
                        'der CASE-Anweisung übereinstimmen.
  CASE 0                'Wenn der Index 0 ist.
    STOP                'Beendet das Programm.
  CASE 1                'Wenn der Index 1 ist.
    HALT "STOP"         'Unterbricht die Programmausführung.
  CASE 2                'Wenn der Index 2 ist.
    HOLD "STOP"         'Unterbricht die Programmausführung.
  CASE 3                'Wenn der Index 3 ist.
    STOPEND             'Stoppt ein kontinuierlich ausgeführtes Programm oder stoppt
                        'das Programm nach einem Zyklus.
  CASE 4                'Wenn der Index 4 ist.
   ON li1 + li2 GOSUB *samp1, *samp2, *samp3
                        'Ruft ein Unterprogramm mit dem Kennungsnamen auf, das auf der
                        'gleichen Stufe wie der
                        'Wert von li1 + li2 geschrieben wurde.
 CASE 5                 'Wenn der Index 5 ist.
   ON li1 + li2 GOTO *samp1, *samp2, *samp3
                        'Springt zu einer Kennung auf der gleichen Stufe wie der Wert
                        'von li1 + li2.
  CASE 6                'Wenn der Index 6 ist.
    END                 'Deklariert das Ende der Bewegung von dem Programm.
END SELECT              'Deklariert das Ende der Entscheidungsanweisung zu einer
                        'Mehrfachbedingung.
```

11.6 Kommentar

REM (Anweisung) [Entspricht SLIM]

Funktion

Gibt einen Kommentar an.

Format

{REM|'}[<Kommentar>]

Erklärung

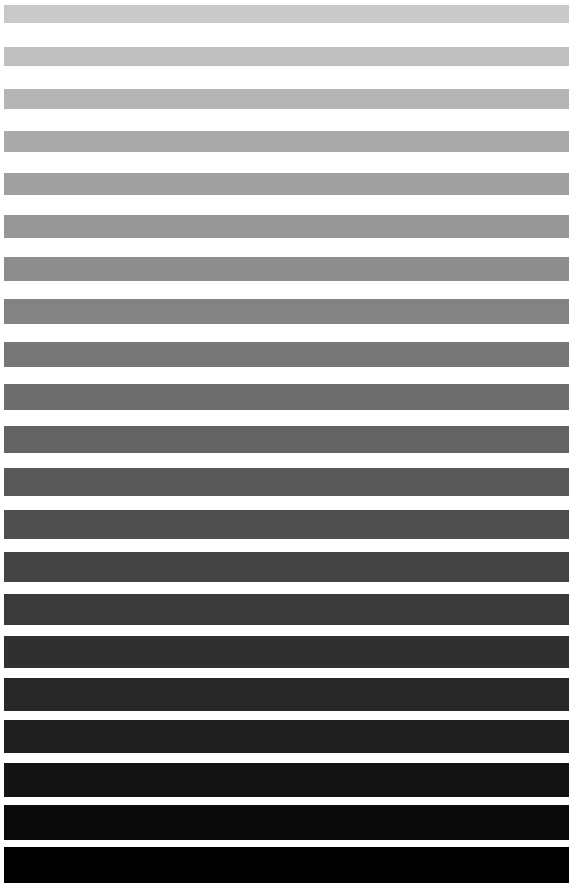
Diese Anweisung wird hinzugefügt, wenn <Kommentar> in einem Programm angegeben ist.
Eine Zeichenfolge nach REM hat keine Auswirkung auf die Programmausführung.
Statt REM kann auch ein einfaches Anführungszeichen (') verwendet werden.

Beispiel

```
REM Führt eine Entscheidung zu einer Mehrfachbedingung aus.
SELECT CASE Index       'Der Befehl wird ausgeführt, wenn der Indexwert und der Wert
                        'der CASE-Anweisung übereinstimmen.
  CASE 0                'Wenn der Index 0 ist.
    STOP                'Beendet das Programm.
  CASE 1                'Wenn der Index 1 ist.
    HALT “STOP”         'Unterbricht die Programmausführung.
  CASE 2                'Wenn der Index 2 ist.
    HOLD “STOP”         'Unterbricht die Programmausführung.
  CASE 3                'Wenn der Index 3 ist.
    STOPEND             'Stoppt ein kontinuierlich ausgeführtes Programm oder stoppt
                        'das Programm nach einem Zyklus.
  CASE 4                'Wenn der Index 4 ist.
    END                 'Deklariert das Ende der Bewegung von dem Programm.
END SELECT              'Deklariert das Ende der Entscheidungsanweisung zu einer
                        'Mehrfachbedingung.
```

Kapitel 12

Roboter-Steueranweisungen



Dieses Kapitel enthält Erläuterungen zu den Befehlen und Anweisungen zur Steuerung der Roboterbewegung.

12.1 Bewegungssteuerung

APPROACH (Anweisung)

Funktion

Führt die im Werkzeugkoordinatensystem angegebene absolute Bewegung aus.

Format

APPROACH <Interpolationsmethode>, <Basisposition>,[<Durchlaufstart-verschiebung>]<Annäherungslänge>[,<Bewegungsoption>][,NEXT]

Erklärung

Für die <Basisposition> kann der Typ Position, Gelenk oder Homogen verwendet werden.

6 Achsen Der Roboter bewegt sich an eine Position um die <Annäherungslänge> weg von der <Basisposition> in -Z-Richtung des Werkzeugkoordinatensystems.

4 Achsen Der Roboter bewegt sich an eine Position um die <Annäherungslänge> weg von der <Basisposition> in +Z-Richtung des Basiskoordinatensystems.

Für die <Interpolationsmethode> können P (oder PTP) oder L ausgewählt werden.

Interpolationsmethode	Bedeutung
P (oder PTP)	Verschieben in Punkt-zu-Punkt-Steuerung.
L	Verschieben in Linearsteuerung.

Der Wert <Durchlaufstartverschiebung> wird ausgedrückt als Radius einer Kugel mit der Zielposition als Mittelpunkt. Nach der Eingabe des Wertes der Bewegungsanweisung geht der Roboter zur nächsten Steuerung über. Geben Sie den Wert in Millimeter an. Das Ziel ist es, mit diesem Wert das Timing des Durchlaufbeginns zu ändern.
Beachten Sie auch, daß der Roboter nicht mit der nächsten Steuerung fortfährt, wenn das Ende des Armes in die Kugel eindringt.
Wenn der Wert nicht angegeben ist, wird er als Standardwert @0 verarbeitet.
Ist der Wert @0, bewegt sich der Roboter in einer Endbewegung.
Bei @P bewegt er sich in einer Durchlaufbewegung.
Bei @E prüft der Roboter die Ankunft an der Zielposition mit dem Wert des Codierers und fährt mit der nächsten Bewegung fort.
Für <Bewegungsoption> können die Optionen SPEED, ACCEL und DECEL verwendet werden.

Bewegungsoption	Bedeutung
SPEED (oder S)	Gibt die Geschwindigkeit der Bewegung an. Die Bedeutung ist die gleiche wie bei der Anweisung SPEED.
ACCEL	Gibt die Beschleunigung an. Die Bedeutung ist die gleiche wie bei der Anweisung ACCEL. Die Verzögerung kann jedoch nicht angegeben werden. Verwenden Sie die Anweisung DECEL zur Angabe der Verzögerung.
DECEL	Gibt die Verzögerung an. Die Bedeutung ist die gleiche wie bei der Anweisung DECEL.

Wenn die Option <NEXT> hinzugefügt wird, fährt der Roboter mit der nächsten Nicht-Bewegungs-Anweisung fort, ohne zu warten, bis die Bewegung abgeschlossen ist. Die folgenden Anweisungen werden jedoch nicht ausgeführt, bis die Roboterbewegung beendet ist (Durchlaufbeginn).
Roboterbewegungsanweisungen (CHANGETOOL, CHANGEWORK, SPEED, JSPEED, ACCEL, JACCEL, DECEL, JDECEL), Einstellungsbibliothek für die optimale Transportmasse (aspACLD, aspChange), Armbewegungsbibliothek (mvSetPulseWidth, etc.)
Wenn dieser Befehl zusammen mit einer Bewegungsoption verwendet wird, ist die NEXT-Option nicht mehr gültig. Wenn die Option <NEXT> hinzugefügt wird und der Schrittstop im Status “Warten auf die nächste Anweisung” ausgeführt wird, stoppt der Roboter nach dem Ende der Bewegung. Es ist daher mit Vorsicht vorzugehen, da das Stoppen einen langen Weg erfordert.

Anmerkungen:
Die Anweisung APPROACH kann mit der Anweisung MOVE neu geschrieben werden.

APPROACH <Interpolationsmethode>,[<Durchlaufstartverschiebung>]<Basisposition>,<Annäherungslänge> [,<Bewegungsoption>][,NEXT]

Die folgende Erläuterung zeigt die Anweisung, wenn die oben beschriebene APPROACH-Anweisung mit MOVE neu geschrieben wird.

6 Achsen MOVE <Interpolationsmethode>,[<Durchlaufstartverschiebung>]<Basisposition>+(0, 0, -<Annäherungslänge>)H[,<Bewegungsoption>][,NEXT]
Beispiel: APPROACH P, P3, 100
'Wie MOVE P, P3+(0, 0, -100)H.

4 Achsen MOVE <Interpolationsmethode>,[<Durchlaufstartverschiebung>]<Basisposition>+(0, 0, <Annäherungslänge>)[,<Bewegungsoption>][,NEXT]
Beispiel: APPROACH P, P3, 100
'Wie MOVE P, P3+(0, 0, 100).

Verwandte Begriffe

DEPART, SPEED

Beispiel

```
    DEFSNG lf1, lf2
    DEFPOS lp1, lp2, lp3
6 Achsen        APPROACH P, (740, 0, 480, 180, 0, 180, 5), 70
                    'Der Roboter bewegt sich (Punkt-zu-Punkt-Steuerung) zu einem
                    'Punkt 70 mm entfernt von der Position von Roboterarmstellung 5
                    'an (740, 0, 480, 180, 0, 180) in der –Zm-Richtung.
                APPROACH L, lp1, lf1, SPEED = 100
                    'Der Roboter bewegt sich (Linearsteuerung, interne
                    'Geschwindigkeit = '100 %) an eine Position mit dem Abstand lf1
                    'von der Position von lp1 in der –Zm-Richtung.
                APPROACH P, lp2, @P lf2, S = 50
                    'Der Roboter bewegt sich an eine Position im Abstand lf2 von der
                    'Position lp2 in der –Zm- Richtung (Punkt-zu-Punkt-Steuerung,
                    'interne Geschwindigkeit = '50 %) über die Durchlaufbewegung.
                APPROACH L, lp3, 80
                    'Der Roboter bewegt sich (Linearsteuerung) an eine Position
                    '80 mm entfernt von 'der Position von lp3 in der –Zm-Richtung.
```

4 Achsen

```
APPROACH P, (100, 200, 300, 45, 1), 70
                    'Bewegung an eine Position 70mm entfernt vom Punkt
                    '(100, 200, 300, 45,1) mit Haltung 1 in +Zb-Richtung
                    '(Punkt-zu-Punkt-Steuerung)
APPROACH L, lp1, lf1, SPEED=100
                    'Bewegung an eine Position lf1 entfernt von lp1 in
                    '+Zb-Richtung (Linearsteuerung, interne
                    'Geschwindigkeit=100%)
APPROACH P, lp2, @P, lf2, S=50
                    'Bewegung an eine Position lf2 entfernt von lp2 in
                    '+Zb-Richtung entlang eines Pfades (Punkt-zu-Punkt-
                    'Steuerung, interne Geschwindigkeit=50%)
APPROACH L, lp3, 80 'Bewegung an eine Position 80 mm entfernt von lp3
                    'in +Zb-Richtung (Linearsteuerung)
```

Anmerkungen

(1) Es kann vorkommen, daß die aus der Basisposition abgerufene Annäherungsposition jenseits des Bewegungsraumes des Roboters liegt. In einem solchen Fall kann ein Fehler mit der Stufe 6070 (J* Software-Bewegungsbegrenzung überschritten, Bewegungsraum überschritten, ein singulärer Punkt) auftreten.

(2) Die Armstellung der Annäherungsposition wird zu der Armstellung der Basisposition. Es kann daher vorkommen, daß die Annäherungsposition jenseits des Bewegungsraumes liegt und der “Fehler 667* Software-Bewegungsbegrenzung überschritten, jenseits Bewegungsraum 2” auftritt. Verwenden Sie in diesem Fall FIGAPRL und FIGAPRP (siehe die Abschnitte zu “FIGAPRL” und “FIGAPRP”) zum Berechnen der Armstellung für die Annäherungsposition, oder ändern Sie die Armstellung mit LETF (siehe Abschnitt 10.3 “LETF” in Kapitel 10 dieses Handbuches) durch Ersetzen mit der MOVE-Anweisung, wie in REMARKS beschrieben.

(3) Wenn bei der Linearbewegung die aktuelle Armstellung (siehe Abschnitt 4.2, “Abbildungen von Schulter, Ellbogen und Handgelenk” in Kapitel 4 des EINSTELLUNGSHANDBUCH) und die Armstellung der Basisposition unterschiedlich sind, tritt Fehler 607F (Roboterarmstellung inkonsistent) auf. Wenn der Roboter jedoch zum Ändern der Armstellung bewegt werden kann, tritt der Fehler nicht auf.

(4) Wenn bei der Linearbewegung der Bewegungspfad nahe an einem singulären Punkt (siehe Abschnitt 4.1.3 “SETTING-UP” in Kapitel 4 des EINSTELLUNGSHANDBUCHES) vorbeiführt, kann ein Fehler der Stufe 6080 “Befehlsgeschwindigkeit überschritten” auftreten, und der Roboter wird gestoppt. Verlangsamen Sie in in diesem Fall die Geschwindigkeit, oder setzen Sie 2 oder 3 für die optimale Lastkapazität (siehe Abschnitt 4.6 “Steuersätze der Bewegungsoptimierung” in Kapitel 4 dieses Handbuches). Wenn der Fehler weiterhin auftritt, vermeiden Sie den Pfad in der Nähe dieses singulären Punkts.

(5) Bei der Linearbewegung kann eine Warnung 601C (Armstellung ändern) auftreten, weil die Armstellung der Annäherungsposition und die Armstellung der Basisposition nicht übereinstimmen. Ändern Sie die Armstellung der Basisposition auf die Armstellung am Ende der Bewegung. (Beachten Sie, daß sich das Auftreten dieser Warnung nicht auf die Operation auswirkt.)

DEPART (Anweisung)

Funktion

Führt die relative Bewegung in dem Werkzeugkoordinatensystem aus.

Format

DEPART <Interpolationsmethode>,[<Durchlaufstartverschiebung>]<Fortbewegungslänge>[,<Bewegungsoption>][,NEXT]

Erklärung

6 Achsen

Der Roboter bewegt sich um den Abstand <Fortbewegungslänge> von der aktuellen Position in die –Z-Richtung des Werkzeugkoordinatensystems.

4 Achsen

Der Roboter bewegt sich um den Abstand <Fortbewegungslänge> von der aktuellen Position in die –Z-Richtung des Werkzeugkoordinatensystems.

Für die <Interpolationsmethode> können P (oder PTP) oder L ausgewählt werden.

Interpolationsmethode	Bedeutung
P (oder PTP)	Verschieben in Punkt-zu-Punkt-Steuerung.
L	Verschieben in Linearsteuerung.

Der Wert <Durchlaufstartverschiebung> wird ausgedrückt als Radius einer Kugel mit der Zielposition als Mittelpunkt. Nach der Eingabe des Wertes der Bewegungsanweisung geht der Roboter zur nächsten Steuerung über. Geben Sie den Wert in Millimeter an. Dieser Wert ändert das Timing des Durchlaufstartes. Wenn das Ende des Armes in die Kugel eintritt, fährt der Roboter nicht immer mit der nächsten Steuerung fort.
Wenn der Wert nicht angegeben ist, wird er als Standardwert @0 verarbeitet.
Ist der Wert @0, bewegt sich der Roboter in einer Endbewegung.
Bei @P bewegt er sich in einer Durchlaufbewegung.
Bei @E prüft der Roboter die Ankunft an der Zielposition mit dem Wert des Codierers und fährt mit der nächsten Bewegung fort.

Für <Bewegungsoption> stehen die Optionen SPEED, ACCEL und DECEL zur Verfügung.

Bewegungsoption	Bedeutung
SPEED (oder S)	Gibt die Geschwindigkeit der Bewegung an. Die Bedeutung ist die gleiche wie bei der Anweisung SPEED.
ACCEL	Gibt die Beschleunigung an. Die Bedeutung ist die gleiche wie bei der Anweisung ACCEL. Die Verzögerung kann jedoch nicht angegeben werden. Verwenden Sie die Anweisung DECEL zur Angabe der Verzögerung.
DECEL	Gibt die Verzögerung an. Die Bedeutung ist die gleiche wie bei der Anweisung DECEL.

Wenn die Option <NEXT> hinzugefügt wird, fährt der Roboter mit der nächsten Nicht-Bewegungs-Anweisung fort, ohne zu warten, bis die Bewegung abgeschlossen ist. Die folgenden Anweisungen werden jedoch nicht ausgeführt, bis die Roboterbewegung beendet ist (Durchlaufbeginn).
Roboterbewegungsanweisungen (CHANGETOOL, CHANGEWORK, SPEED, JSPEED, ACCEL, JACCEL, DECEL, JDECEL), Einstellungsbibliothek für die optimale Transportmasse (aspACLD, aspChange), Armbewegungsbibliothek (mvSetPulseWidth, etc.)
Wenn dieser Befehl zusammen mit der Bewegungsoption verwendet wird, ist die NEXT-Option nicht mehr gültig. Wenn die Option <NEXT> hinzugefügt wird und der Schrittstop im Status “Warten auf die nächste Anweisung” ausgeführt wird, stoppt der Roboter nach dem Ende der Bewegung. Es ist daher mit Vorsicht vorzugehen, da das Stoppen einen langen Weg erfordert.

Anmerkungen: Die Anweisung DEPART kann mit der Anweisung MOVE neu geschrieben werden.
DEPART <Interpolationsmethode>,
[<Durchlaufstartverschiebung>]
<Fortbewegungslänge>[,<Bewegungsoption>][,NEXT]
Die folgende Erläuterung zeigt die Anweisung, wenn die oben beschriebene DEPART-Anweisung mit MOVE neu geschrieben wird.

6 Achsen

MOVE<Interpolationsmethode>,[<Durchlaufstartverschiebung>]<Aktuelle Position>+(0, 0, -<Fortbewegungslänge>)H[,<Bewegungsoption>][,NEXT]

Beispiel 1: DEPART P, 70
'Wie MOVE P, P0+(0, 0, -70)H; P0 ist die aktuelle Position.
Beispiel 2: DEPART P, @P 70
'Wie MOVE P, @P P0+(0, 0, -70)H; P0 ist die aktuelle Position.

4 Achsen

MOVE<Interpolationsmethode>,[<Durchlaufstartverschiebung>]<Aktuelle Position>+(0, 0, <Fortbewegungslänge>) [,<Bewegungsoption>][,NEXT]

Beispiel 1: DEPART P, 70
'Wie MOVE P, P0+(0, 0, 70); P0 ist die aktuelle Position.
Beispiel 2: DEPART P, @P 70
'Wie MOVE P, @P P0+(0, 0, 70); P0 ist die aktuelle Position.

Verwandte Begriffe

APPROACH, SPEED

Beispiel

```
                DEFSNG lf1, lf2
6 Achsen        DEPART P, 70        'Der Roboter bewegt sich (Punkt-zu-Punkt-
                                    'Steuerung) zu einer Position 70 mm entfernt
                                    'von der aktuellen Position in der -Zm-Richtung.
                DEPART L, lf1, SPEED = 100
                                    'Der Roboter bewegt sich (Linearsteuerung,
                                    'S = 100) zu einer Position mit Abstand lf1
                                    'von der aktuellen Position in der -Zm-Richtung.
                DEPART P, lf2, S = 50'Der Roboter bewegt sich (Punkt-zu-Punkt-
                                    'Steuerung, S = 50) zu einer Position im Abstand
                                    'lf2 von der aktuellen Position in der -Zm-Richtung.
                DEPART L, 80        'Der Roboter bewegt sich (Linearsteuerung) an
                                    'eine Position 80 mm entfernt von der aktuellen
                                    'Position in der -Zm-Richtung.
4 Achsen        DEPART P, 70        'Der Roboter bewegt sich (Punkt-zu-Punkt-
                                    'Steuerung) zu einer Position 70 mm entfernt
                                    'von der aktuellen Position in der Zb-Richtung.
                DEPART L, lf1, SPEED = 100
                                    'Der Roboter bewegt sich (Linearsteuerung,
                                    'S = 100) zu einer Position mit Abstand lf1
                                    'von der aktuellen Position in der Zb- Richtung.
                DEPART P, lf2, S = 50'Der Roboter bewegt sich (Punkt-zu-Punkt-
                                    'Steuerung, S = 50) zu einer Position im Abstand
                                    'lf2 von der aktuellen Position in der Zb-Richtung.
                DEPART L, 80        'Der Roboter bewegt sich (Linearsteuerung) an
                                    'eine Position 80 mm entfernt von der von der
                                    'aktuellen Position in der Zb-Richtung.
```

Anmerkungen

Die Armstellung der DEPART-Bewegung ist die Armstellung bei Beginn der Bewegung. Es kann daher vorkommen, daß “Fehler 667* Software-Bewegungsbegrenzung überschritten, jenseits Bewegungsraum 2” auftritt, da die DEPART-Betriebsposition jenseits des Bewegungsraums liegt. Ersetzen Sie in diesem Fall die Anweisung durch die MOVE-Anweisung, wie in REMARKS beschrieben, und ändern Sie die Armstellung mit LETF (siehe Abschnitt 10.3 “Armstellung” in Kapitel 10 dieses Handbuches).

DRAW (Anweisung)

Funktion

Führt die im Arbeitskoordinatensystem angegebene relative Bewegung aus.

Format

DRAW <Interpolationsmethode>,[<Durchlaufstartverschiebung>]<Parallel-bewegungsdistanz>[,<Bewegungsoption>][,NEXT]

Erklärung

Der Roboter bewegt sich von der aktuellen Position um den Abstand <Distanz der Translationsverschiebung>.
Für die <Interpolationsmethode> können P (oder PTP) oder L ausgewählt werden.

Interpolationsmethode	Bedeutung
P (oder PTP)	Verschieben in Punkt-zu-Punkt-Steuerung.
L	Verschieben in Linearsteuerung.

Der Wert <Durchlaufstartverschiebung> wird ausgedrückt als Radius einer Kugel mit der Zielposition als Mittelpunkt. Nach der Eingabe des Wertes der Bewegungsanweisung geht der Roboter zur nächsten Steuerung über. Geben Sie den Wert in Millimeter an. Dieser Wert ändert das Timing des Durchlaufstartes. Wenn das Ende des Armes in die Kugel eintritt, fährt der Roboter nicht immer mit der nächsten Steuerung fort.
Wenn der Wert nicht angegeben ist, wird er als Standardwert @0 verarbeitet.
Ist der Wert @0, bewegt sich der Roboter in einer Endbewegung.
Bei @P bewegt er sich in einer Durchlaufbewegung.
Bei @E prüft der Roboter die Ankunft an der Zielposition mit dem Wert des Codierers und fährt mit der nächsten Bewegung fort.

Für <Bewegungsoption> stehen die Optionen SPEED, ACCEL und DECEL zur Verfügung.

Bewegungsoption	Bedeutung
SPEED (oder S)	Gibt die Geschwindigkeit der Bewegung an. Die Bedeutung ist die gleiche wie bei der Anweisung SPEED.
ACCEL	Gibt die Beschleunigung an. Die Bedeutung ist die gleiche wie bei der Anweisung ACCEL. Die Verzögerung kann jedoch nicht angegeben werden. Verwenden Sie die Anweisung DECEL zur Angabe der Verzögerung.
DECEL	Gibt die Verzögerung an. Die Bedeutung ist die gleiche wie bei der Anweisung DECEL.

Wenn die Option <NEXT> hinzugefügt wird, fährt der Roboter mit der nächsten Nicht-Bewegungs-Anweisung fort, ohne zu warten, bis die Bewegung abgeschlossen ist. Die folgenden Anweisungen werden jedoch nicht ausgeführt, bis die Roboterbewegung beendet ist (Durchlaufbeginn).
Roboterbewegungsanweisungen (CHANGETOOL, CHANGEWORK, SPEED, JSPEED, ACCEL, JACCEL, DECEL, JDECEL), Einstellungsbibliothek für die optimale Transportmasse (aspACLD, aspChange), Armbewegungsbibliothek (mvSetPulseWidth, etc.)
Wenn dieser Befehl zusammen mit der Bewegungsoption verwendet wird, ist die NEXT-Option nicht mehr gültig. Wenn die Option <NEXT> hinzugefügt wird und der Schrittstop im Status “Warten auf die nächste Anweisung” ausgeführt wird, stoppt der Roboter nach dem Ende der Bewegung. Es ist daher mit Vorsicht vorzugehen, da das Stoppen einen langen Weg erfordert.

Anmerkungen: Die Anweisung DRAW kann mit der Anweisung MOVE neu geschrieben werden.
DRAW <Interpolationsmethode>, [<Durchlaufstartverschiebung>]<Distanz der Translationsverschiebung>[,<Bewegungsoption>] [,NEXT]
Die folgende Erläuterung zeigt die Anweisung, wenn die oben beschriebene DRAW-Anweisung mit MOVE neu geschrieben wird.
MOVE <Interpolationsmethode>, [<Durchlaufstartverschiebung>]<Aktuelle Position>+<Distanz der Translationsverschiebung>

Beispiel: DRAW L, (50, 10, 50) 'Wie MOVE L, P0+(50, 10, 50)

Verwandte Begriffe

SPEED, TOOL, CHANGEWORK

Beispiel

```
DEFVEC lv1, lv2
DRAW L, (50, 10, 50)     'Der Roboter bewegt sich (Linearsteuerung) an eine
                         'Position (X = 50, Y = 10, Z = 50) entfernt von der
                         'aktuellen Position.
DRAW L, lv1, SPEED=90    'Der Roboter bewegt sich (Linearsteuerung, S = 90) an
                         'eine Position lv1 entfernt von der aktuellen Position.
DRAW L, lv2, S = 50      'Der Roboter bewegt sich (Linearsteuerung, S = 50) an
                         'eine Position lv2 entfernt von der aktuellen Position.
```

Anmerkungen

Die Armstellung der DRAW-Bewegung ist die Armstellung bei Beginn der Bewegung. Es kann daher vorkommen, daß “Fehler 667* Software-Bewegungsbegrenzung überschritten, jenseits Bewegungsraum 2” auftritt, da die DRAW-Betriebsposition jenseits des Bewegungsraums liegt. Ersetzen Sie in diesem Fall die Anweisung durch die MOVE-Anweisung, wie in REMARKS beschrieben, und ändern Sie die Armstellung mit LETF (siehe Abschnitt 10.3 “Armstellung” in Kapitel 10 dieses Handbuches).

DRIVE (Anweisung) [Entspricht SLIM]

Funktion

Führt die relative Bewegung der einzelnen Achsen aus.

Format

DRIVE[<Durchlaufstartverschiebung>](<Achsennummer>,<Relative Bewegung>)[,(<Achsennummer>,<Relative Bewegung>)…] [,<Bewegungs-option>][,NEXT]

Erklärung

Diese Anweisung verschiebt die mit <Achsennummer> angegebene Achse um den Winkel (DEG), der mit <Relative Bewegung> angegeben wurde. Wenn <Relative Bewegung> positiv ist, bewegt sich die angegebene Achse in die positive Richtung; bei einem negativen Wert dagegen in die negative Richtung. Wenn die gleiche Achse mehrmals angegeben ist, gilt die letzte Angabe.

Der Wert <Durchlaufstartverschiebung> wird ausgedrückt als Radius einer Kugel mit der Zielposition als Mittelpunkt. Nach der Eingabe des Wertes der Bewegungsanweisung geht der Roboter zur nächsten Steuerung über. Geben Sie den Wert in Millimeter an. Dieser Wert ändert das Timing des Durchlaufstartes. Wenn das Ende des Armes in die Kugel eintritt, fährt der Roboter nicht immer mit der nächsten Steuerung fort.
Wenn der Wert nicht angegeben ist, wird er als Standardwert @0 verarbeitet.
Ist der Wert @0, bewegt sich der Roboter in einer Endbewegung.
Bei @P bewegt er sich in einer Durchlaufbewegung.
Bei @E prüft der Roboter die Ankunft an der Zielposition mit dem Wert des Codierers und fährt mit der nächsten Bewegung fort.

Für <Bewegungsoption> stehen die Optionen SPEED, ACCEL und DECEL zur Verfügung.

Bewegungsoption	Bedeutung
SPEED (oder S)	Gibt die Geschwindigkeit der Bewegung an. Die Bedeutung ist die gleiche wie bei der Anweisung JSPEED.
ACCEL	Gibt die Beschleunigung an. Die Bedeutung ist die gleiche wie bei der Anweisung JACCEL. Die Verzögerung kann jedoch nicht angegeben werden. Verwenden Sie die Anweisung DECEL zur Angabe der Verzögerung.
DECEL	Gibt die Verzögerung an. Die Bedeutung ist die gleiche wie bei der Anweisung JDECEL.

Wenn die Option <NEXT> hinzugefügt wird, fährt der Roboter mit der nächsten Nicht-Bewegungs-Anweisung fort, ohne zu warten, bis die Bewegung abgeschlossen ist. Die folgenden Anweisungen werden jedoch nicht ausgeführt, bis die Roboterbewegung beendet ist (Durchlaufbeginn).
Roboterbewegungsanweisungen (CHANGETOOL, CHANGEWORK, SPEED, JSPEED, ACCEL, JACCEL, DECEL, JDECEL), Einstellungsbibliothek für die optimale Transportmasse (aspACLD, aspChange), Armbewegungsbibliothek (mvSetPulseWidth, etc.)
Wenn dieser Befehl zusammen mit der Bewegungsoption verwendet wird, ist die NEXT-Option nicht mehr gültig. Wenn die Option <NEXT> hinzugefügt wird und der Schrittstop im Status “Warten auf die nächste Anweisung” ausgeführt wird, stoppt der Roboter nach dem Ende der Bewegung. Es ist daher mit Vorsicht vorzugehen, da das Stoppen einen langen Weg erfordert.

Verwandte Begriffe

DRIVEA

Beispiel

```
DEFINT li1, li2, li3
DEFSNG lf1, lf2, lf3
DRIVE (li1, 30)              'Bewegt li1 um die Distanz von 30 Grad von der
                             'aktuellen Position.
DRIVE (li1, lf1)             'Bewegt die Achse mit dem Wert li1 von der aktuellen
                             'Position.
DRIVE (li1, 0.78RAD), (li2, lf2), (li3, lf3)
                             'Bewegt li1 um 0.78 (rad), li2 um eine Distanz von
                             'lf2 und li3 um eine Distanz von lf3 von der
                             'aktuellen Position.
```

DRIVEA (Anweisung)

Funktion

Führt die absolute Bewegung der einzelnen Achsen aus.

Format

DRIVEA[<Durchlaufstartverschiebung>] (<Achsennummer>,<Achsen-koordinate>)[,(<Achsennummer>, <Achsennummer>)…][,<Bewegungs-option>][,NEXT]

Erklärung

Diese Anweisung verschiebt die mit <Achsennummer> angegebene Achse um den Winkel (DEG), der mit <Achsenkoordinate> angegeben wurde. Wenn die gleiche Achse mehrmals angegeben ist, gilt die letzte Angabe.

Der Wert <Durchlaufstartverschiebung> wird ausgedrückt als Radius einer Kugel mit der Zielposition als Mittelpunkt. Nach der Eingabe des Wertes der Bewegungsanweisung geht der Roboter zur nächsten Steuerung über. Geben Sie den Wert in Millimeter an. Dieser Wert ändert das Timing des Durchlaufstartes. Wenn das Ende des Armes in die Kugel eintritt, fährt der Roboter nicht immer mit der nächsten Steuerung fort.
Wenn der Wert nicht angegeben ist, wird er als Standardwert @0 verarbeitet.
Ist der Wert @0, bewegt sich der Roboter in einer Endbewegung.
Bei @P bewegt er sich in einer Durchlaufbewegung.
Bei @E prüft der Roboter die Ankunft an der Zielposition mit dem Wert des Codierers und fährt mit der nächsten Bewegung fort.

Für <Bewegungsoption> stehen die Optionen SPEED, ACCEL und DECEL zur Verfügung.

Bewegungsoption	Bedeutung
SPEED (oder S)	Gibt die Geschwindigkeit der Bewegung an. Die Bedeutung ist die gleiche wie bei der Anweisung JSPEED.
ACCEL	Gibt die Beschleunigung an. Die Bedeutung ist die gleiche wie bei der Anweisung JACCEL. Die Verzögerung kann jedoch nicht angegeben werden. Verwenden Sie die Anweisung DECEL zur Angabe der Verzögerung.
DECEL	Gibt die Verzögerung an. Die Bedeutung ist die gleiche wie bei der Anweisung JDECEL.

Wenn die Option <NEXT> hinzugefügt wird, fährt der Roboter mit der nächsten Nicht-Bewegungs-Anweisung fort, ohne zu warten, bis die Bewegung abgeschlossen ist. Die folgenden Anweisungen werden jedoch nicht ausgeführt, bis die Roboterbewegung beendet ist (Durchlaufbeginn).
Roboterbewegungsanweisungen (CHANGETOOL, CHANGEWORK, SPEED, JSPEED, ACCEL, JACCEL, DECEL, JDECEL), Einstellungsbibliothek für die optimale Transportmasse (aspACLD, aspChange), Armbewegungsbibliothek (mvSetPulseWidth, etc.)
Wenn dieser Befehl zusammen mit der Bewegungsoption verwendet wird, ist die NEXT-Option nicht mehr gültig. Wenn die Option <NEXT> hinzugefügt wird und der Schrittstop im Status “Warten auf die nächste Anweisung” ausgeführt wird, stoppt der Roboter nach dem Ende der Bewegung. Es ist daher mit Vorsicht vorzugehen, da das Stoppen einen langen Weg erfordert.

Verwandte Begriffe

DRIVE

Beispiel

```
DEFINT li1, li2, li3
DEFSNG lf1, lf2, lf3
DRIVEA (li1, 30)              'Bewegt li1 um die Distanz von 30 Grad.
DRIVEA (li1, lf1)             'Bewegt die Achse mit dem Wert li1 zu dem Wert lf1
                              'von der aktuellen Position.
DRIVEA @P (li1, 0.78RAD), (li2, lf2), (li3, lf3)
                              'Bewegt li1 um 0.78 (rad), li2 um eine Distanz von
                              'lf2 und li3 um eine Distanz von lf3 von der
                              'aktuellen Position.
```

GOHOME (Anweisung) [Entspricht SLIM]

Funktion

Bewegt den Roboter an die mit der HOME-Anweisung definierte Position (die Ausgangsposition).

Format

GOHOME

Erklärung

Diese Anweisung bewegt den Roboter mit Punkt-zu-Punkt-Steuerung von der aktuellen Position in die Ausgangsposition.
Verwenden Sie die HOME-Anweisung zum Deklarieren der Ausgangsposition.
Wenn die Anweisung ausgeführt wird, ohne daß HOME gesetzt ist, tritt ein Fehler auf.

Verwandte Begriffe

HOME

Beispiel

```
GOHOME      'Der Roboter bewegt sich von der aktuellen Position in die
            'Ausgangsposition.
```

MOVE (Anweisung) [Entspricht SLIM]

Funktion

Bewegt den Roboter an die angegebene Koordinate.

Format

MOVE <Interpolationsmethode>,[@<Durchlaufstartverschiebung>]<Stellung>[,
[@<Durchlaufstartverschiebung>]<Stellung>…][,<Bewegungsoption>][,NEXT]

Erklärung

Diese Anweisung bewegt den Roboter von der aktuellen Position an die angegebene Koordinate <Stellung>.

Für <Stellung> kann der Typ Position (P), Gelenk (J) oder Homogene Transformation (T) verwendet werden.
Für den Typ Position kann eine Variable, ein Stellungs-Array mit numerierten Variablen, eine Konstante oder die aktuelle Position (*, CURPOS) verwendet werden.
Für den Typ Gelenk kann eine Variable, ein Stellungs-Array mit numerierten Variablen oder der aktuelle Winkel (CURJNT) verwendet werden.
Für den Typ Homogene Transformation kann eine Variable oder ein Stellungs-Array mit numerierten Variablen verwendet werden.
Ausdruck des Stellungs-Arrays: P[3 TO 6] From P[3] to P[6].

Für die <Interpolationsmethode> kann P, L oder C ausgewählt werden.

Interpolationsmethode	Bedeutung
P (oder PTP)	Der Roboter bewegt sich mit Punkt-zu-Punkt-Steuerung von der aktuellen Position an die angegebenen Koordinaten.
L	Der Roboter bewegt sich mit Linearsteuerung von der aktuellen Position an die angegebenen Koordinaten.
C	Der Roboter bewegt sich in eine beabsichtigte Stellung über die Durchführung einer Bogeninterpolation über eine Relaiskontaktstellung von der aktuellen Position aus. Der Roboter führt eine Interpolationsbewegung aus der Armstellung der aktuellen Position in die Armstellung der beabsichtigten Stellung aus (die Armstellung der Relaiskontaktstellung wird ignoriert). Bei der Bogeninterpolation müssen eine Relaiskontaktstellung und eine beabsichtigte Stellung angegeben werden. (Für C kann keine Stellung verwendet werden.) Auch wenn eine Durchlaufstartverschiebung zu einer Relaiskontaktstellung angegeben ist, ändert sich die Bewegung nicht. Verwenden Sie MOVE C, P1, @P P2 zur Angabe des verwendeten Durchlaufes, wenn die Bogeninterpolation abgeschlossen ist.

Der Wert <Durchlaufstartverschiebung> wird ausgedrückt als Radius einer Kugel mit den angegebenen Koordinaten als Mittelpunkt. Nach der Eingabe des Wertes der Bewegungsanweisung geht der Roboter zur nächsten Steuerung über. Geben Sie den Wert in Millimeter an. Diesen Wert ändert das Timing des Durchlaufstartes. Wenn das Ende des Armes in die Kugel eintritt, fährt der Roboter nicht immer mit der nächsten Steuerung fort.
Wenn der Wert nicht angegeben ist, wird er als Standardwert @0 verarbeitet.
Ist der Wert @0, bewegt sich der Roboter in einer Endbewegung.
Bei @P bewegt er sich in einer Durchlaufbewegung.
Bei @E prüft der Roboter die Ankunft an der Zielposition mit dem Wert des Codierers und fährt mit der nächsten Bewegung fort.

Für <Bewegungsoption> können die Optionen SPEED, ACCEL und DECEL verwendet werden.

Bewegungsoption	Bedeutung
SPEED (oder S)	Gibt die Geschwindigkeit der Bewegung an. Die Bedeutung ist die gleiche wie bei der Anweisung SPEED.
ACCEL	Gibt die Beschleunigung an. Die Bedeutung ist die gleiche wie bei der Anweisung ACCEL oder JACCEL. Die Verzögerung kann jedoch nicht angegeben werden. Verwenden Sie die Anweisung DECEL zur Angabe der Verzögerung.
DECEL	Gibt die Verzögerung an. Die Bedeutung ist die gleiche wie bei der Anweisung DECEL.

Wenn die Option <NEXT> hinzugefügt wird, fährt der Roboter mit der nächsten Nicht-Bewegungs-Anweisung fort, ohne zu warten, bis die Bewegung abgeschlossen ist. Die folgenden Anweisungen werden jedoch nicht ausgeführt, bis die Roboterbewegung beendet ist (Durchlaufbeginn).
Roboterbewegungsanweisungen (CHANGETOOL, CHANGEWORK, SPEED, JSPEED, ACCEL, JACCEL, DECEL, JDECEL), Einstellungsbibliothek für die optimale Transportmasse (aspACLD, aspChange), Armbewegungsbibliothek (mvSetPulseWidth, etc.)
Wenn dieser Befehl zusammen mit der Bewegungsoption verwendet wird, ist die NEXT-Option nicht mehr gültig. Wenn die Option <NEXT> hinzugefügt wird und der Schrittstop im Status “Warten auf die nächste Anweisung” ausgeführt wird, stoppt der Roboter nach dem Ende der Bewegung. Es ist daher mit Vorsicht vorzugehen, da das Stoppen einen langen Weg erfordert.

Ausdrücke wie MOVE P,P[3 TO 5] oder MOVE P,P3,P6,P9 haben den gleichen Ausdruck wie

```
MOVE P,P[3 TO 5] →  MOVE P, P3, NEXT
                    MOVE P, P4, NEXT
                    MOVE P, P5
```

oder

```
MOVE P,P3,P6,P9 →   MOVE P, P3, NEXT
                    MOVE P, P6, NEXT
                    MOVE P, P9
```

(Die linken Ausdrücke haben bei der Verarbeitung eine höhere Priorität.)
Daher gilt:

(1) Wenn diese Ausdrücke zwischen IOBLOCK ON und OFF stehen, wird die nächste Nicht-Bewegungs-Anweisung nicht ausgeführt, bis die Bewegung der Endstellung (P5 oder P9) beginnt. Siehe hierzu Abschnitt “IOBLOCK ON/OFF” in Kapitel 13.

(2) Wenn die Option NEXT hinzugefügt wird, wird die nächste Nicht-Bewegungs-Anweisung nicht ausgeführt, bis die Bewegung der Endstellung (P5 oder P9) beginnt.

Verwandte Begriffe

SPEED

Beispiel

```
                DIM li1 As Integer
6 Achsen        MOVE P, (740, 0, 480, 180, 0, 180, 5),NEXT
                    'Bewegt sich (mit Punkt-zu-Punkt-Steuerung) zu den Koordinaten
                    'der Roboter-Armstellung 5 mit (740, 0, 480, 180, 0, 180). Nach
                    'dem Beginn der Bewegung wird die nächste Anweisung ausgeführt.
4 Achsen        MOVE P, (100, 200, 300, 45, 1),NEXT
                    'Bewegt sich (mit Punkt-zu-Punkt-Steuerung) zu den Koordinaten
                    'der Roboter-Armstellung 1 mit (100, 200, 300, 45, 1). Nach dem
                    'Beginn der Bewegung wird die nächste Anweisung ausgeführt.
                MOVE L, lp1, SPEED = 100
                    'Bewegt sich (Linearsteuerung, interne Geschwindigkeit = 100) zu
                    'den Koordinaten von lp1.
                MOVE P, @30 lp2, lp3, S = li1
                    'Führt eine sequentielle Bewegung (Punkt-zu-Punkt-Steuerung,
                    'interne Geschwindigkeit = li1) zu den Koordinaten von lp2(@30)
                    'und lp3 durch.
                MOVE L, @20 lp4, @50 lp5, @100 lp6
                    'Führt eine sequentielle Bewegung (Linearsteuerung) zu den
                    'Koordinaten 'von lp4(@20), lp5(@50) und lp6(@100) durch.
                MOVE L, @P P[6 TO 15], lp7
                    'Führt eine sequentielle Bewegung von P[6] zu P[15] in
                    'Durchlaufbewegung durch und bewegt (Linearbewegung) sich zu den
                    'Koordinaten von lp7.
                MOVE C, lp1, @p lp2
                    'Führt eine Bogeninterpolationsbewegung über lp1 zu lp2 durch.
                    'Führt die Durchlaufbewegung an lp2 durch und fährt fort mit der
                    'nächsten Steuerung.
```

Anmerkungen

(1) Wenn eine Stellung im Typ Position und im Typ Homogene Transformation angegeben wird, geht die angegebene Stellung über den Bewegungsraum des Roboters hinaus. Als Ergebnis dieser Aktion kann ein Fehler mit Stufe 6070 (J* Software-Bewegungsbegrenzung überschritten, Bewegungsraum überschritten, ein singulärer Punkt) auftreten. Gehen Sie beim Angeben spezifischer Armstellungen von 16 bis 31 vorsichtig vor.

(2) Wenn bei der Linearbewegung oder der Bogeninterpolationsbewegung die aktuelle Armstellung (siehe Abschnitt 4.1.3 “Setting-up Guide” in Kapitel 4 des EINSTELLUNGSHANDBUCHES) und die Armstellung an den angegebenen Koordinaten falsch sind, kann Fehler 607F (Roboterarmstellung nicht kompatibel) auftreten. Es tritt jedoch kein Fehler auf, wenn der Roboter in der Lage ist, seine Armstellung zu ändern.

(3) Wenn bei der Linearbewegung und der Bogeninterpolationsbewegung der Roboterpfad nahe an einem singulären Punkt vorbeiführt, (siehe Abschnitt 4.1.3 [2] “Setting-up Guide” in Kapitel 4 des EINSTELLUNGSHANDBUCHES), tritt ein Fehler mit Stufe 6080 (Befehlsgeschwindigkeit überschritten) auf, und der Roboter wird gestoppt. Reduzieren Sie in in diesem Fall die Geschwindigkeit, oder setzen Sie 2 oder 3 für die optimale Lastkapazität (siehe Abschnitt 4.6 “Steuersätze der Bewegungsoptimierung” in Kapitel 4 dieses Handbuches). Wenn der Fehler weiterhin auftritt, vermeiden Sie bei dem Pfad die Nähe des singulären Punktes.

(4) Bei der Linearbewegung kann Warnung 601C (Armstellung ändern) auftreten, wenn eine Armstellung am Ende der Bewegung mit der angegebenen Armstellung nicht kompatibel ist. Programmieren Sie den Roboter erneut mit einer Armstellung am Ende der Bewegung (wenn die Warnung auftritt, hat sie keine Auswirkung auf die Bewegung).

(5) Für eine Durchlaufbewegung in einer Bogeninterpolationsbewegung oder einer Bewegung zur Prüfung des Codiererwertes geben Sie den Betrag für die Durchlaufstartverschiebung zu der Zielstellung an. Wenn eine Durchlaufstartverschiebung als Relaiskontaktstellung angegeben ist, ändert sich die Bewegung nicht.

(6) Die Bogeninterpolationsbewegung ignoriert die Armstellung der Relaiskontaktstellung. Damit das Werkzeugende die Relaiskontaktstellung durchlaufen kann, müssen Sie die Werkzeugkoordinaten am Werkzeugende mit der Werkzeugdefinition festlegen.

(7) Wenn bei der Bogeninterpolationsbewegung während der Durchlaufbewegung die Bewegung nach einem sofortigen Stop neu gestartet wird und die vorige Bewegung eine Durchlaufbewegung war, kann der Fehler 60de “Der Roboter führt eine Drehbewegung aus, die nicht mit der angegebenen identisch ist” auftreten.

(8) Wenn bei der Bogeninterpolationsbewegung die aktuelle Stellung und die Zielstellung identisch sind, bewegt sich der Roboter nicht. Wenn die aktuelle Stellung und die Relaiskontaktstellung oder die Relaiskontaktstellung und die Zielstellung identisch sind, bewegt sich der Roboter in einer Linearbewegung zur Zielstellung.

ROTATE (Anweisung) [Entspricht SLIM]

Funktion

Führt eine Rotationsbewegung um eine angegebene Achse aus.

Format

ROTATE <Rotationsebene>,[@<Durchlaufstartverschiebung>],<Relativer Rotationswinkel> [,[<Rotationsmittelpunkt>] [,<Rotationsoption>] [,<Bewegungsoption>][,NEXT]]

Rotationsebenenformat

6 Achsen	{{XY|YZ|ZX}|{XYH|YZH|ZXH}|(Vektor, Vektor, Vektor)}
4 Achsen	{{XY}|{XYH}}

Erklärung

Die Hand dreht sich vertikal um die in der <Rotationsebene> erstellte Achse mit dem angegebenen <Rotationswinkel>. Es gibt zwei Arten von Rotationsachsen in der Rotationsebene. Die Achse mit dem im Anfahrvektor gebildeten kleineren Winkel wird ausgewählt.
Die Rotationsebene kann aus den folgenden sieben Ebenen ausgewählt werden.

Rotationsebenenformat

6 Achsen

Wenn die Rotationsebene parallel zu den Ebenen XY, YZ und ZX der Arbeitskoordinaten liegt, geben Sie XY|YZ|ZX an.

Wenn die Rotationsebene parallel zu den Ebenen XY, YZ und ZX der Werkzeugkoordinaten liegt, geben Sie XYH|YZH|ZXH an.

Aus den drei Punkten der XYZ-Koordinaten (Vektor, Vektor, Vektor) wird eine von den Basiskoordinaten aus sichtbare Ebene erstellt.

4 Achsen

Wenn die Rotationsebene parallel zu den XY-Ebenen der Arbeitskoordinaten liegt, geben Sie XY an.

Wenn die Rotationsebene parallel zu den XY-Ebenen der Werkzeugkoordinaten liegt, geben Sie XYH an.

Aus den drei Punkten der XYZ-Koordinaten (Vektor, Vektor, Vektor) wird eine von den Basiskoordinaten aus sichtbare Ebene erstellt.

Der Wert <Durchlaufstartverschiebung> wird ausgedrückt als Radius einer Kugel mit den angegebenen Koordinaten (Pause) als Mittelpunkt. Nach der Eingabe des Wertes der Bewegungsanweisung geht der Roboter zur nächsten Steuerung über. Geben Sie den Wert in Millimeter an. Dieser Wert ändert das Timing des Durchlaufstartes. Wenn das Ende des Armes in die Kugel eintritt, fährt der Roboter nicht immer mit der nächsten Steuerung fort.
Wenn der Wert nicht angegeben ist, wird er als Standardwert @0 verarbeitet.
Ist der Wert @0, bewegt sich der Roboter in einer Endbewegung.
Bei @P bewegt er sich in einer Durchlaufbewegung.
Bei @E prüft der Roboter die Ankunft an der Zielposition mit dem Wert des Codierers und fährt mit der nächsten Bewegung fort.
Die Maßeinheit des <Relativen Rotationswinkels> ist Grad (DEG), und das Vorzeichen lautet Plus (+) für die Schraube rechts.

6 Achsen

Geben Sie für den <Rotationsmittelpunkt> einen Arbeitskoordinatenpunkt an, wenn die Rotationsebene {XY|YZ|ZX} und (Vektor, Vektor, Vektor) ist. Wenn die Rotationsebene {XYH|YXH|ZXH} ist, geben Sie einen Werkzeugkoordinatenpunkt an.

Wenn der <Rotationsmittelpunkt> nicht angegeben wird, ist der Rotationsmittelpunkt (0,0,0), wenn die Rotationsebene {XY|YZ|ZX} ist. Ist die Rotationsebene (Vektor, Vektor, Vektor), wird der erste Vektor zum Rotationsmittelpunkt. Wird jedoch eine Rotationsebene {XYH|YZH|ZXH} angegeben, kann der Roboter sich nicht um den Rotationsmittelpunkt (0, 0, 0) drehen. Stellen Sie sicher, daß Sie einen Rotationsmittelpunkt angeben.

4 Achsen

Geben Sie für den <Rotationsmittelpunkt> einen Arbeitskoordinatenpunkt an, wenn die Rotationsebene {XY} und (Vektor, Vektor, Vektor) ist. Wenn die Rotationsebene {XYH} ist, geben Sie einen Werkzeugkoordinatenpunkt an.

Wenn der <Rotationsmittelpunkt> nicht angegeben wird, ist der Rotationsmittelpunkt (0,0,0), wenn die Rotationsebene {XY} ist. Ist die Rotationsebene (Vektor, Vektor, Vektor), wird der erste Vektor zum Rotationsmittelpunkt. Wird jedoch eine Rotationsebene {XYH} angegeben, kann der Roboter sich nicht um den Rotationsmittelpunkt (0, 0, 0) drehen. Stellen Sie sicher, daß Sie einen Rotationsmittelpunkt angeben.

Geben Sie die <Rotationsoption> mit Stellung=1 oder Stellung=2 an.
Mit Stellung=1 bewegt sich der Roboter mit konstanter Haltung zum Rotationsmittelpunkt. Der Bereich des Rotationswinkels ist ±540°. Mit Stellung=2 behält der Roboter die aktuelle Haltung bei. Wenn keine Option angegeben ist, bewegt sich der Roboter in der gleichen Weise wie im Fall Stellung=2.

Für <Bewegungsoption> können die Optionen SPEED, ACCEL und DECEL verwendet werden.

Bewegungsoption	Bedeutung
SPEED (oder S)	Gibt die Geschwindigkeit der Bewegung an. Die Bedeutung ist die gleiche wie bei der Anweisung SPEED.
ACCEL	Gibt die Beschleunigung an. Die Bedeutung ist die gleiche wie bei der Anweisung ACCEL. Die Verzögerung kann jedoch nicht angegeben werden. Verwenden Sie die Anweisung DECEL zur Angabe der Verzögerung.
DECEL	Gibt die Verzögerung an. Die Bedeutung ist die gleiche wie bei der Anweisung DECEL.

Wenn die Option <NEXT> hinzugefügt wird, fährt der Roboter mit der nächsten Nicht-Bewegungs-Anweisung fort, ohne zu warten, bis die Bewegung abgeschlossen ist. Die folgenden Anweisungen werden jedoch nicht ausgeführt, bis die Roboterbewegung beendet ist (Durchlaufbeginn).
Roboterbewegungsanweisungen (CHANGETOOL, CHANGEWORK, SPEED, JSPEED, ACCEL, JACCEL, DECEL, JDECEL), Einstellungsbibliothek für die optimale Transportmasse (aspACLD, aspChange), Armbewegungsbibliothek (mvSetPulseWidth, etc.)
Wenn dieser Befehl zusammen mit der Bewegungsoption verwendet wird, ist die NEXT-Option nicht mehr gültig. Wenn die Option <NEXT> hinzugefügt wird und der Schrittstop im Status “Warten auf die nächste Anweisung” ausgeführt wird, stoppt der Roboter nach dem Ende der Bewegung. Es ist daher mit Vorsicht vorzugehen, da das Stoppen einen langen Weg erfordert.

Verwandte Begriffe

ROTATEH

Beispiel

```
                ROTATE XY,45,V1     'Der Roboter dreht sich um 45 Grad bei einer
                                    'konstanten Haltung um die Achse um Punkt V1 und
                                    'vertikal zur XY-Ebene.
Nur 6 Achsen    ROTATE(V2, V3, V4),F1
                                    'Der Roboter dreht sich um die Achse vertikal zu
                                    'der Ebene, die aus (V2, V3, V4) Durchlaufpunkt
                                    'V2 mit dem Wert von F1 erstellt wurde.
                ROTATE XYH,43,V1,S=100
                                    'Der Roboter dreht sich um einen Anfahrvektor,
                                    'der die Werkzeugkoordinaten V1 um 43 Grad mit
                                    'einer Bewegungsgeschwindigkeit von 100 % durchläuft.
```

Anmerkungen

Wenn Sie während der Durchlaufbewegung einen sofortigen Stop ausführen und das System neu starten, kann der Fehler 60de “Der Roboter führt eine Drehbewegung aus, die nicht mit der angegebenen identisch ist” auftreten.

ROTATEH (Anweisung)

Funktion

Führt eine Drehbewegung aus, indem ein Anfahrvektor als Achse verwendet wird.

Format

ROTATEH [@<Durchlaufstartverschiebung>]<Relativer Rotationswinkel um Anfahrvektor>[,<Bewegungsoption>][,NEXT]

Erklärung

Diese Anweisung dreht das Ende der Hand um den <Relativen Rotationswinkel um den Anfahrvektor>, wobei der Anfahrvektor als Achse verwendet wird.

Die Maßeinheit für <Relativer Rotationswinkel um Anfahrvektor> ist Grad (DEG). Geben Sie den Rotationswinkel in einem Bereich von -180(DEG) bis 180(DEG) an.

Der Wert <Durchlaufstartverschiebung> wird ausgedrückt als Radius einer Kugel mit den angegebenen Koordinaten (Pause) als Mittelpunkt. Nach der Eingabe des Wertes der Bewegungsanweisung geht der Roboter zur nächsten Steuerung über. Geben Sie den Wert in Millimeter an. Dieser Wert ändert das Timing des Durchlaufstartes. Wenn das Ende des Armes in die Kugel eintritt, fährt der Roboter nicht immer mit der nächsten Steuerung fort.
Wenn der Wert nicht angegeben ist, wird er als Standardwert @0 verarbeitet.
Ist der Wert @0, bewegt sich der Roboter in einer Endbewegung.
Bei @P bewegt er sich in einer Durchlaufbewegung.
Bei @E prüft der Roboter die Ankunft an der Zielposition mit dem Wert des Codierers und fährt mit der nächsten Bewegung fort.

Für <Bewegungsoption> können die Optionen SPEED, ACCEL und DECEL verwendet werden.

Bewegungsoption	Bedeutung
SPEED (oder S)	Gibt die Geschwindigkeit der Bewegung an. Die Bedeutung ist die gleiche wie bei der Anweisung SPEED.
ACCEL	Gibt die Beschleunigung an. Die Bedeutung ist die gleiche wie bei der Anweisung ACCEL. Die Verzögerung kann jedoch nicht angegeben werden. Verwenden Sie die Anweisung DECEL zur Angabe der Verzögerung.
DECEL	Gibt die Verzögerung an. Die Bedeutung ist die gleiche wie bei der Anweisung DECEL.

Wenn die Option <NEXT> hinzugefügt wird, fährt der Roboter mit der nächsten Nicht-Bewegungs-Anweisung fort, ohne zu warten, bis die Bewegung abgeschlossen ist. Die folgenden Anweisungen werden jedoch nicht ausgeführt, bis die Roboterbewegung beendet ist (Durchlaufbeginn).
Roboterbewegungsanweisungen (CHANGETOOL, CHANGEWORK, SPEED, JSPEED, ACCEL, JACCEL, DECEL, JDECEL), Einstellungsbibliothek für die optimale Transportmasse (aspACLD, aspChange), Armbewegungsbibliothek (mvSetPulseWidth, etc.)
Wenn dieser Befehl zusammen mit der Bewegungsoption verwendet wird, ist die NEXT-Option nicht mehr gültig. Wenn die Option <NEXT> hinzugefügt wird und der Schrittstop im Status “Warten auf die nächste Anweisung” ausgeführt wird, stoppt der Roboter nach dem Ende der Bewegung. Es ist daher mit Vorsicht vorzugehen, da das Stoppen einen langen Weg erfordert.

Anmerkungen: Die Anweisung ROTATEH kann mit der Anweisung MOVE neu geschrieben werden. Wenn Sie die Anweisung ROTATE von ROTATEH, [<Durchlaufstartverschiebung>] <Rotationswinkel>[, <Bewegungsoption>][NEXT] mit der MOVE-Anweisung neu schreiben, kann dies wie folgt beschrieben werden.
MOVE L, [<Durchlaufstartverschiebung>]<Aktuelle Position>+(0, 0, 0, 0, 0, <Rotationswinkel>)H[, NEXT]

Beispiel: Gleichbedeutend mit
ROTATEH @P, F1 'MOVE L, @P PO+(0, 0, 0, 0, 0, F1)H:PO ist eine aktuelle Position.

Verwandte Begriffe

ROTATE

Beispiel

```
DEFSNG FF1=50       'Setzt den relativen Rotationswinkel auf 50 Grad.
TAKEARM             'Ruft die Robotersteuerungspriorität ab.
MOVE P,P1           'Führt eine Punkt-zu-Punkt-Bewegung zum Punkt P1 durch.
CHANGETOOL 1        'Macht 1 von WERKZEUG gültig.
ROTATEH @50 FF1     'Setzt die Durchlaufstartverschiebung auf 50 und den relativen
                    'Rotationswinkel auf FF1.
CHANGETOOL 0        'Setzt WERKZEUG auf 0.
```

CURJNT (Systemvariable)

Funktion

Ruft den aktuellen Winkel des Roboters mit Typ J ab.

Format

{CURJNT | *}

Erklärung

Die von jedem Achsencodierer erkannten Gelenkwinkel des Roboters werden mit Daten des Typs J gespeichert.
Wenn der Roboter in Betrieb ist, werden die Werte bei der Ausführung dieser Anweisung abgerufen.
Verwenden Sie nach dem Abrufen der Zielstellung DESTJNT.

Verwandte Begriffe

DESTJNT, CURPOS, CURTRN

Beispiel

```
                DIM lj1 As Joint
                lj1 = CURJNT                        'Weist den aktuellen Gelenkwinkel lj1 zu.
6 Achsen        lj1 = CURJNT + (10, 10, 0, 0, 0, 10)
                                                    'Weist die aktuellen Gelenkwinkel +
                                                    '(10, 10, 0, 0, 0, 10) lj1 zu
4 Achsen        lj1 = CURJNT + (10, 10, 0, 0)
                                                    'Weist die aktuellen Gelenkwinkel +
                                                    '(10, 10, 0, 0) lj1 zu
```

Anmerkungen

Während des Betriebs mit der Maschinenverriegelungsfunktion werden die von den Codierern der einzelnen Achsen erkannten Gelenkwinkel nicht gespeichert, sondern es werden virtuelle Gelenkwinkel (Anweisungswinkel) gespeichert.

CURPOS (Systemvariable) [Entspricht SLIM]

Funktion

Ruft die aktuelle Position in dem Werkzeugkoordinatensystem mit Typ P ab.

Format

{CURPOS|*}

Erklärung

Die von den einzelnen Achsencodierern erkannten aktuellen Positionen im Werkzeugkoordinatensystem werden mit Daten des Typs P gespeichert.
Wenn der Roboter in Betrieb ist, werden die Werte bei der Ausführung dieser Anweisung abgerufen.
Verwenden Sie nach dem Abrufen der Zielstellung DESTPOS.

Verwandte Begriffe

DESTPOS, CURJNT, CURTRN

Beispiel

```
                DEFPOS lp1, lp2
                lp1 = CURPOS      'Weist die aktuelle Position im
                                  'Werkzeugkoordinatensystem lp1 zu
                lp2 = *           'Weist die aktuelle Position im
                                  'Werkzeugkoordinatensystem lp2 zu
6 Achsen        lp1 = CURPOS + (100, 200, 0, 10, 10, 0)
                                  'Weist die aktuelle Position + (100, 200, 0, 10,
                                  '10, 0) im Werkzeugkoordinatensystem lp1 zu
4 Achsen        lp1 = CURPOS + (100, 200, 0, 10)
                                  'Weist die aktuelle Position + (100, 200, 0, 10)
                                  'im Werkzeugkoordinatensystem lp1 zu
```

Anmerkungen

Während des Betriebs mit der Maschinenverriegelungsfunktion werden die von den Codierern der einzelnen Achsen erkannten Gelenkwinkel nicht gespeichert, sondern es werden virtuelle Gelenkwinkel (Anweisungswinkel) gespeichert.

CURTRN (Systemvariable) [Entspricht SLIM]

Funktion

Ruft die aktuelle Position in dem Werkzeugkoordinatensystem mit Typ T ab.

Format

{CURTRN | *}

Erklärung

Die von den einzelnen Achsencodierern erkannten aktuellen Positionen im Werkzeugkoordinatensystem werden mit Daten des Typs T gespeichert.
Wenn der Roboter in Betrieb ist, werden die Werte bei der Ausführung dieser Anweisung abgerufen.
Verwenden Sie nach dem Abrufen der Zielstellung DESTTRN.

Verwandte Begriffe

DESTTRN, CURJNT, CURPOS

Beispiel

```
DEFPOS lp1, lp2
lp1 = CURPOS      'Weist eine Position im aktuellen Werkzeugkoordinatensystem
                  'lp1 zu.
Lp2 = *           'Weist eine Position im aktuellen Werkzeugkoordinatensystem
                  'lp2 zu.
```

Anmerkungen

Während des Betriebs mit der Maschinenverriegelungsfunktion werden die von den Codierern der einzelnen Achsen erkannten Gelenkwinkel nicht gespeichert, sondern es werden virtuelle Gelenkwinkel (Anweisungswinkel) gespeichert.

DESTJNT (Systemvariable)

Funktion

Ermittelt die Zielposition der aktuellen Bewegungsanweisung mit Typ J.
Die aktuelle Position (Wert der Anweisung) wird abgerufen, wenn der Roboter stoppt.

Format

DESTJNT

Erklärung

Die Zielposition der vorherigen Bewegungsanweisung wird mit Daten des Typs J gespeichert. Da die Daten über eine Systemvariable abgerufen werden können, ist eine Übertragung der Daten an ein anderes Programm mit lokalen oder globalen Variablen nicht erforderlich.
Verwenden Sie CURJNT zum Abrufen der von den Achsencodierern erkannten Positionen.

Verwandte Begriffe

CURJNT, DESTPOS, DESTTRN

Beispiel

```
                DEFJNT lj1, lj2
                MOVE P, @P lj1, NEXT
6 Achsen        lj2 = DESTJNT+(100, 0, 0, 0, 0, 0) 'In diesem Fall ist DESTJNT = lj1.
4 Achsen        lj2 = DESTJNT+(100, 0, 0, 0)       'In diesem Fall ist DESTJNT = lj1.
```

Anmerkungen

Eine Stopposition wird abgerufen, wenn die Bewegung stoppt, nachdem die Anweisung zum Stoppen der Bewegung (siehe Abschnitt “INTERRUPT ON/OFF” in Kapitel 12 diese Handbuches) eingegeben wurde.

(Beispiel)

INTERRUPT ON
MOVE P, J1 ←Ein Unterbrechungssignal wird während der Bewegung ON.
INTERRUPT OFF
J2=DESTJNT
J1=J2 ist nicht erfüllt. J2 ist eine Stopposition.

DESTPOS (Systemvariable)

Funktion

Ermittelt die Zielposition der aktuellen Bewegungsanweisung mit Typ P.
Der aktuelle Wert (Wert der Anweisung) wird abgerufen, wenn der Roboter stoppt.

Format

DESTPOS

Erklärung

Die Zielposition der vorherigen Bewegungsanweisung wird mit Daten des Typs P gespeichert. Da die Daten über eine Systemvariable abgerufen werden können, ist eine Übertragung der Daten an ein anderes Programm mit lokalen oder globalen Variablen nicht erforderlich.
Verwenden Sie CURPOS zum Abrufen der von den Achsencodierern erkannten Positionen.

Verwandte Begriffe

CURPOS, DESTJNT, DESTTRN

Beispiel

```
                DEFPOS lp1, lp2
                MOVE P, @P lp1, NEXT
6 Achsen        lp2 = DESTPOS+(100, 10, 10, 0, 0, 0) 'In diesem Fall ist DESTPOS =
                                                      'lp1.
4 Achsen        lp2 = DESTPOS+(100, 10, 10, 0)        'In diesem Fall ist DESTPOS =
                                                      'lp1.
```

Anmerkungen

Eine Stopposition wird abgerufen, wenn die Bewegung stoppt, nachdem die Anweisung zum Stoppen der Bewegung (siehe Abschnitt “INTERRUPT ON/OFF” in Kapitel 12 diese Handbuches) eingegeben wurde.

(Beispiel)

INTERRUPT ON
MOVE P, P1 ←Ein Unterbrechungssignal wird während der Bewegung ON.
INTERRUPT OFF
P2=DESTPOS
P1=P2 ist nicht erfüllt. P2 ist eine Stopposition.

DESTTRN (Systemvariable)

Funktion

Ermittelt die Zielposition der aktuellen Bewegungsanweisung mit Typ T.
Die aktuelle Position (Wert der Anweisung) wird abgerufen, wenn der Roboter stoppt.

Format

DESTTRN

Erklärung

Die Zielposition der vorherigen Bewegungsanweisung wird mit Daten des Typs T gespeichert. Da die Daten über eine Systemvariable abgerufen werden können, ist eine Übertragung der Daten an ein anderes Programm mit lokalen oder globalen Variablen nicht erforderlich.
Verwenden Sie CURTRN zum Abrufen der von den Achsencodierern erkannten Positionen.

Verwandte Begriffe

CURTRN, DESTJNT, DESTPOS

Beispiel

```
DEFPOS lp1, lp2
MOVE P, @P lp1, NEXT
lp2 = DESTTRN+(100, 10, 10)       'In diesem Fall ist DESTTRN = lp1.
```

Anmerkungen

Wenn die Anweisung zum Stoppen der Bewegung (siehe Abschnitt “INTERRUPT ON/OFF” in Kapitel 12 diese Handbuches) eingegeben wird und die Bewegung stoppt, wird die Stopposition gespeichert.

(Beispiel) INTERRUPT ON
MOVE P, T1 ←Ein Unterbrechungssignal wird während der Bewegung ON.
INTERRUT OFF
T2=DESTTRN
T1=T2 ist nicht erfüllt. T2 ist eine Stopposition.

ARRIVE (Anweisung) [V1.2 oder neuer]

Funktion

Definiert das Bewegungsverhältnis relativ zu der programmierten vollständigen Distanz bis zum Zielpunkt, um das Programm zur Ausführung des nächsten Schrittes bereit zu machen, bis der Roboter das definierte Bewegungsverhältnis erzielt.

Format

ARRIVE <Bewegungsverhältnis>

Erklärung

Wenn ein normaler Bewegungsbefehl ausgeführt wird, kann kein anderer Befehl in einem nachfolgenden Schritt ausgeführt werden, bis die aktuelle Bewegung abgeschlossen oder die Durchlaufbewegung gestartet wurde. Wenn ein Bewegungsbefehl eine Option IOBLOCK oder NEXT enthält, kann das Programm jedoch nach der Hälfte der Ausführung des aktuellen Befehls mit dem folgenden Schritt fortfahren. Dieser ARRIVE-Befehl macht das Programm zur Ausführung des nächsten Schrittes bereit, bis der Roboter das definierte Bewegungsverhältnis erreicht hat.

<Bewegungsverhältnis> ist das Bewegungsverhältnis relativ zu der gesamten Bewegungsdistanz des Programmes entsprechend einem Bewegungsbefehl. Der Eingabebereich reicht von 1 bis 99 (%); er kann mit numerischen Angaben, ganzzahligen Variablen oder Gleitpunktvariablen angegeben werden.

Beispiel

```
PROGRAM PRO1
TAKEARM
MOVE P, P1, NEXT
ARRIVE 50           'Wenn das Bewegungsverhältnis 50% erreicht,
SET IO[240]         'IO[240] einschalten.
ARRIVE I1           'Wenn das Bewegungsverhältnis den Prozentsatz erreicht, der
RESET IO[240]       'Integer 1 zugeordnet wurde, IO[240] ausschalten.
END
```

Anmerkungen

Ein ARRIVE-Befehl definiert das Bewegungsverhältnis für den unmittelbar vorangegangenen Befehl in einer TAKEARMed-Aufgabe, die einen Arm-Semaphor abgerufen hat.

Wenn ein ARRIVE-Befehl nach dem Abrufen eines Arm-Semaphor und vor der Ausführung eines Bewegungsbefehls ausgeführt wird, führt dies zu einem Fehler 648C.

Wenn ein ARRIVE-Befehl ohne Abrufen eines Arm-Semaphor ausgeführt wird, führt dies zu einem Fehler 21F7.

Wenn der Roboter während der Ausführung eines ARRIVE-Befehls sofort stoppt und der Positionierungsfehler beim Neustart 10 mm zwischen der Stopposition und der Neustartposition überschreitet, führt dies zu einem Fehler 6486.

Wenn der Roboter während der Ausführung eines ARRIVE-Befehls sofort stoppt und beim Neustart kein Bewegungsbefehl ausgeführt wurde, erreicht der Roboter das <Bewegungsverhältnis> nicht; dies führt zu einem Fehler 6489.

Das <Bewegungsverhältnis> gibt das Verhältnis relativ zu der Bewegungsdistanz des Roboters an. Je nach den Betriebsbedingungen des Roboters (z. B. Geschwindigkeit und Beschleunigung) kann die tatsächliche Roboterposition nach Abschluß des ARRIVE-Befehls variieren. Das Ändern des <Bewegungsverhältnisses> bringt jedoch nicht unbedingt eine Änderung der tatsächlichen Roboterposition nach Abschluß des ARRIVE-Befehls mit sich. Wenn Sie den ARRIVE-Befehl verwenden und den Roboter mit den Peripheriegeräten synchronisieren wollen, überprüfen Sie das Timing.

Wenn Sie einen ARRIVE-Befehl mit einer Stromgrenzwertfunktion verwenden, kann der Roboter das <Bewegungsverhältnis> wegen seiner begrenzten Antriebsspannung nicht durchlaufen. In diesem Fall wartet das Programm unbegrenzt lange auf den Abschluß des ARRIVE-Befehls. Sie müssen das Programm stoppen und wiederaufnehmen.

Wenn Sie einen ARRIVE-Befehl mit einem INTERRUPT-Befehl verwenden, wird ein Bewegungsbefehl im Anschluß an INTERRUPT ON durch INTERRUPT SKIP-Signale nicht ausgeführt, sondern der ARRIVE-Befehl wird ausgeführt. Je nach dem Timing der INTERRUPT SKIP-Signale kann daher der ARRIVE-Befehl für einen unerwarteten Bewegungsbefehl ausgeführt werden, oder der Roboter durchläuft das <Bewegungsverhältnis> nicht. Wenn einer dieser Fälle eintritt, wartet das Programm unbegrenzt lange auf den Abschluß des ARRIVE-Befehls. Sie müssen das Programm stoppen und wiederaufnehmen.

12.2 Steuerung der Armstellung

CURFIG (Systemvariable)

Funktion

Ruft den aktuellen Wert der Roboterarmstellung ab.

Format

CURFIG

Erklärung

Die aktuelle Roboterarmstellung wird mit Ganzzahlen gespeichert. Das Verhältnis zwischen dem Wert der Ganzzahl und der Armstellung wird in der folgenden Tabelle aufgelistet.

Roboter-Armstellungen

Wert	Armstellung (Für Roboter mit 6 Achsen)	Armstellung (Für Roboter mit 4 Achsen)
0	EINFACH4–EINFACH6–KIPPEN–OBEN–RECHTS	EINFACH– RECHTS
1	EINFACH4–EINFACH6–KIPPEN–OBEN–LINKS	EINFACH– LINKS
2	EINFACH4–EINFACH6–KIPPEN–UNTEN–RECHTS	
3	EINFACH4–EINFACH6–KIPPEN–UNTEN–LINKS	
4	EINFACH4–EINFACH6–NICHT KIPPEN–OBEN–RECHTS	
5	EINFACH4–EINFACH6–NICHT KIPPEN–OBEN–LINKS	
6	EINFACH4–EINFACH6–NICHT KIPPEN–UNTEN–RECHTS	
7	EINFACH4–EINFACH6–NICHT KIPPEN–UNTEN–LINKS	
8	EINFACH4–DOPPELT6–KIPPEN–OBEN–RECHTS	DOPPELT– RECHTS
9	EINFACH4–DOPPELT6–KIPPEN–OBEN–LINKS	DOPPELT– LINKS
10	EINFACH4–DOPPELT6–KIPPEN–UNTEN–RECHTS	
11	EINFACH4–DOPPELT6–KIPPEN–UNTEN–LINKS	
12	EINFACH4–DOPPELT6–NICHT KIPPEN–OBEN–RECHTS	
13	EINFACH4–DOPPELT6–NICHT KIPPEN–OBEN–LINKS	
14	EINFACH4–DOPPELT6–NICHT KIPPEN–UNTEN–RECHTS	
15	EINFACH4–DOPPELT6–NICHT KIPPEN–UNTEN–LINKS	
16	DOPPELT4–EINFACH6–KIPPEN–OBEN–RECHTS	
17	DOPPELT4–EINFACH6–KIPPEN–OBEN–LINKS	
18	DOPPELT4–EINFACH6–KIPPEN–UNTEN–RECHTS	
19	DOPPELT4–EINFACH6–KIPPEN–UNTEN–LINKS	
20	DOPPELT4–EINFACH6–NICHT KIPPEN–OBEN–RECHTS	
21	DOPPELT4–EINFACH6–NICHT KIPPEN–OBEN–LINKS	

22	DOPPELT4–EINFACH6–NICHT KIPPEN–UNTEN–RECHTS	
23	DOPPELT4–EINFACH6–NICHT KIPPEN–UNTEN–LINKS	
24	DOPPELT4–DOPPELT6–KIPPEN–OBEN–RECHTS	
25	DOPPELT4–DOPPELT6–KIPPEN–OBEN–LINKS	
26	DOPPELT4–DOPPELT6–KIPPEN–UNTEN–RECHTS	
27	DOPPELT4–DOPPELT6–KIPPEN–UNTEN–LINKS	
28	DOPPELT4–DOPPELT6–NICHT KIPPEN–OBEN–RECHTS	
29	DOPPELT4–DOPPELT6–NICHT KIPPEN–OBEN–LINKS	
30	DOPPELT4–DOPPELT6–NICHT KIPPENVUNTEN–RECHTS	
31	DOPPELT4–DOPPELT6–NICHT KIPPEN–UNTEN–LINKS	

Verwandte Begriffe

LETF, Roboter-Armstellung (Anhang 2)

Beispiel

```
DIM li1 As Integer
li1 = CURFIG 'Weist die aktuelle Roboter-Armstellung der Variablen li1 des Typs
             'Ganze Zahl zu.
```

FIGAPRL (Funktion)

Funktion

Berechnet Armstellungen in einer Anfahrposition und in einer Standardposition, die für die Linearbewegung verfügbar sind.

Format

FIGAPRL (<Bezugsstellung>, <Annäherungslänge>)

Erklärung

Für <Bezugsstellung> können Sie nur einen Positionstyp verwenden.

6 Achsen

Berechnen Sie die Armstellung von <Bezugsstellung>, die eine Linearbewegung von der Position (Annäherungsposition) weg von <Annäherungslänge> zu <Bezugsstellung> in -Z-Richtung des Handkoordinatensystems ausführen kann.
Das System sucht automatisch eine für die Linearbewegung verfügbare Armstellung von der Annäherungsposition zu <Bezugsstellung> in der folgenden Verarbeitungsreihenfolge, bis die Bewegung verfügbar wird.

(1) Die Armstellung von <Bezugsstellung> (Wenn die Armstellung von <Bezugsstellung> nicht definiert ist, wird die aktuelle Armstellung verwendet.)
(2) Austauschen der Handgelenksstellung (Kippen/Nicht kippen).
(3) Austauschen einer Ellbogenstellung (Oben-Unten) mit einer Handgelenksstellung “Nicht Kippen”.
(4) Einstellen der Handgelenksstellung auf “Kippen”.
(5) Austauschen der Armstellung (links-rechts) mit der Ellbogenstellung “Oben” und mit der Handgelenksstellung “Nicht Kippen”.
(6) Einstellen der Handgelenksstellung auf “Kippen”.
(7) Einstellen der Ellbogenstellung auf “Unten” und der Handgelenksstellung auf “Nicht Kippen”.
(8) Einstellen der Handgelenksstellung auf “Kippen”.

Wenn sich das System nach dem achten Suchvorgang nicht mehr bewegen kann, kann ein Fehler 6070 (J* Softwarebewegungsbegrenzung überschritten) auftreten. Ändern Sie in diesem Fall die aktuelle Position oder die Annäherungslänge.

4 Achsen

Berechnen Sie die Armstellung von <Bezugsstellung>, die eine Linearbewegung von der Position (Annäherungsposition) weg von <Annäherungslänge> zu <Bezugsstellung> in -Z-Richtung des Basiskoordinatensystems ausführen kann.
Das System sucht automatisch eine für die Linearbewegung verfügbare Armstellung von der Annäherungsposition zu <Bezugsstellung> in der folgenden Verarbeitungsreihenfolge, bis die Bewegung verfügbar wird.

(1) Die Armstellung von <Bezugsstellung> (Wenn die Armstellung von <Bezugsstellung> nicht definiert ist, wird die aktuelle Armstellung verwendet.)

Verwandte Begriffe

FIGAPRP, APPROACH

Beispiel

6/4 Achsen

```
I1=FIGAPRL(P1, 100.0)
LETF P1,  I1
APPROACH  P, P1, 100.0
MOVE  L, P1
```

6 Achsen

```
I1=FIGAPRL(P1 + (100.200, 0, 0, 10, 0), 100)
LETF  P1,  I1
APPROACH  P, P1, 100.0
MOVE  L, P1
```

4 Achsen

```
I1=FIGAPRL(P1 + (100.200, 0, 0), 100)
LETF  P1,  I1
APPROACH  P, P1, 100.0
MOVE  L, P1
```

Anmerkungen

(1) Verwenden Sie FIGAPRL, wenn die Bewegungsaktion nach der Annäherung eine Linearbewegung ist. Ist es dagegen eine Punkt-zu-Punkt-Bewegung, verwenden Sie FIGAPRP.

(2) <Bezugsstellung> und die in FIGAPRL verwendete Annäherungslänge sollten konsistent sein mit <Bezugsstellung> und der in der APPROACH-Bewegung angegebenen Annhäherungslänge. Wenn <Bezugsstellung> oder Annäherungslänge unterschiedlich sind, kann bei der Linearbewegung nach der Annäherung ein Fehler der Stufe 6070 auftreten.

(3) Auch wenn Sie mit FIGAPRL eine Armstellung abgerufen haben, stoppt der Roboter eventuell bei der Linearbewegung nach der Annäherung, und ein Fehler der Stufe 6080 (Befohlene Geschw.begr. überschr.) tritt auf. Reduzieren Sie in in diesem Fall die Geschwindigkeit, oder setzen Sie 2 oder 3 für die optimale Lastkapazität (siehe Abschnitt 4.6 “Steuersätze der Bewegungsoptimierung” in Kapitel 4 dieses Handbuches). Wenn weiterhin ein Fehler auftritt, passen Sie die Annäherungslänge an.

FIGAPRP (Funktion)

Funktion

Berechnet eine Anfahrposition, bei der die Punkt-zu-Punkt-Bewegung verfügbar ist, und eine Armstellung einer Referenzposition.

Format

FIGAPRP (<Bezugsstellung>, <Annäherungslänge>)

Erklärung

Für <Bezugsstellung> können Sie nur einen Positionstyp verwenden.

6 Achsen

Berechnen Sie die Armstellung von <Bezugsstellung>, die eine Punkt-zu-Punkt-Bewegung von der Position (Annäherungsposition) weg von <Annäherungslänge> zu <Bezugsstellung> in -Z-Richtung des Handkoordinatensystems ausführen kann.
Das System sucht automatisch eine für die Punkt-zu-Punkt-Bewegung verfügbare Armstellung von der Annäherungsposition zu <Bezugsstellung> in der folgenden Verarbeitungsreihenfolge, bis die Bewegung verfügbar wird.

(1) Die Armstellung von <Bezugsstellung> (Wenn die Armstellung von <Bezugsstellung> nicht definiert ist, wird die aktuelle Armstellung verwendet.)
(2) Austauschen der Handgelenksstellung (Kippen-Nicht Kippen); wenn der Bewegungswinkel der 4. Achse von der Annäherungsposition zu <Bezugsstellung> innerhalb von ±90 Grad liegt, wird die Bewegung als verfügbar angenommen.

Wenn der Bewegungswinkel der 4. Achse ±90 Grad überschreitet oder die Bewegung nicht verfügbar ist, kann ein Fehler der Stufe 6070 (J* Softwarebewegungsbegrenzung überschritten) auftreten. Ändern Sie in diesem Fall die aktuelle Position oder die Annäherungslänge.

4 Achsen

Berechnen Sie die Armstellung von <Bezugsstellung>, die eine Punkt-zu-Punkt-Bewegung von der Position (Annäherungsposition) weg von <Annäherungslänge> zu <Bezugsstellung> in -Z-Richtung des Basiskoordinatensystems ausführen kann.
Das System sucht automatisch eine für die Punkt-zu-Punkt-Bewegung verfügbare Armstellung von der Annäherungsposition zu <Bezugsstellung> in der folgenden Verarbeitungsreihenfolge, bis die Bewegung verfügbar wird.

(1) Die Armstellung von <Bezugsstellung> (Wenn die Armstellung von <Bezugsstellung> nicht definiert ist, wird die aktuelle Armstellung verwendet.)

Verwandte Begriffe

FIGAPRL, APPROACH

Beispiel

```
I1 = FIGAPRP(P1, 100.0)
LETF  P1, I1
APPROACH  P, P1, 100.0
MOVE  P, P1
```

6 Achsen
```
I1 = FIGAPRP(P1 + (100, 200, 0, 0, 0, 10), 100, 0)
```

4 Achsen
```
I1 = FIGAPRP(P1 + (100, 200, 0, 0), 100, 0)
```

Anmerkungen

(1) Verwenden Sie FIGAPRP, wenn die Bewegungsaktion nach der Annäherung eine Punkt-zu-Punkt-Bewegung ist. Ist es dagegen eine Linearbewegung, verwenden Sie FIGAPRL.
(2) <Bezugsstellung> und die in FIGAPRP verwendete Annäherungslänge sollten konsistent sein mit <Bezugsstellung> und der in der APPROACH-Bewegung angegebenen Annhäherungslänge. Wenn <Bezugsstellung> oder Annäherungslänge unterschiedlich sind, kann bei der Punkt-zu-Punkt-Bewegung nach der Annäherung ein Fehler der Stufe 6070 auftreten.

12.3 Steuerungsanweisung für Stop

HOLD (Anweisung) [Entspricht SLIM]

Funktion

Hält die Programmverarbeitung für eine bestimmte Zeit an.

Format

HOLD <Meldung>

Erklärung

Diese Anweisung hält die Programmverarbeitung an und gibt den in <Meldung> angegebenen Inhalt an das Programmiergerät aus. Erläuterungen zum Inhalt von <Meldung> finden Sie bei der Beschreibung der Anweisung PRINTMSG.
Wenn Sie das System neu starten, führt es das unterbrochene Programm ab der nächsten Anweisung nach der HOLD-Anweisung aus.
Die Anzahl der Zeichen für eine Meldung beträgt 60 Zeichen (Ein-Byte-Zeichen).

Verwandte Begriffe

DELAY, HALT, STOP

Beispiel

```
DEFINT li1, li2
HOLD li1 + li2     'Hält die Programmverarbeitung an und setzt das System auf den
                   'Stop-Status im Schrittmodus und gibt den Wert von (li1 + li2) an das
                   'Programmiergerät aus.
HOLD "stop"        'Hält die Programmverarbeitung an und setzt das System auf den
                   'Stop-Status im Schrittmodus und gibt die Zeichenfolge "stop" an das
                   'Programmiergerät aus.
```

Anmerkungen

Wenn Sie eine Variable des Typs F und eine Variable des Typs D als Meldung verwenden, zeigt das System bis zu 4 Ziffern nach dem Dezimalpunkt an.

HALT (Anweisung) [Entspricht SLIM]

Funktion

Stoppt die Ausführung eines Programmes.

Format

HALT <Meldung>

Erklärung

Diese Anweisung stoppt die Programmausführung und gibt den in <Meldung> angegebenen Inhalt an das Programmiergerät aus. Erläuterungen zum Inhalt von <Meldung> finden Sie bei der Beschreibung der Anweisung PRINTMSG.

Wenn Sie das System neu starten, wird das Programm wieder von der ersten Zeile an ausgeführt.

Die maximale Anzahl von Zeichen für eine Meldung beträgt 60.

Verwandte Begriffe

DELAY, HOLD, STOP

Beispiel

```
DEFINT li1, li2
HALT li1 + li2     'Stoppt das Programm und gibt den Wert (li1 + li2) an das
                   'Programmiergerät aus.
HALT "end"         'Stoppt das Programm und gibt die Zeichenfolge "end" an das
                   'Programmiergerät aus.
```

Anmerkungen

Wenn Sie eine Variable des Typs F und eine Variable des Typs D verwenden, zeigt das System nur 4 Ziffern nach dem Dezimalpunkt an.

INTERRUPT ON/OFF (Anweisung)

Funktion

Unterbricht eine Roboterbewegung.

Format

INTERRUPT {ON|OFF}

Erklärung

INTERRUPT ON und INTERRUPT OFF werden als Paar verwendet; wenn während der Anweisungsausführung in dem Bereich zwischen diesen beiden Markierungen ein Interruption Skip-Signal ON wird, fährt das System mit dem nächsten Schritt nach der Unterbrechung der Bewegungsanweisung fort.
Sofern Sie das System nicht durch die Ausführung von INTERRUPT ON wieder unterbrechungsfähig machen, wird die ausgeführte Bewegungsanweisung nicht beendet, auch wenn ein spezifischer E/A-Port ON wird.
Auch wenn der Zweck bereits erreicht ist, wird eine Unterbrechung nach Ausführung der Anweisung INTERRUPT ON gültig, sofern Sie nicht eine Anweisung INTERRUPT OFF ausführen.
Wenn Sie ein Programm im Status INTERRUPT ON beenden, setzt das System automatisch INTERRUPT OFF.
Wenn Sie eine Anweisung GIVEARM auführen wie in Abschnitt 7.3.2 beschrieben, setzt das System automatisch INTERRUPT OFF.

Verwandte Begriffe

Beispiel

```
DIM lp1 As Position
INTERRUPT ON      'Fährt mit dem nächsten Schritt nach der Unterbrechung einer ausgeführten
                  'Bewegungsanweisung fort, wenn das Unterbrechungssignal des speziellen
                  'E/A-Ports eingeschaltet (ON) wird.
MOVE P, lp1
INTERRUPT OFF
```

Anmerkungen

(1) Wenn Sie eine relative Bewegungsanweisung direkt nach der Anweisung INTERRUPT OFF ausführen, wird die Bewegung eine relative ab der Stop-Position. Verwenden Sie eine absolute Bewegungsanweisung für Bewegungsanweisungen nach der Anweisung INTERRUPT OFF.
Das System führt keine Durchlaufbewegung aus, während INTERRUPT ON ist. Wenn das System feststellt, daß Durchlaufbewegungen angegeben wurden, werden alle Durchlaufbewegungen in der Endbewegung ausgeführt.

(2) Beachten Sie, daß die INTERRUPT-Anweisung nicht nur für aufrufende Programme gültig ist, sondern auch für andere Programme.

(3) INTERRUPT ist für alle Programme gültig.
Nachdem ein Programm INTERRUPT ON schaltet, ist es für alle Programme gültig.
Wenn Sie INTERRUPT OFF schalten, muß das Programm nicht immer dasjenige sein, das INTERRUPT ON geschaltet hat.

```
Beispiel
PRO1
INTERRUPT ON
END
PRO2
INTERRUPT OFF
END
```

Wie in diesem Beispiel, muß nicht immer das gleiche Programm INTERRUPT ON und OFF schalten. Nachdem das System PRO1 ausgeführt hat, ist INTERRUPT für alle Programme gültig, bis PRO2 ausgeführt wird.

12.4 Steuerung der Geschwindigkeit

SPEED (Anweisung) [Entspricht SLIM]

Funktion

Gibt die interne Geschwindigkeit der Hand an.

Format

SPEED <Bewegungsgeschwindigkeit>

Erklärung

Diese Anweisung gibt die maximale interne Geschwindigkeitsrate (%) der Hand zur <Bewegungsgeschwindigkeit> im Bereich von 0,1 bis 100 an.

ANMERKUNG: Bei einer Angabe unter 0,1 tritt zwar kein Fehler auf, aber die tatsächliche Geschwindigkeit kann von der Angabe abweichen. Bei Angabe von 0 oder weniger tritt ein Fehler auf.

Der tatsächliche Wert für die Geschwindigkeit ist (externe × interne ÷ 100).

Bei der Einstellung von SPEED werden auch ACCEL und DECEL geändert.

	Linearsteuerung	**Punkt-zu-Punkt-Steuerung**
Geschwindigkeitseinstellung	SPEED	JSPEED
Beschleunigungseinstellung	ACCEL	JACCEL
Verzögerungseinstellung	DECEL	JDECEL
Aktuelle Geschwindigkeit abrufen	CURSPD	CURJSPD
Aktuelle Beschleunigung abrufen	CURACC	CURJACC
Aktuelle Verzögerung abrufen	CURDEC	CURJDEC

Verwandte Begriffe

ACCEL, DECEL, JSPEED, MPS

Beispiel

```
DIM li1 As Integer
SPEED 100       'Legt die Bewegungsgeschwindigkeit der Hand auf 100 fest.
SPEED li1/100   'Legt die Bewegungsgeschwindigkeit der Hand auf den Wert von
                '(li1/100) fest.
```

JSPEED (Anweisung)

Funktion

Gibt die Bewegungsgeschwindigkeit der Roboterachse an.

Format

JSPEED <Bewegungsgeschwindigkeit>

Erklärung

Diese Anweisung gibt die maximale interne Rate der Bewegungsgeschwindigkeit (%) einer Achse zur <Bewegungsgeschwindigkeit> im Bereich von 0,1 bis 100 an.

ANMERKUNG: Bei einer Angabe unter 0,1 tritt zwar kein Fehler auf, aber die tatsächliche Geschwindigkeit kann von der Angabe abweichen. Bei Angabe von 0 oder weniger tritt ein Fehler auf.

Der tatsächliche Wert für die Geschwindigkeit ist (externe × interne ÷ 100).

Bei der Einstellung von JSPEED werden auch JACCEL und JDECEL geändert.

Verwandte Begriffe

CURJSPD, SPEED

Beispiel

```
DIM li1 As Integer
JSPEED 100     'Legt die Bewegungsgeschwindigkeit einer Achse auf 100 fest.
JSPEED li1/100 'Legt die Bewegungsgeschwindigkeit einer Achse auf den Wert von
               '(li1/100) fest.
```

ACCEL (Anweisung) [Entspricht SLIM]

Funktion

Gibt die interne Beschleunigung und die interne Verzögerung an.

Format

ACCEL <Beschleunigung> [,<Verzögerung>]

Erklärung

Diese Anweisung gibt die maximalen internen Beschleunigungs-/Verzögerungsraten (%) für <Beschleunigung>/<Verzögerung> im Bereich von 0,0001 bis 100 an.

ANMERKUNG: Bei einer Angabe unter 0,0001 tritt zwar kein Fehler auf, aber die tatsächliche Beschleunigung/Verzögerung kann von der Angabe abweichen. Bei Angabe von 0 oder weniger tritt ein Fehler auf.

Die tatsächlichen Werte für Beschleunigung/Verzögerung sind (extern $\times$ intern $\div$ 100).

Beim Ändern der Geschwindigkeit werden die Werte für Beschleunigung/Verzögerung automatisch in einen Wert geändert, der durch ($SPEED^2 \div 100$) ausgedrückt werden kann.

Anmerkung:	**Wenn eine Sperre einer Achsengeschwindigkeit im Parameter "Bedingung verwenden" gültig ist und ACCEL gesetzt ist, erhält JACCEL den gleichen Wert.**

Verwandte Begriffe

DECEL, SPEED

Beispiel

```
DEFINT li1, li2
ACCEL 100              'Stellt die Beschleunigung auf 100 ein.
ACCEL 50, 25           'Stellt die Beschleunigung auf 50 und die Verzögerung auf
                       '25 ein.
ACCEL li1/100, li2/100 'Stellt die Beschleunigung auf (li1/100) und die
                       'Verzögerung auf (li2/100) ein.
```

JACCEL (Anweisung) [Entspricht SLIM]

Funktion

Gibt die interne Achsenbeschleunigung und die interne Achsenverzögerung an.

Format

JACCEL <Achsenbeschleunigung> [,<Achsenverzögerung>]

Erklärung

Diese Anweisung gibt die maximale interne Rate für die Achsenbeschleunigung/-verzögerung (%) als <Achsenbeschleunigung>/<Achsenverzögerung> in einem Bereich von 0,0001 bis 100 an.

ANMERKUNG: Bei einer Angabe unter 0,0001 tritt zwar kein Fehler auf, aber die tatsächliche Beschleunigung/Verzögerung kann von der Angabe abweichen. Bei Angabe von 0 oder weniger tritt ein Fehler auf.

Die tatsächlichen Werte für die Achsenbeschleunigung/-verzögerung sind (extern $\times$ intern $\div$100).

Beim Ändern der Achsengeschwindigkeit werden die Werte für Achsenbeschleunigung/-Verzögerung automatisch in einen Wert geändert, der durch (JSPEED$^2 \div$ 100) ausgedrückt werden kann.

Verwandte Begriffe

JDECEL, SPEED

Beispiel

```
DEFINT li1, li2
JACCEL 100                'Stellt die Achsenbeschleunigung auf 100 ein.
JACCEL 50, 25             'Stellt die Achsenbeschleunigung auf 50 und die
                          'Achsenverzögerung auf 25 ein.
JACCEL li1/100, li2/100   'Stellt die Achsenbeschleunigung auf den Wert von
                          '(li1/100) und die Achsenverzögerung auf den Wert von
                          '(li2/100) ein.
```

DECEL (Anweisung) [Entspricht SLIM]

Funktion

Gibt die interne Verzögerung an.

Format

DECEL <Verzögerung>

Erklärung

Diese Anweisung gibt die maximale interne Verzögerungsrate (%) als <Verzögerung> im Bereich von 0,0001 bis 100 an.

ANMERKUNG: Bei einer Angabe unter 0,0001 tritt zwar kein Fehler auf, aber die tatsächliche Verzögerung kann von der Angabe abweichen. Bei Angabe von 0 oder weniger tritt ein Fehler auf.

Der tatsächliche Wert für die Verzögerung ist (extern × intern ÷ 100).

Beim Ändern der Geschwindigkeit wird der Wert für die Verzögerung automatisch in einen Wert geändert, der durch ($SPEED^2$ ÷ 100) ausgedrückt werden kann.

Anmerkung:	**Wenn eine Sperre einer Achsengeschwindigkeit im Parameter "Bedingung verwenden" gültig ist und DECEL gesetzt ist, erhält JDECEL den gleichen Wert.**

Verwandte Begriffe

ACCEL, SPEED

Beispiel

```
DIM li1 As Integer
DECEL 100              'Stellt die Verzögerung auf 100 ein.
DECEL li1/100          'Stellt die Verzögerung auf den Wert von (li1/100) ein.
```

JDECEL (Anweisung) [Entspricht SLIM]

Funktion

Gibt die interne Achsenverzögerung an.

Format

JDECEL <Achsenverzögerung>

Erklärung

Diese Anweisung gibt die maximale interne Achsenverzögerungsrate (%) als <Achsenverzögerung> im Bereich von 0,0001 bis 100 an.

ANMERKUNG: Bei einer Angabe unter 0,0001 tritt zwar kein Fehler auf, aber die tatsächliche Verzögerung kann von der Angabe abweichen. Bei Angabe von 0 oder weniger tritt ein Fehler auf.

Der tatsächliche Wert für die Achsenverzögerung ist (extern $\times$ intern $\div$ 100).

Beim Ändern der Achsengeschwindigkeit wird der Wert für die Achsenverzögerung automatisch in einen Wert geändert, der durch (JSPEED^2 $\div$ 100) ausgedrückt werden kann.

Verwandte Begriffe

JACCEL, SPEED

Beispiel

```
DIM li1 As Integer
JDECEL 100              'Stellt die Achsenverzögerung auf 100 ein.
JDECEL li1/100          'Stellt die Achsenverzögerung auf den Wert von (li1/100)
                        'ein.
```

CURACC (Systemvariable)

Funktion

Ruft den aktuellen internen Beschleunigungswert ab.

Format

CURACC

Erklärung

Die momentan eingestellte interne Beschleunigung wird gespeichert.

Verwandte Begriffe

ACCEL, DECEL, CURDEC

Beispiel

```
DIM lf1 As Single
lf1 = CURACC            'Weist den aktuellen Wert der Beschleunigung lf1 zu.
ACCEL CURACC * 0.5      'Stellt die Beschleunigung auf 50% der aktuellen
                        'Geschwindigkeit ein.
```

CURJACC (Systemvariable)

Funktion

Ruft den aktuellen internen Achsenbeschleunigungswert ab.

Format

CURJACC

Erklärung

Diese Anweisung speichert die momentan eingestellte interne Achsenbeschleunigung.

Verwandte Begriffe

CURJDEC, JACCEL, JDECEL, JSPEED, SPEED

Beispiel

```
DIM lf1 As Single
lf1 = CURJACC          'Weist den aktuellen Wert der Achsenbeschleunigung lf1 zu.
JACCEL CURJACC * 0.5   'Stellt die Achsenbeschleunigung auf 50% der aktuellen
                       'Geschwindigkeit ein.
```

CURDEC (Systemvariable)

Funktion

Ruft den aktuellen internen Verzögerungswert ab.

Format

CURDEC

Erklärung

Diese Anweisung speichert den momentan eingestellten Wert der internen Verzögerung.

Verwandte Begriffe

ACCEL, DECEL, CURACC

Beispiel

```
DIM lf1 As Single
lf1 = CURDEC           'Weist den aktuellen Wert der Beschleunigung lf1 zu.
DECEL CURDEC * 0.5     'Stellt die Verzögerung auf 50% des aktuellen Wertes ein.
```

CURJDEC (Systemvariable)

Funktion

Ruft den aktuellen internen Achsenverzögerungswert ab.

Format

CURJDEC

Erklärung

Diese Anweisung speichert den momentan eingestellten Wert der internen Achsenverzögerung.

Verwandte Begriffe

CURJACC, JACCEL, JDECEL, JSPEED, SPEED

Beispiel

```
DIM lf1 As Single
lf1 = CURDEC           'Weist den aktuellen Wert der Beschleunigung lf1 zu.
DECEL CURDEC * 0.5     'Stellt die Verzögerung auf 50% des aktuellen Wertes ein.
```

CURJSPD (Systemvariable)

Funktion

Ruft die interne Bewegungsgeschwindigkeit der momentan eingestellten Achse ab.

Format

CURJSPD

Erklärung

Diese Anweisung speichert die interne Bewegungsgeschwindigkeit der momentan eingestellten Achse.

Verwandte Begriffe

JSPEED

Beispiel

```
DIM lf1 As Single
lf1 = CURJSPD           'Weist den aktuellen Wert der Achsenbewegungsgeschwindigkeit
                        'lf1 zu.
JSPEED CURJSPD * 0.5    'Stellt die Geschwindigkeit der Achsenbewegung auf 50% des
                        'aktuellen Wertes ein.
```

CURSPD (Systemvariable)

Funktion

Ruft den aktuellen Wert der internen Bewegungsgeschwindigkeit an der Hand ab.

Format

CURSPD

Erklärung

Diese Anweisung speichert die interne Bewegungsgeschwindigkeit der momentan eingestellten Hand.

Verwandte Begriffe

CURACC, CURDEC, CURJSPD, CURJACC, CURJDEC

Beispiel

```
DIM lf1 As Single
lf1 = CURJSPD          'Weist den aktuellen Wert der
                       'Achsenbewegungsgeschwindigkeit zu.
JSPEED CURJSPD * 0.5   'Stellt die Geschwindigkeit der Achsenbewegung auf 50% des
                       'aktuellen Wertes ein.
```

CUREXTACC (Systemvariable) [V1.4 oder neuer]

Funktion

Ruft den aktuellen Wert der externen Beschleunigung ab.

Format

CUREXTACC

Erklärung

Speichert die momentan eingestellte externe Beschleunigung.

Verwandte Begriffe

ACCEL, DECEL, CUREXTDEC

Beispiel

```
DIM lf1 As Single
lf1 = CUREXTACC
ACCEL CUREXTACC * 0.5
```

CUREXTDEC (Systemvariable) [V1.4 oder neuer]

Funktion

Ruft den aktuellen Wert der externen Verzögerung ab.

Format

CUREXTDEC

Erklärung

Speichert die momentan eingestellte externe Verzögerung.

Verwandte Begriffe

ACCEL, DECEL, CUREXTACC

Beispiel

```
DIM lf1 As Single
lf1 = CUREXTDEC
DECEL CUREXTDEC * 0.5
```

CUREXTSPD (Systemvariable) [V1.4 oder neuer]

Funktion

Ruft den aktuellen Wert der externen Geschwindigkeit ab.

Format

CUREXTSPD

Erklärung

Speichert die momentan eingestellte externe Geschwindigkeit.

Verwandte Begriffe

SPEED

Beispiel

```
DIM lf1 As Single
lf1 = CUREXTSPD
SPEED CUREXTSPD * 0.5
```

12.5 Zeitsteuerung

DELAY (Anweisung) [Entspricht SLIM]

Funktion

Unterbricht die Programmverarbeitung für die angegebene Dauer.

Format

DELAY <Verzögerungszeit>

Erklärung

Die Programmverarbeitung wird gestoppt, bis die mit <Verzögerungszeit> angegebene Zeit abgelaufen ist.
<Verzögerungszeit> wird in ms angegeben, die tatsächliche Verzögerungszeit wird jedoch in Inkrementen zu 1/60 geändert. Wenn mehrere Aufgaben gleichzeitig verarbeitet werden, kann die Verzögerungszeit länger als der angegebene Wert sein.

Verwandte Begriffe

HOLD, WAIT

Beispiel

```
DIM li1 As Integer
DELAY 100              'Wartet bis 100ms(0,1s) verstrichen sind.
DELAY li1 + 10         'Wartet bis die Dauer (li1 + 10) verstrichen ist.
```

Anmerkungen

Wenn bei Verwendung der Verzögerungszeit während der Ausführung einer Anweisung ein sofortiger Stop erfolgt und das System neu gestartet wird, verstreicht die Verzögerungszeit auch während der Unterbrechung.

WAIT (Anweisung) [Entspricht SLIM]

Funktion

Stoppt die Programmverarbeitung abhängig von einer Bedingung.

Format

WAIT <Bedingter Ausdruck> [,<Zeitüberschreitungswert>]

Erklärung

Diese Anweisung stoppt die Programmverarbeitung, bis <Bedingter Ausdruck> erfüllt ist.
Wenn <Zeitüberschreitungswert> gesetzt ist, wird die Ausführung einer WAIT-Anweisung nach dem Verstreichen der angegebenen Zeit gestoppt, und die Verarbeitung wird mit der nächsten Anweisung fortgesetzt. Auf diese Weise kann ein unbegrenztes Stoppen vermieden werden.
<Zeitüberschreitungswert> wird in ms angegeben.
Das Neubewertungsintervall für die Überwachung von <Bedingter Ausdruck> oder <Zeitüberschreitungswert> hängt von der Priorität der Aufgabe ab.

Verwandte Begriffe

DELAY

Beispiel

```
DEFINT li1, li2, li3, li4, li5
WAIT li1 = 1          'Wartet bis li1 = 1 erfüllt ist.
WAIT li2 = 0, 2000    'Wartet bis li2 = 0 erfüllt ist. Auch wenn die Bedingung
                      'nach 2 Sekunden nicht erfüllt ist, fährt das System mit der
                      'nächsten Anweisung fort.
WAIT li3 = li4, li5   'Wartet bis li3 = li4 erfüllt ist. Auch wenn die Bedingung
                      'nach der Dauer von li5 nicht erfüllt ist, fährt das System
                      'mit der nächsten Anweisung fort.
WAIT IO[10] = ON      'Wartet bis die 10. IO ON ist
```

Anmerkungen

Wenn bei Verwendung des Zeitüberschreitungswertes während der Ausführung einer Anweisung ein sofortiger Stop erfolgt und das System neu gestartet wird, verstreicht der Zeitüberschreitungswert auch während der Unterbrechung.

12.6 Koordinatentransformation

CHANGETOOL (Anweisung)

Funktion

Ändert das Werkzeugkoordinatensystem.

Format

CHANGETOOL <Werkzeugkoordinatensystemnummer>

Erklärung

Diese Anweisung ändert das Werkzeugkoordinatensystem in das Koordinatensystem, das mit der <Werkzeugkoordinatensystemnummer> angegeben wurde.
Diese Deklaration ist bis zur Ausführung der nächsten CHANGETOOL-Anweisung gültig.
Zahlen von 0 bis 63 sind für die <Werkzeugkoordinatensystemnummer> gültig.
Werkzeug 0 ist ein mechanisches Schnittstellen-Koordinatensystem.

Verwandte Begriffe

TOOL, TOOLPOS

Beispiel

```
DIM li1 As Integer
DIM lp1 As Position
Li1 = 1
Lp1 = (10, 10, 5, 0, 9, 0, 1)
WERKZEUG 1, lp1         'Deklariert das Werkzeugkoordinatensystem, das mit der Variablen
                        'lp1 des Typs Position angegeben wurde.
CHANGETOOL li1          'Wechselt zu dem Werkzeugkoordinatensystem, das mit der
                        'in li1 angegebenen Werkzeugkoordinatensystemnummer angegeben
                        'wurde.
```

Anmerkungen

Bei Ausführung von TAKEARM wird WERKZEUG0 eingestellt.

CHANGEWORK (Anweisung)

Funktion

Ändert das Benutzerkoordinatensystem.

Format

CHANGEWORK <Benutzerkoordinatensystemnummer>

Erklärung

Diese Anweisung ändert das Benutzerkoordinatensystem in das Koordinatensystem, das mit der <Benutzerkoordinatensystemnummer> angegeben wurde.
Diese Deklaration ist bis zur Ausführung der nächsten CHANGEWORK-Anweisung gültig.
Zahlen von 0 bis 7 sind für die <Benutzerkoordinatensystemnummer> gültig.
Benutzer 0 gibt das Basiskoordinatensystem an.

Verwandte Begriffe

WORK, WORKPOS

Beispiel

```
DIM li1 As Integer
DIM lp1 As Position
Li1 = 1
Lp1 = (10, 10, 5, 0, 9, 0, 1)
WORK 1, lp1             'Deklariert das Benutzerkoordinatensystem, das mit der
                        'Variablen lp1 des Typs Position
                        'im Benutzerkoordinatensystem 1 angegeben wurde.
CHANGEWORK li1          'Wechselt zu dem Benutzerkoordinatensystem, das mit der in
                        'li1 angegebenen Benutzerkoordinatensystemnummer angegeben
                        'wurde.
```

Anmerkungen

Bei Ausführung von TAKEARM wird ARBEIT0 eingestellt.
(Dies gilt jedoch nicht für Unterprogramme.)

CURTOOL (Systemvariable) [V1.4 oder neuer]

Funktion

Ruft die Nummer des momentan angegebenen Werkzeugs ab.

Format

CURTOOL

Erklärung

Speichert die Nummer des momentan angegebenen Werkzeugs.

Verwandte Begriffe

TOOL, CHANGETOOL, TOOLPOS

Beispiel

```
DIM li1 As Integer
li1 = CURTOOL
TOOL CURTOOL+Il, Pl
CHANGETOOL CURTOOL+Il
Pl = TOOLPOS (CURTOOL)
```

CURWORK (Systemvariable) [V1.4 oder neuer]

Funktion

Ruft die Nummer der momentan angegebenen Arbeit ab.

Format

CURWORK

Erklärung

Speichert die Nummer der momentan angegebenen Arbeit.

Verwandte Begriffe

WORK, CHANGEWORK, WORKPOS

Beispiel

```
DIM li1 As Integer
li1 = CURWORK
WORK CURWORK+Il, Pl
CHANGEWORK CURWORK+Il
Pl = WORKPOS (CURWORK)
```

12.7 Interferenzprüfung

SETAREA (Anweisung)

Funktion

Wählt den Bereich aus, in dem eine Interferenzprüfung ausgeführt wird.

Format

SETAREA <Bereichsprüfung für Interferenzprüfung>

Erklärung

<Bereichsnummer für Interferenzprüfung> wählt den zuvor definierten Interferenzprüfungsbereich aus. Gültige Nummern sind von 0 bis 7. Dies muß vor Verwendung des Befehls AREA deklariert werden.
Eine Interferenzprüfung wird nach der Ausführung von SETAREA ausgeführt und vor der Ausführung von RESETAREA. Wenn eine Interferenz erkannt wird, wird der Befehl RESETAREA ausgeführt. Die nächste Interferenz wird nicht erkannt, bis SETAREA erneut ausgeführt wird.
Eine Interferenzprüfung kann gleichzeitig in maximal 8 Bereichen ausgeführt werden, die mit den Bereichsnummern 0 bis 7 angegeben werden. Voraussetzung ist, daß der SETAREA-Befehl ausgeführt wurde.
Der SETAREA-Befehl initialisiert die mit dem AREA-Befehl deklarierte angegebene E/A nicht. In einem solchen Fall wird eine E/A direkt nicht in der Anweisung zur Ein-/Ausgabesteuerung gesetzt, und die Initialisierung des RESETAREA-Befehls ist erforderlich.

Verwandte Begriffe

AREA, RESETAREA, AREAPOS, AREASIZE

Beispiel

```
DIM lp1 As Position
DIM lv1 As Vector
Lp1 = (10, 10, 5, 0, 9, 0, 1)
Lv1 = (50, 10, 50)
AREA 2, lp1, lv1, 104, 1   'Deklariert die E/A-Nummer 104 in dem mit lv1
                           'angegebenen Bereich an der mit lp1 in Bereichsnummer 2
                           'angegebenen Position.
SETAREA 2                  'Macht die Bereichsprüfung von Nummer 2 gültig.
RESETAREA 2                'Initialisiert die Bereichsprüfung von Nummer 2.
```

RESETAREA (Anweisung)

Funktion

Initialisiert eine Interferenzprüfung.

Format

RESETAREA <Initialisieren der Bereichsnummer>

Erklärung

Diese Anweisung setzt die eingestellte E/A zurück, wenn eine Interferenz erkannt wurde, und macht die Interferenzprüfung ungültig.
Wenn die Ein-/Ausgangs-Steueranweisung eine E/A direkt zurücksetzt, auch wenn erneut eine Interferenz auftritt, wird die E/A nicht gesetzt. Führen Sie zur erneuten Prüfung der Interferenz den Befehl RESETAREA zur Initialisierung aus, und führen Sie anschließend den Befehl SETAREA erneut aus.
Gültige Bereichsnummern sind 0 bis 7.

Verwandte Begriffe

AREA, SETAREA, AREAPOS, AREASIZE

Beispiel

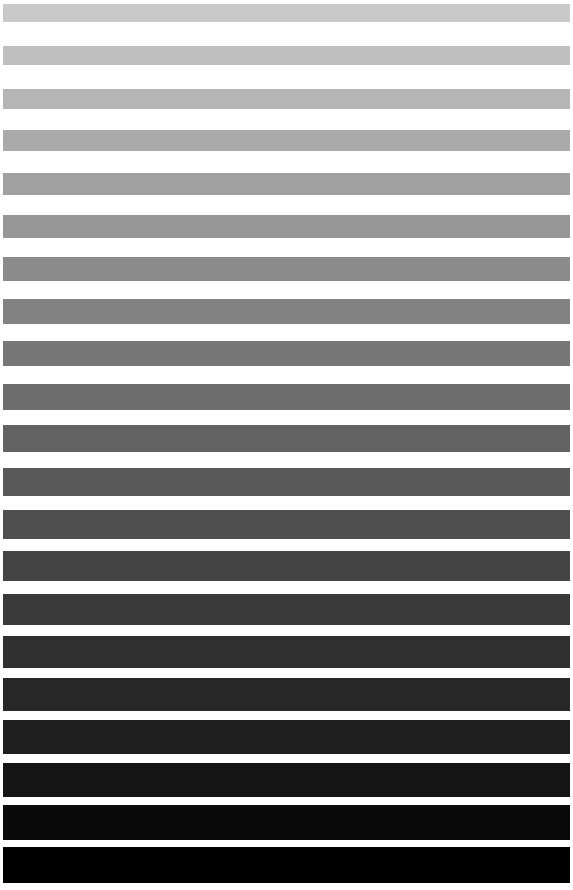
```
DIM lp1 As Position
DIM lv1 As Vector
Lp1 = (10, 10, 5, 0, 9, 0, 1 )
Lv1 = (50, 10, 50 )
AREA 2, lp1, lv1, 104, 1     'Deklariert die E/A-Nummer 104, die durch lv1 an der
                             'mit lp1 in Bereichsnummer 2 angegebenen Position
                             'angegeben wurde.
SETAREA 2                    'Macht die Bereichsprüfung von Nummer 2 gültig.
RESETAREA 2                  'Initialisiert die Bereichsprüfung von Nummer 2.
```

Anmerkungen

Der Roboter arbeitet in einer Multitasking-Betriebsumgebung. Wenn RESET unmittelbar auf die Ausführung von SETAREA zur Prüfung eines Bereichs folgt, kann es daher passieren, daß RESET wegen des Timings tatsächlich vor SETAREA ausgeführt wird.

Kapitel 13

Steuerungs-anweisungen für die Ein-/ Ausgabe



Dieses Kapitel enthält eine Erläuterung zu den Befehlen zur Steuerung der verschiedenen E/A-Komponenten.

13.1 E/A-Anschluß

IN (Anweisung) [Entspricht SLIM]

Funktion

Liest Daten aus dem E/A-Anschluß, der mit einer E/A-Variablen angegeben wurde.

Format

IN <Name der arithmetischen Variable> = <E/A-Variable>

Erklärung

Diese Anweisung weist die Daten des mit der <E/A-Variablen> angegebenen E/A-Anschlusses der mit <Name der arithmetischen Variable> angegebenen Variablen zu.
Die <E/A-Variable> wird mit einer DEFIO-Anweisung oder einer Variablen des Typs E/A deklariert.

Verwandte Begriffe

OUT, DEFIO

Beispiel

```
DEFINT Li1, Li2
DEFIO samp1 = INTEGER, 220  'Deklariert samp1 als eine E/A-Variable des Typs Ganze Zahl
                            'mit Anfang bei Anschluß 220.
IN Li1 = samp1              'Weist die Daten von samp1 Li1 zu.
IN Li2 = IO[240]            'Weist die Daten von Anschluß 240 Li2 zu.
OUT samp1 = Li1             'Gibt die Li1-Daten an den in samp1 deklarierten Anschluß aus.
OUT IO[240] = Li2           'Gibt die Li2-Daten an den Anschluß 240 aus.
```

OUT (Anweisung) [Entspricht SLIM]

Funktion

Gibt Daten an den mit einer E/A-Variablen angegebenen E/A-Anschluß aus.

Format

OUT <E/A-Variable> = <Ausgabedaten>

Erklärung

Diese Anweisung gibt den Wert von <Ausgabedaten> an die mit <E/A-Variable> angegebene Anschlussadresse aus.
<E/A-Variable> wird mit einer DEFIO-Anweisung oder einer Variablen des Typs E/A deklariert.

Verwandte Begriffe

IN, DEFIO

Beispiel

```
DEFINT Li1, Li2
DEFIO samp1 = INTEGER, 220  'Deklariert samp1 als eine E/A-Variable des Typs Ganze Zahl
                            'mit Anfang bei Anschluß 220.
IN Li1 = samp1              'Weist die Daten von samp1 Li1 zu.
IN Li2 = IO[240]            'Weist die Daten von Anschluß 240 Li2 zu.
OUT samp1 = Li1             'Gibt die Li1-Daten an den in samp1 deklarierten Anschluß aus.
OUT IO[240] = Li2           'Gibt die Li2-Daten an den Anschluß 240 aus.
```

IOBLOCK ON/OFF (Anweisung) [Entspricht SLIM]

Funktion

Führt während der Ausführung einer Bewegungsanweisung gleichzeitig eine Nicht-Bewegungsanweisung wie beispielsweise eine E/A- oder Berechnungsanweisung durch.

Format

IOBLOCK {ON|OFF}

Erklärung

Verwenden Sie IOBLOCK ON und IOBLOCK OFF paarweise. Nicht-Bewegungsanweisungen wie beispielsweise mehrfache E/A-Anweisungen oder Berechnungsanweisungen im Anschluß an eine Bewegungsanweisung können gleichzeitig mit einer Roboter-Bewegungsanweisung ausgeführt werden. Bei der Ausgabe einer Roboter-Bewegungsanweisung unterbricht der Controller die parallel ausgeführte Anweisung und wartet, bis die momentan ausgeführte Operation (Durchlaufstart im Fall einer Durchlaufbewegung) abgeschlossen ist. Anschließend führt der Controller die nächste Bewegungsanweisung aus. Wenn der nächsten Bewegungsanweisung eine Nicht-Bewegungsanweisung folgt, wird diese ebenfalls gleichzeitig mit der Bewegungsanweisung ausgeführt. Wenn IOBLOCK OFF ist während der Ausführung einer Bewegungsanweisung in IOBLOCK, fährt das System nach der Ausführung der Bewegungsanweisung mit dem nächsten Schritt fort.

Verwandte Begriffe

Beispiel

```
TAKEARM               'Ruft die Robotersteuerungspriorität ab.
IOBLOK ON             'Führt gleichzeitig eine E/A-Anweisung mit der nächsten
                      'Bewegungsanweisung aus.
MOVE P(902.7,0,415.3,180,50,180,1)
                      'Bewegung (Punkt-zu-Punkt-Steuerung) an die Koordinaten
                      '(902.7,0,415.3,180,50,180,1).
SET IO[240]           'Setzt den Anschluß vom Typ 240 BIT auf ON.
SET IO[241],40        'Setzt den Anschluß vom Typ 241 BIT auf ON für 40 ms.
SET IO[SOL1]          'Setzt den mit der E/A-Variablen SOL1 angegebenen Anschluß
                      'auf ON.
SET IO[104 TO 110]    'Setzt den Anschluß vom Typ 104 ~ 110 auf ON.
IF IO[242] THEN
RESET IO[240]         'Setzt den Anschluß vom Typ 240 auf OFF.
RESET IO[SOL1]        'Setzt den mit der E/A-Variablen SOL1 angegebenen Anschluß
                      'auf OFF.
RESET IO[104 TO 110]  'Setzt den Anschluß vom Typ 104 ~ 110 auf OFF.
ENDIF
IOBLOCK OFF
```

Anmerkungen

(1) Die gleichzeitige Verarbeitung wird in den folgenden Fällen nicht ausgeführt:
   i) Wenn eine Bewegungsoption einer Bewegungsanweisung hinzugefügt wird.
   ii) Wenn CHANGEWORK, CHANGETOOL, SPEED, JSPEED, ACCEL oder JACCEL ausgeführt wird.
   iii) Wenn die konventionelle Sprachbibliothek von [aspChange], [aspACLD], Armbewegungsbibliothek [mvSetPulseWidth], [mvSetTime-Out], [mvReverseFlip], [mvResetPulsetWidth] oder [mvResetTimeOut] ausgeführt wird.

   Die Ausführung des nächsten Schrittes sollte in allen Fällen gestoppt werden, bis die Bewegungsanweisung abgeschlossen ist (von der Angabe der Durchlaufbewegung bis zum Beginn der Durchlaufbewegung).

(2) Wenn das System nach einem Schritt oder einem sofortigen Stopp des Roboters während der Roboterbewegung neu gestartet wird, kann der Roboter keine Bewegung gleichzeitig mit einem Nicht-Bewegungsbefehl ausführen.

(3) Wenn der Roboter nach einem Schritt während der ersten Bewegung gestoppt wird, wenn eine Bewegungsanweisung in IOBLOCK zwei Mal wiederholt wird, stoppt der Roboter nach einem Schritt, sobald die nächste Bewegung abgeschlossen ist. Wenn Sie während der Ausführung der ersten Bewegung einen sofortigen Stop ausführen und den Schritt ausführen, wird der Roboter nach dem Schritt ebenfalls gestoppt, sobald die nächste Bewegung abgeschlossen ist.
   Beispiel
   ```
   IOBLOCK ON
   MOVE P, JO    ← Während der Bewegung zu JO wird die nächste
                    Bewegung von MOVE P, J1 auch dann ausgeführt,
                    wenn der Roboter nach einem Schritt gestoppt wird.
                    Danach stoppt der Roboter nach einem Schritt.
   MOVE P, J1
   SET IO[1]
   IOBLOCK OFF
   ```

(4) Der Bereich von IOBLOCK ist nur gültig in einem definierten Programm, nicht jedoch in einem Unterprogramm.
   Beispiel
   ```
   PROGRAM PRO1
   TAKEARM
   DEFPOS 1p1, 1p2
   IOBLOCK ON
   CALL SUB1
   MOVE P, 1p1, 1p2
   SET IO[1]
   SET IO[2]
   IOBLOCK OFF
    :
    :
   ```
Für diese Art Programm ist die Definition von IOBLOCK in PRO1 in SUB1 nicht zulässig. Der gültige Bereich für IOBLOCK ist für jedes Programm unabhängig.

(5) IOBLOCK wird im Programmierungs-Prüfmodus deaktiviert.

SET (Anweisung) [Entspricht SLIM]

Funktion

Stellt einen E/A-Anschluß auf ON ein.

Format

SET <E/A-Variable>[,<Ausgabezeit>]

Erklärung

Diese Anweisung setzt den mit <E/A-Variable> angegebenen Anschluß auf ON.
Wenn <Ausgabezeit> angegeben ist, wird ein Impuls ausgegeben. (Die Ausgabezeit ist ms.)
Wenn <Ausgabezeit> angegeben ist, fährt das System nicht mit der nächsten Anweisung fort, bis diese Zeit verstrichen ist. Der angegebene Wert für die Ausgabezeit ist die minimale Ausgabezeit, während die tatsächliche Ausgabezeit sich entsprechend der Aufgabenpriorität ändert.

Verwandte Begriffe

RESET, DEFIO

Beispiel

```
TAKEARM               'Ruft die Priorität der Robotersteuerung ab.
IOBLOK ON             'Führt gleichzeitig mit der nächsten Bewegungsanweisung eine
                      'E/A-Anweisung aus.
MOVE P,(902.7,0,415.3,18050,180,1)
                      'Bewegung (Punkt-zu-Punkt-Steuerung) an die Koordinaten
                      '(902.7,0,415.3,180,50,180,1).
SET IO[240]           'Setzt den Anschluß vom Typ 240 BIT auf ON.
SET IO[241],40        'Setzt den Anschluß vom Typ 241 BIT auf ON für 40 ms.
SET IO[SOL1]          'Setzt den mit der E/A-Variablen SOL1 angegebenen Anschluß auf ON.
SET IO[104 TO 110]    'Setzt den Anschluß vom Typ 104 bis 110 BIT auf ON.
IF IO[242] THEN
RESET IO[240]         'Setzt den Anschluß vom Typ 240 auf OFF.
RESET IO[SOL1]        'Setzt den mit der E/A-Variablen SOL1 angegebenen Anschluß auf OFF.
RESET IO[104 TO 110]  'Setzt den Anschluß vom Typ 104 bis 110 BIT auf OFF.
ENDIF
IOBLOCK OFF
```

Anmerkungen

(1) Wenn die Ausgabezeit angegeben ist, kann sie durch Faktoren wie die Präsenz eines anderen Programmes während der Bewegung, eine Operation des Programmiergeräts oder die Kommunikation mit externen Geräten verlängert werden.

(2) Wenn die Ausgabezeit angegeben ist, kann sie um ±16,7 ms abweichen, da der Standardtakt für die Controller-Verarbeitung 16,7 ms beträgt.

(3) Beachten Sie die beiden folgenden Punkte bei der Verwendung der Zeitangabe SET.

1) Wenn Sie den gleichen Anschluß über eine andere Task mit RESET zurücksetzen, während der Anschluß ON ist, wird der Anschluß wegen der Zeitangabe SET ab dem Zeitpunkt von RESET auf OFF gesetzt (dies bedeutet, die Zeitangabe SET ist gültig).

2) Wenn Sie den gleichen Anschluß weiterhin zurücksetzen und über eine andere Task mit SET setzen, während der Anschluß ON ist, wird der Anschluß wegen der Zeitangabe SET auf OFF gesetzt, nachdem die angegebene Dauer verstrichen ist (dies bedeutet, die Zeitangabe SET ist gültig).

(4) Wenn die Ausgabezeit verwendet wird, beachten Sie, daß auch bei einem vorübergehenden Stoppen die Ausgabezeit nach dem sofortigen Stop während der Ausführung der Anweisung und dem Neustart des Systems verstreicht.

RESET (Anweisung) [Entspricht SLIM]

Funktion

Stellt einen E/A-Anschluß auf OFF ein.

Format

RESET <E/A-Variable>

Erklärung

Setzt den mit der <E/A-Variablen> angegebenen Anschluß auf OFF.

Verwandte Begriffe

SET, DEFIO

Beispiel

```
TAKEARM               'Ruft die Priorität der Robotersteuerung ab.
IOBLOK ON             'Führt gleichzeitig eine E/A-Anweisung mit der nächsten
                      'Anweisung aus.
MOVE P, (902.7,0,415.3,180,50,180,1)
                      'Bewegung zu den Koordinaten (902.7,0,415.3,180,50,180,1)
                      '(Punkt-zu-Punkt-Steuerung)
SET IO[240]           'Setzt den Anschluß vom Typ 240 BIT auf ON.
SET IO[241],40        'Setzt den Anschluß vom Typ 241 BIT auf ON für 40 ms.
SET IO[SOL1]          'Setzt den mit der E/A-Variablen SOL1 angegebenen Anschluß auf ON.
SET IO[104 TO 110]    'Setzt den Anschluß vom Typ 104 bis 110 BIT auf ON.
IF IO[242] THEN
RESET IO[240]         'Setzt den Anschluß vom Typ 104 bis 110 BIT auf OFF.
RESET IO[SOL1]        'Setzt den mit der E/A-Variablen SOL1 angegebenen Anschluß auf OFF.
RESET IO[104 TO 110]  'Setzt den Anschluß vom Typ 104 bis 110 BIT auf OFF.
ENDIF
IOBLOCK OFF
```

13.2 RS232C Port

INPUT (Anweisung) [Entspricht SLIM]

Funktion

Ruft Daten von dem RS232C-Anschluß ab.

Format

INPUT [#<Schaltkreisnummer>,]<Variablenname>[,<Variablenname>…]

Erklärung

Diese Anweisung speichert Daten, die über den RS232C-Anschluß empfangen wurden, in der mit <Variablenname> angegebenen Variablen.
Geben Sie eine Schaltkreisnummer an, die für <Schaltkreisnummer> verwendet werden soll. Wenn <Schaltkreisnummer> nicht angegeben ist, wird der Standardwert ch2 gesetzt. Ch1 kann nicht angegeben werden, da es für das Programmiergerät verwendet wird. (Siehe Abschnitt 2.4.1 “Schaltkreisnummer” in Kapitel 2 dieses Handbuches.)
Wenn mehrfache Informationen für die gleiche Anzahl von Variablen empfangen werden, fügen Sie ein Komma (,) zwischen den einzelnen Variablen ein.
Geben Sie die Baudrate mit einem Systemparameter ein.

Anmerkung (1):	**Führen Sie vor dem Empfangen von Daten einen FLUSH-Befehl aus, um die Daten zu löschen, die vom vorigen Empfang noch im Eingabepuffer stehen.**
Anmerkung (2):	**Wenn die Eingabedaten den maximalen Wert des Variablentyps, dem sie zugeordnet wurden, übersteigen, entspricht das Ergebnis dem maximalen Wert dieses Typs.**

Verwandte Begriffe

FLUSH, PRINT, WRITE

Beispiel

```
DIM li1 As Integer
DEFSTR ls1, ls2, ls3, ls4
INPUT #1, ls1               'Schreibt Daten von ch2 in ls1.
INPUT #1, li1, ls2, ls3     'Schreibt Daten von ch2 in li1, ls2 und ls3.
INPUT ls4                   'Schreibt Daten von ch2 in ls4.
```

LINEINPUT (Anweisung)

Funktion

Liest Daten über einen RS232C-Anschluß in einen Begrenzer ein und weist sie einer Variablen des Typs Zeichenfolge zu.

Format

LINEINPUT [#<Schaltkreisnummer>,] <Variable des Typs Zeichenfolge>

Erklärung

Alle Zeichen, von den über einen RS232C-Anschluß empfangenen Daten bis zu einem Rücklauf- (CR) oder Zeilenschaltungscode (CR + LF), werden in einer mit <Variable des Typs Zeichenfolge> Variablen gespeichert. Der Rücklauf- und Zeilenschaltungscode wird nicht in der Variablen gelesen.
Geben Sie eine Schaltkreisnummer an, die für <Schaltkreisnummer> verwendet werden soll. Wenn <Schaltkreisnummer> nicht angegeben ist, wird der Standardwert ch2 gesetzt. Ch1 kann nicht angegeben werden, da es für das Programmiergerät verwendet wird. (Siehe Abschnitt 2.4.1 “Schaltkreisnummer” in Kapitel 2 dieses Handbuches.)
Geben Sie den Namen der Variablen des Typs Zeichenfolge für <Variable des Typs Zeichenfolge> an.

Verwandte Begriffe

Beispiel

```
DEFSTR ls1,ls2,ls3
LINEINPUT #0, ls1  'Empfängt eine Zeichenfolge von Anschluß 1 und ordnet sie ls1 zu.
LINEINPUT #1, ls2  'Empfängt eine Zeichenfolge von Anschluß 2 und ordnet sie ls2 zu.
LINEINPUT ls3      'Empfängt eine Zeichenfolge von Anschluß 2 und ordnet sie ls3 zu.
```

PRINT (Anweisung) [Entspricht SLIM]

Funktion

Gibt Daten an einen RS232C-Anschluß aus.

Format

PRINT [#<Schaltkreisnummer>,] <Meldung> [<Trennzeichen> <Meldung>…] [<Trennzeichen>]

Erklärung

Diese Anweisung gibt den Wert von <Meldung> von einem RS232C-Anschluß an ein externes Gerät aus.
Geben Sie eine Schaltkreisnummer an, die für <Schaltkreisnummer> verwendet werden soll.
Wenn <Schaltkreisnummer> nicht angegeben ist, wird der Standardwert ch2 gesetzt. Ch1 kann nicht angegeben werden, da es für das Programmiergerät verwendet wird. (Siehe Abschnitt 2.4.1 “Schaltkreisnummer” in Kapitel 2 dieses Handbuches.)
Für <Trennzeichen> kann ein Komma (,) oder Semikolon (;) verwendet werden.
Bei Verwendung eines Kommas (,) fügen Sie ein Leerzeichen zwischen dem Komma und der darauffolgenden <Meldung> ein.
Wenn ein Komma (,) nach <Meldung> angegeben ist, aber keine <Meldung> gefunden wird, wird nach der letzten <Meldung> ein Zeilenschaltungscode (CR + LF) übertragen, sobald die Ausgabe abgeschlossen ist.
Bei Verwendung eines Semikolons (;) darf kein Leerzeichen zwischen dem Semikolon und der <Meldung> eingefügt werden. Wenn ein Semikolon (,) nach <Meldung> angegeben ist, aber keine <Meldung> gefunden wird, wird nach der letzten <Meldung> ein Zeilenschaltungscode (CR + LF) übertragen, sobald die Ausgabe abgeschlossen ist.
Bei der Ausgabe von Daten vom Typ Stellung werden die einzelnen Elemente für die Ausgabe durch ein Leerzeichen voneinander getrennt.

Verwandte Begriffe

WRITE

Beispiel

```
DEFINT li1, li2
DEFSTR ls1, ls2
PRINT #1, li1, ls1; ls2      'Gibt die Werte von li1, ls1 und ls2 von ch2 aus.
                             '(Zwischen li1 und ls1 wird ein Leerzeichen
                             'eingefügt, nicht jedoch zwischen ls1 und ls2.)
PRINT li2                    'Gibt den Wert von li2 von ch2 aus.
```

WRITE (Anweisung)

Funktion

Gibt Daten an einen RS232C-Anschluß aus.

Format

WRITE [#<Schaltkreisnummer>,]<Meldung>[,<Meldung>…]

Erklärung

Diese Anweisung gibt den Wert von <Meldung> von einem RS232C-Anschluß an ein externes Gerät as.
Geben Sie für <Schaltkreisnummer> die Schaltkreisnummer des zu verwendenden RS232C-Anschlusses an. Wenn <Schaltkreisnummer> nicht angegeben ist, wird der Standardwert ch2 gesetzt. Ch1 kann nicht angegeben werden, da es für das Programmiergerät verwendet wird. (Siehe Abschnitt 2.4.1 “Schaltkreisnummer” in Kapitel 2 dieses Handbuches.)
Wenn mehrere <Meldung>en geschrieben werden, müssen die Meldungen durch ein Komma (,) getrennt werden.
In den folgenden Punkten unterscheidet sich diese Anweisung von der PRINT-Anweisung.

- Zeichenfolgedaten werden in doppelten Anführungszeichen (”) ausgegeben.
- Am Ende der Ausgabeinformationen wird ein Zeilenschaltungscode (CR+LF) eingefügt.
- Kommas (,) zur Trennung von Meldungen (<Meldung>) werden unverändert ausgegeben.
- Bei der Ausgabe von Daten vom Typ Stellung werden die einzelnen Elemente für die Ausgabe durch ein Komma (,) voneinander getrennt.

Verwandte Begriffe

PRINT

Beispiel

```
DEFINT li1, li2
DEFSTR ls1, ls2
WRITE #1, li1, ls1, ls2     'Gibt die Werte von li1, ls1 und ls2 von ch2 aus.
WRITE li2                   'Gibt den Wert von li2 von ch2 aus.
```

FLUSH (Anweisung)

Funktion

Löscht den Eingabepuffer.

Format

FLUSH [#<Schaltkreisnummer>]

Erklärung

Diese Anweisung löscht den Eingabepuffer des RS232C-Anschlusses.
Geben Sie eine Schaltkreisnummer an, um den Eingabepuffer von <Schaltkreisnummer> zu löschen.
Wenn <Schaltkreisnummer> nicht angegeben ist, wird der Standardwert ch2 gesetzt. Ch1 kann nicht angegeben werden, da es für das Programmiergerät verwendet wird. (Siehe Abschnitt 2.4.1 “Schaltkreisnummer” in Kapitel 2 dieses Handbuches.)
Führen Sie vor dem Empfangen von Daten einen FLUSH-Befehl aus, um die Daten im Eingabepuffer zu löschen.

Verwandte Begriffe

EINGABE

Beispiel

```
FLUSH #1          'Löscht den Eingabepuffer des Schaltkreises ch2.
FLUSH             'Löscht den Eingabepuffer des Schaltkreises ch2.
```

13.3 Programmiergerät

PRINTMSG (Anweisung)

Funktion

Zeigt eine Meldung mit einer Beschriftung und einem Symbol auf der Farb-LCD des Programmiergeräts an.

Format

PRINTMSG <Meldungszeichen>,<Symboltyp>,<Bezeichnungstext>

Erklärung

Das System zeigt die mit den einzelnen Argumenten angegebenen Meldungen mit einer Bezeichnung und einem Symbol auf der Farb-LCD des Programmiergeräts an.

Symboltyp	Symbol
0	
1	
2	
3	
4	

Maximal 60 Zeichen können für eine Meldungszeichenfolge verwendet werden, und maximal 40 Zeichen für einen Bezeichnungstext.

Verwandte Begriffe

PRINTDBG, PRINTLBL

Beispiel

```
PRINTMSG “Hallo Welt !”, 1, “Meldung”        'Zeigt “Hallo Welt !” mit einer
                                             'Bezeichnung und einem Symbol an.
```

PRINTDBG (Anweisung)

Funktion

Gibt Daten an das Debug-Fenster aus.

Format

PRINTDBG <Meldung>[<Trennzeichen><Meldung>…][<Trennzeichen>]

Erklärung

Diese Anweisung gibt den Wert von <Meldung> im Debug-Fenster des Programmiergeräts aus.
Für <Trennzeichen> kann ein Komma (,) oder Semikolon (;) verwendet werden.
Bei Verwendung eines Kommas (,) fügen Sie ein Leerzeichen zwischen dem Komma und der darauffolgenden <Meldung> ein.
Bei Verwendung eines Semikolons (;) darf kein Leerzeichen zwischen dem Semikolon und der <Meldung> eingefügt werden.
Bei der Ausgabe von Daten vom Typ Stellung werden die einzelnen Elemente durch ein Leerzeichen voneinander getrennt und angezeigt.
Der Zeilenvorschub erfolgt, wenn kenn <Trennzeichen> am Ende der <Meldung> steht.

Verwandte Begriffe

PRINTMSG, PRINTLBL

Beispiel

```
PRINTDBG “DEBUG”  'Gibt “DEBUG” an das Debug-Fenster aus.
```

BUZZER (Anweisung)

Funktion

Gibt einen Buzzer-Ton aus.

Format

BUZZER <Tondauer>

Erklärung

Diese Anweisung gibt am Programmiergerät einen Buzzer-Ton mit der unter <Tondauer> angegebene Länge aus.
Die Einheit von <Tondauer > ist msec.

Verwandte Begriffe

Beispiel

```
BUZZER 3000       'Der Buzzer ertönt 3 Sekunden lang.
```

PRINTLBL (Anweisung)

Funktion

Definiert eine Kennung (Beschriftung) für eine benutzerdefinierte Schaltfläche.

Format

PRINTLBL <Bedienfeldnummer>, <Tastennummer>,<Bezeichnungstext>

Erklärung

Diese Anweisung definiert eine Kennung (Bezeichnung) für jede benutzerdefinierte Taste am Bedienfeld des Programmiergeräts.
Für <Bedienfeldnummer> kann eine Zahl von 0 bis 6 angegeben werden.
Geben Sie für <Tastennummer> eine Zahl von 0 bis 11 an.
Geben Sie die Zeichenfolge mit maximal 6 Zeichen in <Bezeichnungstext> ein.

Verwandte Begriffe

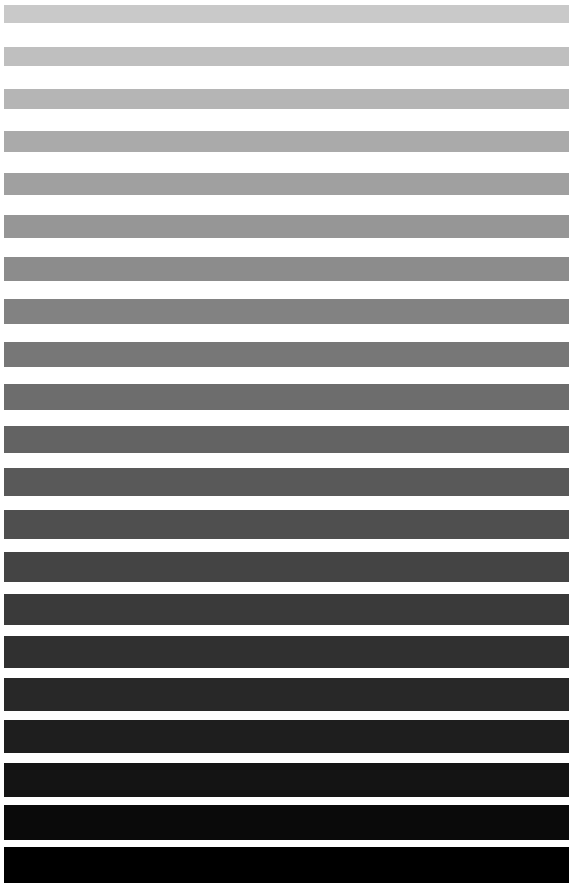
PRINTDBG, PRINTMSG

Beispiel

```
PRINTLBL 3, 1, “E_STOP”    'Legt die Kennung der ersten Taste am dritten Bedienfeld
                           'auf “E_STOP” fest.
```

Kapitel 14

Die Multitasking-Steueranweisungen



Multitasking-Steuerung ist eine der Funktionen von PAC. Dieses Kapitel enthält Erläuterungen zu den Befehlen für die Multitasking-Steuerung.

14.1 Task-Steuerung

RUN (Anweisung)

Funktion

Führt gleichzeitig ein anderes Programm aus.

Format

RUN <Programmname> [(<Argument>[,<Argument>…])][,<RUN-Option>]

Erklärung

Diese Anweisung ermöglicht dem momentan ausgeführten Programm die Ausführung eines mit <Programmname> angegebenen Programmes. Das aktuelle Programm kann es jedoch nicht selbst ausführen.
Für <Argument> sind nur Werte zulässig. Auch wenn Sie Daten als Referenz übergeben, werden die Referenzdaten automatisch in Werte umgewandelt. Sie können jedoch keine lokalen Arrays verwenden.
Für <RUN-Option> kann PRIORITY (oder P) und CYCLE (oder C) angegeben werden.

PRIORITY (oder P)
Gibt die Priorität eines Programmes an. Wenn diese Option nicht angegeben wird, gilt der Standardwert 128. Je kleiner der Wert, desto höher ist die Priorität. Der gültige Bereich für die Einstellung reicht von 101 bis 255.

CYCLE (oder C)
Gibt einen alternativen Zyklus (Anzahl der Ausführungszyklen bei wiederholter Ausführung eines Programmes) an. Diese Option wird in ms angegeben. Der gültige Bereich reicht von 1 bis 2.147.483.647.

Bei Verwendung der CYCLE-Option können Sie kein Programm starten, das Argumente enthält.

Verwandte Begriffe

CALL, GOSUB

Beispiel

```
DEFINT Li1 = 1, Li2 =2, Li3 = 3
RUN samp1 C=1000    'Führt samp1 parallel (C=1000) aus.
RUN samp2(Li1)      'Führt samp2 mit dem Argument Li1 parallel aus.
RUN samp3(Li1,Li2),PRIORITY = 129
                    'Führt samp3 mit den Argumenten Li1 und Li2 parallel aus
                    '(P = 129).
RUN samp4(Li1,Li2),PRIORITY = 150
                    'Führt samp4 mit den Argumenten Li1 und Li2 parallel aus
                    '(P = 150).
RUN samp5(Li1,Li2,Li3)P = 120
                    'Führt samp5 mit den Argumenten Li1, Li2 und Li3 parallel aus
                    '(P = 120)
```

Anmerkungen

(1) Wenn eine Task, für die die Bewegung unterbrochen wurde, erneut mit einer SUSPEND-Anweisung ausgeführt wird, führen Sie RUN aus, nachdem die Bewegung vollständig abgeschlossen ist. Wenn RUN ausgeführt und der Roboter erneut bewegt wird, bevor das Stoppen der Bewegung abgeschlossen ist, kann ein Fehler wie “Befehlsgeschwindigkeitsbegrenzung überschritten” auftreten.
(2) Beachten Sie bei Verwendung einer CYCLE-Option, daß die Zyklussynchronisation auch während der Unterbrechung weiter abläuft, wenn die Ausführung der Anweisung durch einen sofortigen Stop unterbrochen und der Roboter anschließend erneut gestartet wurde.
(3) Der RUN-Befehl wird im Prüfmodus des Programmiergeräts nicht ausgeführt.

KILL (Anweisung)

Funktion

Erzwingt das Beenden einer Task.

Format

KILL <Programmname>

Erklärung

Diese Anweisung erzwingt das Beenden der mit <Programmname> angegebenen Task (des Programmes). Sie kann jedoch kein Programm, das die Anweisung enthält, beenden. Bei dem Versuch einer solchen Beendigung tritt ein Fehler auf. Verwenden Sie die Anweisung STOP, um das Beenden eines Programmes mit der Anweisung zu erzwingen.

Verwandte Begriffe

SUSPEND, STOP

Beispiel

```
RUN samp1              'Führt samp1 gleichzeitig aus.
   .
   .
   .
KILL samp1             'Beendet samp1.
```

Anmerkungen

Wenn eine Task während des Abrufens eines Arm-Semaphor erzwungenermaßen beendet wird und ein Arm-Semaphor über eine andere Task abgerufen wird, kann ein “Fehler beim Abrufen des Arm-Semaphor” auftreten. In einem solchen Fall muß ein Timer eingefügt werden, bevor eine weitere Task den Arm-Semaphor abruft.

SUSPEND (Anweisung)

Funktion

Unterbricht eine Task.

Format

SUSPEND <Programmname>

Erklärung

Diese Anweisung unterbricht die Verarbeitung einer angegebenen Task. Sie kann jedoch kein Programm, das die Anweisung enthält, unterbrechen. Verwenden Sie die Anweisung HOLD, um das Unterbrechen eines Programmes mit der Anweisung zu erzwingen.

Verwandte Begriffe

KILL, HOLD

Beispiel

```
SUSPEND samp1          'Unterbricht die Task-Ausführung von samp1.
```

Anmerkungen

Wenn eine Task, für die die Bewegung mit einer SUSPEND-Anweisung unterbrochen wurde, erneut mit RUN gestartet wird, sollte sie ausgeführt werden, nachdem die Bewegung vollständig gestoppt wurde. Wenn RUN ausgeführt und der Roboter erneut bewegt wird, bevor das Stoppen der Bewegung abgeschlossen ist, kann ein Fehler wie “Befehlsgeschwindigkeitsbegrenzung überschritten” auftreten.

DEFEND (Anweisung)

Funktion

Verteidigt eine Task.

Format

DEFEND {ON|OFF}

Erklärung

Eine Programm-Task gibt normalerweise die Ausführungspriorität an ein anderes Programm mit der gleichen oder einer höheren Priorität und festen Intervallen frei.
Verwenden Sie einen DEFEND-Befehl, wenn ein Programm verarbeitet wird, ohne die Ausführungspriorität an eine andere Task freizugeben. Nach der Ausführung von DEFEND ON wird die Ausführungspriorität bis zur Ausführung von DEFEND OFF beibehalten. Bei jeder Ausführung eines Zielbefehls für DELAY, WAIT und SET IO wird die Ausführungspriorität jedoch an eine andere Programm-Task freigegeben.

Anmerkung (1): **Definieren Sie den Abschnitt eines Programmes, der mit DEFEND ON verteidigt werden soll, so kurz wie möglich. Wenn die Task nach der Ausführung von DEFEND ON in einer Endlosschleife ist, wird die Ausführungspriorität nie an andere Programm-Tasks weitergegeben.**

Anmerkung (2): **Auch bei Verteidigung einer Task mit DEFEND ON wird der Verteidigungsstatus in den folgenden Fällen automatisch freigegeben.**

- **Wenn ein END-Befehl (außer bei einem END-Befehl am Ende eines aufgerufenen Programmes) ausgeführt wird**
- **Wenn ein KILL-Befehl ausgeführt wird**
- **Wenn die Roboter-Steuerung mit dem Programmiergerät oder der E/A initialisiert wird**

Verwandte Begriffe

Beispiel

```
DEFEND ON       'Verteidigt die eigene Task.
SET IO[100]     'Die folgenden 3 Anweisungen werden beim Verteidigen einer Task
                'immer fortlaufend ausgeführt.
SET IO[102]
SET IO[104]
DEFEND OFF      'Gibt die Verteidigung der eigenen Task frei.
```

STATUS (Funktion)

Funktion

Ruft den Programmstatus ab.

Format

STATUS (<Programmname>)

Erklärung

Diese Anweisung speichert den Programmstatus des in <Programmname> angegebenen Programmes mit einer ganzen Zahl.

Wert	Status	
1	Aktiv	Ausführung
2	Stoppen	Stoppen wird ausgeführt
3	Unterbrechen	Unterbrechung wird ausgeführt
4	Verzögerung	Verzögerung wird ausgeführt
5	Anstehend	Aktuell anstehend
6	Schritt gestoppt	Stoppen des Schrittes wird ausgeführt

Verwandte Begriffe

DELAY, HALT, HOLD, STOP

Beispiel

```
DIM li1 As Integer
li1 = STATUS(samp1)  'Weist den Programmstatus von samp1 mit einer ganzen Zahl li1 zu.
```

Anmerkungen

Diese Anweisung kann den Status nicht selbständig abrufen.

14.2 Semaphor

Ein Semaphor kann zur Kommunikation (Verbindung zu einem Signal) zwischen Tasks verwendet werden, wenn mehrere Tasks synchronisiert werden (synchronisierte Steuerung) oder wenn mehrere Tasks nicht gleichzeitig betrieben werden können (exklusive Steuerung).
Erstellen Sie zur Verwendung eines Semaphor einen Semaphor mit einem CREATESEM-Befehl, um eine Semaphor-ID abzurufen. Anschließend kann ein spezifischer Semaphor unter mehreren Semaphoren angegeben werden.
Warten Sie bei Ausführung der synchronisierten oder exklusiven Steuerung, bis die Task, die nach der Ausführung eines TAKESEM-Befehls einen GIVESEM-Befehls sendet, auf die Anweisung einer anderen Task wartet. Wenn die Task, die einen Befehl sendet, bereit ist, führt sie einen GIVESEM-Befehl aus und erlaubt die Verarbeitung der Task, die auf die Ausführung eines Semaphor wartet.
Der GIVESEM-Befehl ist nur für eine Task gültig, die auf einen Semaphor wartet.
Haben mehrere Tasks einen Semaphor mit der gleichen Semaphor-ID, kann eines von zwei Warteschleifensystemen (Warten auf Ausführung) zur Verarbeitung ausgewählt werden: Reihenfolge nach dem Eingang oder der Prioritätsreihenfolge.

CREATESEM (Funktion)

Funktion

Erstellt einen Semaphor.

Format

CREATESEM (<Arithmetischer Ausdruck>)

Erklärung

Diese Funktion erstellt einen Semaphor und ruft eine Semaphor-ID auf.
Wenn keine Semaphor-ID vorhanden ist, können keine weiteren auf Semaphoren bezogenen Befehle verwendet werden. Stellen Sie daher sicher, daß dieser CREATESEM-Befehl vor der Verwendung eines Semaphor verwendet wird.
Geben Sie ein Warteschleifensystem (Warten auf Ausführung) mit einem Argument an. Legen Sie die Reihenfolge der Ausführung fest, wenn mehrere Tasks den gleichen Semaphor haben.
Die beiden folgenden Arten von Warteschleifensystemen stehen zur Verfügung.

Argument	
0	Reihenfolge des Eingangs
Nicht Null	Task-Reihenfolge nach Priorität

Die Reihenfolge nach Eingang ist die Reihenfolge, in der TAKESEM-Befehle ausgeführt werden.
Die Priorität-Ausführung wird mit einer Option (PRIORITY) des RUN-Befehls angegeben.
Sobald CREATESEM ausgeführt wurde, ändert sich der Systemstatus in den Status, in dem ein Semaphor vorhanden ist. TAKESEM kann daher ausgeführt werden, auch wenn GIVESEM nicht ausgeführt wird.
Es können bis zu 32 Semaphoren erstellt werden.

Verwandte Begriffe

DELETESEM, FLUSHSEM, GIVESEM, TAKESEM

Beispiel

```
DEFINT Li1, Li2, Li3 = 1
Li1 = CREATESEM(Li3) 'Erstellt einen Semaphor mit dem in Li3 angegebenen
                     'Warteschleifensystem und der Semaphor-ID, die in Li1 abgerufen
'wurde.
Li2 = CREATESEM(Li3) 'Erstellt einen Semaphor mit dem in Li3 angegebenen
                     'Warteschleifensystem und der Semaphor-ID, die in Li2 abgerufen
'wurde.
TAKESEM Li1          'Ruft den in Li1 angegebenen Semaphor ab.
TALESEM Li2, 100     'Ruft den in Li1 angegebenen Semaphor ab. Es tritt jedoch nach
                     '100 ms eine Zeitüberschreitung auf.
RUN samp1
GIVESEM Li1          'Gibt eine Task, die den in Li1 angegebenen Semaphor hat, aus
                     'dem Wartestatus frei.
FLUSHSEM Li2         'Gibt alle Tasks, die den in Li2 angegebenen Semaphor haben,
                     'aus dem Wartestatus frei.
DELETESEM Li1        'Löscht den Semaphor mit der in Li1 angegebenen Semaphor-ID.
DELETESEM Li2        'Löscht den Semaphor mit der in Li2 angegebenen Semaphor-ID.
```

Anmerkungen

Anmerkungen zur Verwendung einer **CREATESEM**- Anweisung:

(1) Phänomene, die bei falscher Verwendung auftreten können

Wenn eine der folgenden Aktionen ausgeführt wird, während ein Programm, das bereits über eine **CREATESEM**-Anweisung eine Semaphor-ID erstellt hat, ausgeführt oder unterbrochen wird und im Wartestatus ist, verbleibt das Programm mit der Semaphor-ID im Wartestatus, ohne daß es einen Semaphor abrufen kann.

1) Schreiben Sie mit **CREATESEM** eine Semaphor-ID-Speichervariable neu oder
2) schreiben Sie die Variable vom Programmiergerät aus gezielt neu.

(2) Beispiel

Starten Sie pro1 und führen Sie mit der STOP-Taste während der Ausführung von pro2 einen sofortigen Stop aus. Wenn das System daran anschließend neu gestartet wird, wartet pro3 unbegrenzt lange auf einen Semaphor, da die in i1 gespeicherte Semaphor-ID geändert wird.

```
PROGRAM PRO1
       i1 = CREATESEM(0)
       RUN PRO2
       RUN PRO3
END

PROGRAM PRO2
       TAKESEM i1
       .
       .
       .
       GIVESEM i1
END

PROGRAM PRO3
       TAKESEM i1
       .
       .
       .
       GIVESEM i1
END
```

(3) Es erfolgt eine Gegenmessung mit einer unbegrenzten Wartezeit.
Führen Sie nach dem Stoppen des Programmes im Endlos-Wartestatus (in diesem Beispiel pro3) ein ausführendes Programm (in diesem Beispiel ipro1) aus, um diesen Status zu beenden.

(4) Beachten Sie zur richtigen Verwendung dieser Anweisung die folgenden Punkte.
Schreiben Sie die ID eines Semaphor, der nicht im Wartestatus ist, nicht neu. Beachten Sie insbesondere die beiden folgenden Punkte.
1) Erstellen Sie einen Semaphor mit der gleichen Variablennummer immer nur einmal, solange der Semaphor nicht explizit gelöscht wird.
2) Verwenden Sie keine Variable, die eine Semaphor-ID für andere Zwecke steuert.

DELETESEM (Anweisung)

Funktion

Löscht einen Semaphor.

Format

DELETESEM <Semaphor-ID>

Erklärung

Diese Anweisung löscht einen Semaphor mit der in <Semaphor-ID> angegebenen Semaphor-ID.

Verwandte Begriffe

CREATESEM, FLUSHSEM, GIVESEM, TAKESEM

Beispiel

```
DEFINT Li1, Li2, Li3 = 1
Li1 = CREATESEM(Li3) 'Erstellt einen Semaphor mit dem in Li3 angegebenen
                     'Warteschleifensystem und der Semaphor-ID, die in Li1 abgerufen
'wurde.
Li2 = CREATESEM(Li3) 'Erstellt einen Semaphor mit dem in Li3 angegebenen
                     'Warteschleifensystem und der Semaphor-ID, die in Li2
                     'abgerufen wurde.
TAKESEM Li1          'Ruft den in Li1 angegebenen Semaphor ab.
TALESEM Li2, 100     'Ruft den in Li1 angegebenen Semaphor ab. Es tritt jedoch nach
                     '100 ms eine Zeitüberschreitung auf.
RUN samp1
GIVESEM Li1          'Gibt eine Task, die den in Li1 angegebenen Semaphor hat, aus
                     'dem Wartestatus frei.
FLUSHSEM Li2         'Gibt alle Tasks, die den in Li2 angegebenen Semaphor haben,
                     'aus dem Wartestatus frei.
DELETESEM Li1        'Löscht den Semaphor mit der in Li1 angegebenen Semaphor-ID.
DELETESEM Li2        'Löscht den Semaphor mit der in Li2 angegebenen Semaphor-ID.
```

FLUSHSEM (Anweisung)

Funktion

Gibt die auf einen Semaphor wartenden Tasks frei.

Format

FLUSHSEM <Semaphor-ID>

Erklärung

Diese Anweisung ermöglicht allen Tasks, die auf den in <Semaphor-ID> angegebenen Semaphor warten, die Fortsetzung der Verarbeitung.

Verwandte Begriffe

CREATESEM, DELETESEM, GIVESEM, TAKESEM

Beispiel

```
DEFINT Li1, Li2, Li3 = 1
Li1 = CREATESEM(Li3) 'Erstellt einen Semaphor mit dem in Li3 angegebenen
                     'Warteschleifensystem und der Semaphor-ID, die in Li1 abgerufen
'wurde.
Li2 = CREATESEM(Li3) 'Erstellt einen Semaphor mit dem in Li3 angegebenen
                     'Warteschleifensystem und der Semaphor-ID, die in Li2 abgerufen
'wurde.
TAKESEM Li1          'Ruft den in Li1 angegebenen Semaphor ab.
TALESEM Li2, 100     'Ruft den in Li1 angegebenen Semaphor ab. Es tritt jedoch nach
                     '100 ms eine Zeitüberschreitung auf.
RUN samp1
GIVESEM Li1          'Gibt eine Task, die den in Li1 angegebenen Semaphor hat, aus dem
'Wartestatus frei.
FLUSHSEM Li2         'Gibt alle Tasks, die den in Li2 angegebenen Semaphor haben,
                     'aus dem Wartestatus frei.
DELETESEM Li1        'Löscht den Semaphor mit der in Li1 angegebenen Semaphor-ID.
DELETESEM Li2        'Löscht den Semaphor mit der in Li2 angegebenen Semaphor-ID.
```

GIVESEM (Anweisung)

Funktion

Gibt eine auf einen Semaphor wartende Task frei.

Format

GIVESEM <Semaphor-ID>

Erklärung

Diese Anweisung gibt den in <Semaphor-ID> angegebenen Semaphor frei. Das System erlaubt einen Neustart eines Prozesses, wenn eine Task vorliegt, die den in <Semaphor-ID> angegebenen Semaphor bei seiner Freigabe hat. Liegt eine Task vor, die auf mehrere Semaphoren wartet, legt das System die Ausführungssequenz mit dem Warteschleifensystem fest, das beim Erstellen der Semaphoren mit dem CREATESEM-Befehl angegeben wurde.

Verwandte Begriffe

CREATESEM, DELETESEM, FLUSHSEM, TAKESEM

Beispiel

```
DEFINT Li1, Li2, Li3 = 1
Li1 = CREATESEM(Li3) 'Erstellt einen Semaphor mit dem in Li3 angegebenen
                     'Warteschleifensystem und der Semaphor-ID, die in Li1 abgerufen
'wurde.
Li2 = CREATESEM(Li3) 'Erstellt einen Semaphor mit dem in Li3 angegebenen
                     'Warteschleifensystem und der Semaphor-ID, die in Li2 abgerufen
'wurde.
TAKESEM Li1          'Ruft den in Li1 angegebenen Semaphor ab.
TALESEM Li2, 100     'Ruft den in Li1 angegebenen Semaphor ab. Es tritt jedoch nach
                     '100 ms eine Zeitüberschreitung auf.
RUN samp1
GIVESEM Li1          'Gibt eine Task, die den in Li1 angegebenen Semaphor hat, aus
                     'dem Wartestatus frei.
FLUSHSEM Li2         'Gibt alle Tasks, die den in Li2 angegebenen Semaphor haben,
                     'aus dem Wartestatus frei.
DELETESEM Li1        'Löscht den Semaphor mit der in Li1 angegebenen Semaphor-ID.
DELETESEM Li2        'Löscht den Semaphor mit der in Li2 angegebenen Semaphor-ID.
```

TAKESEM (Anweisung)

Funktion

Ruft einen Semaphor mit einer angegenen Semaphor-ID ab.

Format

TAKESEM <Semaphor-ID>[,<Zeitüberschreitungswert>]

Erklärung

Diese Anweisung ruft den in <Semaphor-ID> angegebenen Semaphor ab.
Wenn eine andere Task einen Semaphor abruft, wartet das System, bis der Semaphor freigegeben wird, bevor der Semaphor abgerufen wird.
Die Wartezeit kann in Millisekunden (ms) mit der Option <Zeitüberschreitungswert> angegeben werden. Wenn das System innerhalb der angegebenen Zeit keinen Semaphor abrufen kann, tritt ein Fehler auf.

Verwandte Begriffe

CREATESEM, DELETESEM, FLUSHSEM, GIVESEM

Beispiel

```
DEFINT Li1, Li2, Li3 = 1
Li1 = CREATESEM(Li3) 'Erstellt einen Semaphor mit dem in Li3 angegebenen
                     'Warteschleifensystem und der Semaphor-ID, die in Li1 abgerufen
'wurde.
Li2 = CREATESEM(Li3) 'Erstellt einen Semaphor mit dem in Li3 angegebenen
                     'Warteschleifensystem und der Semaphor-ID, die in Li2 abgerufen
'wurde.
TAKESEM Li1          'Ruft den in Li1 angegebenen Semaphor ab.
TALESEM Li2, 100     'Ruft den in Li1 angegebenen Semaphor ab. Es tritt jedoch nach
                     '100 ms eine Zeitüberschreitung auf.
RUN samp1
GIVESEM Li1          'Gibt eine Task, die den in Li1 angegebenen Semaphor hat, aus
                     'dem Wartestatus frei.
FLUSHSEM Li2         'Gibt alle Tasks, die den in Li2 angegebenen Semaphor haben,
                     'aus dem Wartestatus frei.
DELETESEM Li1        'Löscht den Semaphor mit der in Li1 angegebenen Semaphor-ID.
DELETESEM Li2        'Löscht den Semaphor mit der in Li2 angegebenen Semaphor-ID.
```

Anmerkungen

Wenn die Option “Zeitüberschreitungswert” verwendet wird, beachten Sie, daß die Wartezeit auch während der Unterbrechung nach einem sofortigen Stop während der Ausführung eines Befehls und einem anschließenden Neustart des Roboters verstreicht.

14.3 Arm-Semaphor

TAKEARM (Anweisung)

Funktion

Ruft die Priorität der Robotersteuerung ab, gibt eine Werkzeugkoordinate und eine Arbeitskoordinate an die Ausgangsposition zurück und stellt die interne Geschwindigkeit, Beschleunigung und Verzögerung auf 100 ein.

Format

TAKEARM [<KEEP=Einstellungswert für die Initialisierung>]

Erklärung

Diese Anweisung setzt den Prozeß unverändert fort, wenn ein TAKEARM-Befehl in einer Task, die bereits eine Steuerpriorität hat, oder in einem von der Task aufgerufenen Unterprogramm ausgeführt wird.
Es tritt ein Fehler auf, wenn eine Task ohne Roboter-Steuerungspriorität versucht, eine der Roboterbewegungsanweisungen in der folgenden Tabelle auszuführen. Stellen Sie sicher, daß Sie mit einem TAKEARM-Befehl eine Steuerpriorität für Programme abrufen, die zur Ausführung dieser Bewegungsanweisungen verwendet werden.

Roboter-Bewegungsanweisungen, die eine Steuerpriorität erfordern

Typ	Befehle
Deklarationsanweisung	HOME, TOOL, WORK
Robotersteuerungsanweisung	APPROACH, DEPART, DRAW, DRIVE, DRIVEA, GOHOME, MOVE, ROTATEH, ROTATE, SPEED, JSPEED, ACCEL, JACCEL, DECEL, JDECEL, CHANGETOOL, CHANGEWORK, LETENV Bibliothek zur richtigen Einstellung der beweglichen Masse, Armbewegungsbibliothek

Anmerkung 1: Die Roboter-Steuerungspriorität wird in den folgenden Fällen automatisch freigegeben:
- **Wenn ein END-Befehl (außer bei einem END-Befehl am Ende eines aufgerufenen Programmes) ausgeführt wird**
- **Wenn ein KILL-Befehl ausgeführt wird**
- **Wenn ein HALT-Befehl ausgeführt wird**
- **Wenn ein STOP-Befehl ausgeführt wird**
- **Wenn die Roboter-Steuerung mit dem Programmiergerät oder der E/A initialisiert wird**

Anmerkung 2: Bei Ausführung eines TAKEARM-Befehls führt das System automatisch die folgenden Prozesse aus:
- **Die Werkzeugdefinition wird auf WERKZEUG 0 initialisiert. Die Arbeitskoordinaten werden auf die Basiskoordinaten des Roboters eingestellt.**
- **Die interne Geschwindigkeit, die interne Beschleunigung und die interne Verzögerung werden auf 100 eingestellt.**

Anmerkung 3: Das System ermöglicht keine Freigabe während der Unterbrechung des Programmes.

<KEEP = Einstellwert für die Initialisierung > [V1.4 oder neuer]

Einstellungswert für die Initialisierung

0: Die Werkzeugkoordinate und die Arbeitskoordinate werden wieder auf die Ausgangsposition zurückgesetzt, und die interne Geschwindigkeit, die interne Beschleunigung und die interne Verzögerung werden auf 100 eingestellt.

1: Die Werkzeugkoordinate, die Arbeitskoordinate, die interne Geschwindigkeit, die interne Beschleunigung und die interne Verzögerung werden mit ihrer aktuellen Einstellung beibehalten.

Wenn <KEEP = Einstellungswert für die Initialisierung> weggelassen wird, wird KEEP=0 verwendet (die Werkzeugkoordinate und die Arbeitskoordinate werden auf den Ausgangswert zurückgesetzt, und die interne Geschwindigkeit, die interne Beschleunigung und die interne Verzögerung werden auf 100 eingestellt).

Verwandte Begriffe

GIVEARM, TAKEVIS, GIVEVIS

Beispiel

```
Beispiel 1:
    PROGRAM PRO1        '
    TAKEARM             'Führt TAKEARM in der ersten Zeile des Programmes aus, das die
                        'Anweisungen zur Roboterbewegung enthält.
    MOVE P, P1          '
    MOVE P, P2          '
    GIVEARM 1           'Gibt die Roboter-Steuerpriorität mit GIVEARM frei, sobald sie
                        'abgeschlossen ist.
                        'Dies ist nicht immer erforderlich, da die Priorität mit END unmittelbar
                        'nach dieser Anweisung automatisch freigegeben wird.
    END                 '

Beispiel 2:
    PROGRAM PRO2        '
    MOVE P, P3          '×: Wenn eine MOVE-Anweisung ausgeführt wird ohne Ausführung von
                        'TAKEARM, tritt ein Fehler auf.
                        '
    END                 '

Beispiel 3:
    PROGRAM PRO3        '
    TAKEARM             '
    MOVE P, P4          '
    CALL SUB1           '
    END                 '
    PROGRAM SUB1        '
    MOVE P, P5          'Es tritt kein Fehler auf, da TAKEARM bereits von PRO3 ausgeführt
    'wurde.
    END                 '

Beispiel 4:
    PROGRAM PRO4        '
    TAKEARM             '
    SPEED 50            '
    CHANGEWORK 3        '
    CHANGETOOL 1        '
    MOVE P, P5          '
    CALL PRO5           '
    END                 '

    PROGRAM PRO5        '
    TAKEARM             'Es tritt kein Fehler auf, auch wenn TAKEARM zwei Mal
                        'von PRO4 aufgerufen wird, da der Aufruf als Unterprogramm von PRO4
erfolgt.
    MOVE P, P6          'Die Werkzeugkoordinaten und die Arbeitskoordinaten sind jedoch
                        'Null, und die interne Geschwindigkeit ist 100.
    END                 '
```

Beispiel 5:

```
    PROGRAM PRO6          '
    TAKEARM               '
    RUN PRO7              '
    MOVE P, P7            '
    END                   '

    PROGRAM PRO7          '
    TAKEARM               '×: Es tritt ein Fehler auf bei dem Versuch, die
                          'Roboter-Ausführungspriorität
                          'mit PRO7 abzurufen, die als eine weitere Task von
                          'PRO6 ausgeführt wird, da PRO6 bereits eine
                          'Roboter-Steuerungspriorität hat.
    MOVE P, P7            '
    END                   '
```

GIVEARM (Anweisung)

Funktion

Gibt die Priorität der Robotersteuerung frei.

Format

GIVEARM

Erklärung

Diese Anweisung gibt die Roboter-Steuerungspriorität frei. Damit können andere Tasks eine Steuerungs-Task abrufen.

Wenn versucht wird, einen GIVEARM-Befehl auszuführen, obwohl eine andere Task bereits eine Roboter-Steuerungspriorität abgerufen hat, tritt ein Fehler auf.

Anmerkung: Die Roboter-Steuerungspriorität wird in den folgenden Fällen automatisch freigegeben. In diesen Fällen muß daher kein GIVEARM-Befehl angegeben werden.

- **Wenn ein END-Befehl ausgeführt wird (außer bei einem END-Befehl am Ende des aufgerufenen Programmes)**
- **Wenn ein KILL-Befehl ausgeführt wird**
- **Wenn die Roboter-Steuerung mit dem Programmiergerät oder der E/A initialisiert wird**

Verwandte Begriffe

TAKEARM, TAKEVIS, GIVEVIS

Beispiel

```
TAKEARM         'Führt TAKEARM am Anfang des Programmes aus, das die Roboterbewegung ausführt.
MOVE P, lp1     '
MOVE P, lp2     '
GIVEARM         'Gibt die Roboter-Steuerpriorität mit GIVEARM frei, sobald 1 abgeschlossen ist.
                'Diese Ausführung ist nicht immer erforderlich, da der Roboter bei END unmittelbar
                'danach automatisch freigegeben wird.
```

Anmerkungen

Das System wartet mit der Ausführung von GIVEARM, bis der Roboter vollständig gestoppt wurde. Auch wenn eine Durchlaufbewegung vor GIVEARM angegeben ist, wird die Durchlaufbewegung nicht ausgeführt.

TAKEVIS (Anweisung)

Funktion

Ruft die Priorität der visuellen Verarbeitung ab.

Format

TAKEVIS

Erklärung

Diese Anweisung ruft die viselle Prozeßpriorität auf der in der Robotersteuerung optional installierten Bildverarbeitungskarte ab. Ohne installierte Bildverarbeitungskarte tritt bei Ausführung des TAKEVIS-Befehls ein Fehler auf. Wenn bereits eine Task eine visuelle Prozeßpriorität abgerufen hat und das System diese nicht abrufen kann, tritt ein Fehler auf.
Wenn eine Task, die keine visuelle Prozeßpriorität abruft, eine visuelle Anweisung auszuführen versucht, tritt ein Fehler auf. In einem Programm, in dem eine visuelle Anweisung ausgeführt wird, muß vor der visuellen Anweisung mit dem TAKEVIS-Befehl eine visuelle Prozeßpriorität abgerufen werden.
Wenn ein TAKEVIS-Befehl in einer Task mit visueller Prozeßpriorität ausgeführt wird oder in einem von der Task aufgerufenen Unterprogramm, wird die Verarbeitung ohne Änderung fortgesetzt.

Anmerkung: Die visuelle Steuerungspriorität wird in den folgenden Fällen automatisch freigegeben.

- **Wenn ein END-Befehl (außer bei einem END-Befehl am Ende des aufgerufenen Programmes) ausgeführt wird**
- **Wenn ein KILL-Befehl ausgeführt wird**
- **Wenn die Roboter-Steuerung mit dem Programmiergerät oder der E/A initialisiert wird**

Verwandte Begriffe

TAKEARM, GIVEARM, GIVEVIS

Beispiel

```
DEFINT Li1, Li2                         'Ruft eine visuelle Prozeßpriorität ab.
TAKEVIS
VISSCREEN 1, 0, 1
VISCLS
FOR Li1 = 0 TO 255
   Li2 = VISREFTABLE(1, Li1)            'Ruft Daten aus Tabelle 1 ab.
   VISLOC 10, 10                        'Gibt die Anzeigeposition an.
   VISDEGCHAR 1, 1, 2                   'Gibt das Anzeigezeichen an.
   VISPRINT ”Data”; Li1; ”=”; Li2       'Anzeigen.
NEXT Li1
GIVEVIS                                 'Gibt die visuelle Prozeßpriorität frei.
```

GIVEVIS (Anweisung)

Funktion

Gibt die Priorität der visuellen Verarbeitung frei.

Format

GIVEVIS

Erklärung

Diese Anweisung gibt die visuelle Prozeßpriorität auf der in der Robotersteuerung optional installierten Bildverarbeitungskarte frei. Damit kann eine andere Task eine visuelle Prozeßpriorität abrufen. Ohne installierte Bildverarbeitungskarte tritt bei Ausführung des GIVEVIS-Befehls ein Fehler auf. Wenn versucht wird, einen GIVEVIS-Befehl auszuführen, obwohl eine andere Task bereits eine visuelle Prozeßpriorität abgerufen hat, tritt ein Fehler auf.

Anmerkung: Die Roboter-Steuerungspriorität wird in den folgenden Fällen automatisch freigegeben. Der GIVEVIS-Befehl braucht daher in diesen Fällen nicht angegeben zu werden.

- **Wenn ein END-Befehl (außer bei einem END-Befehl am Ende des aufgerufenen Programmes) ausgeführt wird**
- **Wenn ein KILL-Befehl ausgeführt wird**
- **Wenn die Roboter-Steuerung mit dem Programmiergerät oder der E/A initialisiert wird**

Verwandte Begriffe

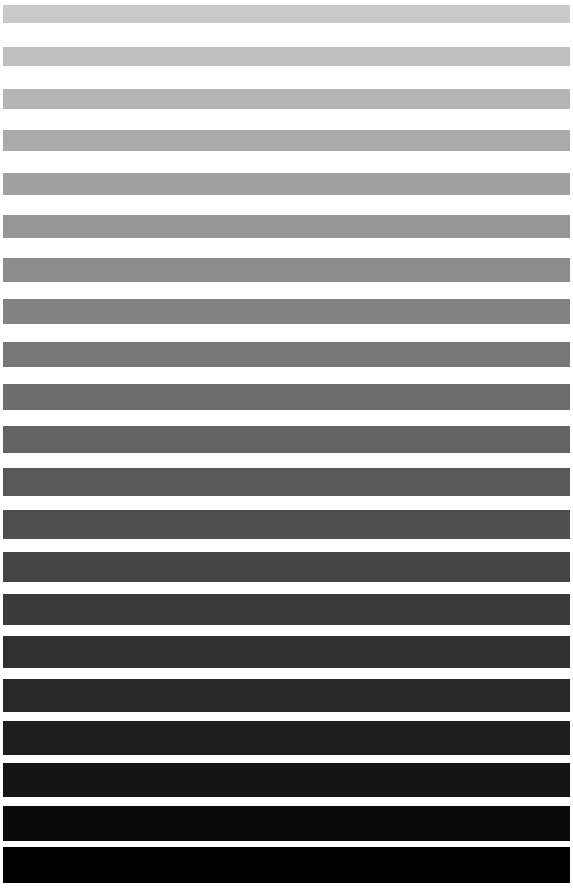
TAKEARM, GIVEARM, TAKEVIS

Beispiel

```
DEFINT Li1, Li2                        'Ruft eine visuelle Prozeßpriorität ab.
TAKEVIS
VISSCREEN 1, 0, 1
VISCLS
FOR Li1 = 0 TO 255
   Li2 = VISREFTABLE (1, Li1)          'Ruft Daten aus Tabelle 1 ab.
   VISLOC 10, 10                       'Gibt die Anzeigeposition an.
   VISDEGCHAR 1, 1, 2                  'Gibt das Anzeigezeichen an.
   VISPRINT ”Data”; Li1; ”=”; Li2      'Anzeigen.
NEXT Li1
GIVEVI                                 'Gibt die visuelle Prozeßpriorität frei.
```

Kapitel 15

Funktionen



Dieses Kapitel enthält eine Erläuterung zu den verschiedenen in PAC vorbereiteten Funktionen.

15.1 Arithmetische Funktion

ABS (Funktion) [Entspricht SLIM]

Funktion

Ruft den absoluten Wert eines Ausdruckwertes ab.

Format

ABS (<Ausdruck>)

Erklärung

Diese Anweisung ruft den absoluten Wert des in <Ausdruck> angegebenen Wertes ab.

Wenn <Ausdruck> eine Real-Zahl mit doppelter Genauigkeit enthält, erhält der abgerufene Wert doppelte Genauigkeit. Andernfalls wird eine einfache Genauigkeit verwendet.

Erklärung:	**Der absolute Wert ist ein numerischer Wert, von dem das Vorzeichen entfernt wurde. Sowohl ABS (-1) als auch ABS (1) geben 1 zurück.**

Verwandte Begriffe

Beispiel

```
DEFSNG lf1, lf2, lf3, lf4
lf1 = ABS(-2)                  'Weist den absoluten Wert von –2 der Variablen
                               'If1 zu.
lf2 = ABS(lf3/lf4)             'Weist den absoluten Wert von (lf3/lf4) der
                               'Variablen lf2 zu.
```

EXP (Funktion) [Entspricht SLIM]

Funktion

Ruft eine Exponentialfunktion mit einem natürlichen Logarithmus als Basis auf.

Format

EXP (<Ausdruck>)

Erklärung

Wenn <Ausdruck> eine Real-Zahl mit doppelter Genauigkeit enthält, erhält der abgerufene Wert doppelte Genauigkeit. Andernfalls wird eine einfache Genauigkeit verwendet.

Erklärung:	**Wenn der Wert eines Argumentes 709,782712893 überschreitet, tritt ein Überlauf auf. Die Konstante "e" beträgt ca. 2,718282.**
Hinweis:	**Die EXP-Funktion ist die Umkehrfunktion des Logarithmus; sie wird häufig als inverser Logarithmus bezeichnet.**

Verwandte Begriffe

LOG, LOG10

Beispiel

```
DEFSNG lf1, lf2, lf3, lf4
lf1 = EXP(2)         'Weist die zweite Potenz der natürlichen Logarithmusbasis
                     'der Variablen lf1 zu.
lf2 = EXP(lf3/lf4)   'Weist die (lf3/lf4) Potenz der natürlichen Logarithmusbasis der
                     'Variablen lf2 zu.
```

INT (Funktion) [Entspricht SLIM]

Funktion

Ruft den maximalen Wert aus einem angegebenen Wert ab.

Format

INT (<Ausdruck>)

Erklärung

Diese Anweisung gibt die maximale ganze Zahl zurück, die den Wert von <Ausdruck> nicht überschreitet.
Wenn der Wert einen Überlauf ergibt, wird die maximale ganze Zahl mit dem gleichen Vorzeichen wie <Ausdruck> verwendet.

Beispiel

```
DIM li1 As Integer
li1 = INT(123.456)      'Rundet nach dem Dezimalzeichen ab und weist den Wert
                        'der ganzzahligen Variablen li1 zu.
                        '123,456->123
```

LOG (Funktion) [Entspricht SLIM]

Funktion

Ermittelt einen natürlichen Logarithmus.

Format

LOG (<Ausdruck>)

Erklärung

Diese Anweisung ruft den natürlichen Logarithmus des in <Ausdruck> angegebenen Wertes ab.

Geben Sie für <Numerischer Wert> einen Wert größer als Null an.

Wenn <Ausdruck> eine Real-Zahl mit doppelter Genauigkeit enthält, erhält der abgerufene Wert doppelte Genauigkeit. Andernfalls wird eine einfache Genauigkeit verwendet.

Erklärung: **Der natürliche Logarithmus ist ein Logarithmus mit der Konstanten "e" als Basis. Der Wert der Konstanten "e" beträgt ca. 2,718282. Der Logarithmus eines beliebigen numerischen wertes x mit n als Basis kann abgerufen werden durch Dividieren des natürlichen Logarithmus von x durch den natürlichen Logarithmus von n, wie nachfolgend gezeigt.**

$Log_n x = Log_e x / Log_e n$

Verwandte Begriffe

EXP, LOG10

Beispiel

```
DEFSNG lf1, lf2, lf3, lf4
lf1 = LOG(2)          'Weist den natürlichen Logarithmus von 2 der Variablen
                      'lf1 zu.
lf2 = LOG(lf3/lf4)    'Weist den natürlichen Logarithmus von (lf3/lf4) der
                      'Variablen lf2 zu.
```

LOG10 (Funktion) [Entspricht SLIM]

Funktion

Ermittelt einen allgemeinen Logarithmus.

Format

LOG10 (<Ausdruck>)

Erklärung

Diese Anweisung ruft den Zehnerlogarithmus des in <Ausdruck> angegebenen Wertes ab. Die Basis dieses Logarithmus ist 10.

Geben Sie einen Wert größer als Null für <Numerischer Wert> ein.

Wenn <Ausdruck> eine Real-Zahl mit doppelter Genauigkeit enthält, erhält der abgerufene Wert doppelte Genauigkeit. Andernfalls wird eine einfache Genauigkeit verwendet.

Verwandte Begriffe

EXP, LOG

Beispiel

```
DEFSNG lf1, lf2, lf3, lf4
lf1 = LOG10(2)          'Weist den Zehnerlogarithmus von 2 der Variablen lf1 zu.
lf2 = LOG10(lf3/lf4)    'Weist den Zehnerlogarithmus von (lf3/lf4) der Variablen
                        'lf2 zu.
```

POW (Funktion) [Entspricht SLIM]

Funktion

Ermittelt einen Exponenten.

Format

POW (<Basis>,<Exponent>)

Erklärung

Diese Anweisung ruft den Exponenten des mit <Basis> von <Exponent> angegebenen Wertes ab.

Wenn <Basis> oder <Exponent> eine Real-Zahl mit doppelter Genauigkeit enthält, erhält der abgerufene Wert doppelte Genauigkeit. Andernfalls wird eine einfache Genauigkeit verwendet.

Verwandte Begriffe

EXP

Beispiel

```
DEFSNG lf1, lf2
DEFINT li1, li2
li1 = POW(5, 2)         'Weist die zweite Potenz von 5 der Variablen li1 zu.
lf1 = POW(lf2, li2)     'Weist die li2. Potenz von lf2 der Variablen lf1 zu.
```

Anmerkungen

In einer exponentiellen Berechnung tritt ein Bereichsfehler auf, wenn das erste Argument negativ ist.
Wenn <Basis> und <Exponent> Real-Zahlen mit doppelter Genauigkeit enthalten, ist eine Genauigkeit bis zu 15 Stellen garantiert.
Wenn <Basis> und <Exponent> keine Real-Zahlen mit doppelter Genauigkeit enthalten, ist die Genauigkeit nur bis zu 7 Stellen garantiert.

MAX (Funktion) [Entspricht SLIM]

Funktion

Extrahiert den maximalen Wert.

Format

MAX (<Ausdruck>,<Ausdruck>[,<Ausdruck>…])

Erklärung

Diese Anweisung extrahiert den maximalen Wert aus einer beliebigen Anzahl von <Ausdrücken>.
Die maximale Anzahl von Ausdrücken ist 32.

Erklärung:	**Durch Verwendung der Funktionen MIN und MAX kann der minimale oder maximale Wert der gewünschten Akkumulationsmethode oder der Gruppenfelder abgerufen werden. Mit dieser Funktion können beispielsweise die maximalen und minimalen Versandkosten ermittelt werden. Wenn keine Akkumulationsmethode angegeben ist, wird die gesamte Tabelle zu einem Objekt.**

Verwandte Begriffe

MIN

Beispiel

```
DEFSNG lf1, lf2, lf3, lf4, lf5, lf6
DEFINT li1, li2, li3, li4, li5, li6
li1 = MAX(1, 2)                      'Weist den maximalen Wert aus (1, 2) der
                                     'Variablen li1 zu.
li2 = MAX(li3, li4, li5, li6)        'Weist den maximalen Wert aus (li3, li4,
                                     'li5, li6) der Variablen li2 zu.
lf1 = MAX(lf2, lf3, lf4, lf5, lf6)   'Weist den maximalen Wert aus (lf2, lf3,
                                     'lf4, lf5, lf6) der Variablen lf1 zu.
```

MIN (Funktion) [Entspricht SLIM]

Funktion

Extrahiert den minimalen Wert.

Format

MIN (<Ausdruck>,<Ausdruck>[,<Ausdruck>…])

Erklärung

Diese Anweisung extrahiert den minimalen Wert aus einer beliebigen Anzahl von <Ausdrücken>.
Die minimale Anzahl von Ausdrücken ist 32.

Erklärung:	**Durch Verwendung der Funktionen MIN und MAX kann der minimale oder maximale Wert der gewünschten Akkumulationsmethode oder der Gruppenfelder abgerufen werden. Mit dieser Funktion können beispielsweise die maximalen und minimalen Versandkosten ermittelt werden. Wenn keine Akkumulationsmethode angegeben ist, wird die gesamte Tabelle zu einem Objekt.**

Verwandte Begriffe

MAX

Beispiel

```
DEFSNG lf1, lf2, lf3, lf4, lf5, lf6
DEFINT li1, li2, li3, li4, li5, li6
li1 = MIN(1, 2)                      'Weist den minimalen Wert aus (1, 2) der Variablen
                                     'li1 zu.
li2 = MIN(li3, li4, li5, li6)        'Weist den minimalen Wert aus (li3, li4,
                                     'li5, li6) der Variablen li2 zu.
lf1 = MIN(lf2, lf3, lf4, lf5, lf6)   'Weist den minimalen Wert aus (lf2, lf3,
                                     'lf4, lf5, lf6) der Variablen lf1 zu.
```

RND (Funktion)

Funktion

Generiert Zufallszahlen von 0 bis 1.

Format

RND (<Ausdruck>)

Erklärung

Entsprechend dem Wert in <Ausdruck> variiert die Verarbeitung wie in der folgenden Tabelle gezeigt.
<Ausdruck> gilt für ganzzahlige Werte. Bei Angabe eines Real-Wertes wird dieser abgerundet und in eine ganze Zahl umgewandelt.

Wert des Ausdrucks	Verarbeitung
Ausdruck < 0	<Ausdruck> wird als Ausgangswert für die Zufallszahl verwendet. Wenn der Ausgangswert identisch ist, gibt die RND-Funktion immer den gleichen Wert zurück.
Ausdruck = 0	Es wird eine zuvor generierte Zufallszahl verwendet. Es wird Null zurückgegeben, wenn keine Zufallszahl generiert wurde, nachdem die Stromversorgung der Robotersteuerung eingeschaltet wurde.
Ausdruck > 0	Die nächste Zufallszahl in der Reihe der Zufallszahlen wird generiert.

Erklärung: **Für die RND-Funktion wird ein Wert zwischen 0 und 1 (je einschließlich) zurückgegeben. Je nach dem Wert des Arguments variiert die von der RND-Funktion zurückgegebene Zufallszahl. Solange der anfängliche Ausgangswert gleich bleibt, ist die Reihe der von den RND-Funktionen zurückgegebenen Zufallszahlen gleich. Das liegt daran, daß jede folgende RND-Funktion beim Generieren der nächsten Zufallszahl die von der vorigen RND-Funktion generierte Zufallszahl als Ausgangswert verwendet.**

Bemerkung: **Wenn der Ausgangswert in der Zufallszahl gleich ist, ist auch die Reihe der Zufallszahlen gleich. Wenn Sie jedes Mal unterschiedliche Zufallszahlen erhalten wollen, verwenden Sie den Rückgabewert der TIMER-Funktion als Ausgangswert.**

Beispiel

```
DIM array(10) As Single
array(0) = RND(-TIMER) 'Der Ausgangswert für die Zufallszahl wird mit der
                       'TIMER-Funktion gesetzt.
FOR I1 = 1 TO 9
  array(I1) = RND(1)   'Ruft Zufallszahlen ab.
NEXT I1
```

SGN (Funktion)

Funktion

Überprüft ein Vorzeichen.

Format

SGN (<Ausdruck>)

Erklärung

Diese Anweisung prüft das Vorzeichen von <Ausdruck> und gibt den folgenden numerischen Wert zurück.

Wenn positiv	1
Wenn 0	0
Wenn negativ	-1

Beispiel

```
DEFSNG lf1, lf2
lf2 = SGN (lf1)        'Gibt das Vorzeichen der Real-Variablen lf1 zurück.
```

SQR (Funktion) [Entspricht SLIM]

Funktion

Ermittelt die Quadratwurzel.

Format

SQR (<Ausdruck>)

Erklärung

Diese Anweisung ruft die Quadratwurzel des Wertes in <Ausdruck> ab.

Wenn <Ausdruck> eine Real-Zahl mit doppelter Genauigkeit enthält, erhält der abgerufene Wert doppelte Genauigkeit. Andernfalls wird eine einfache Genauigkeit verwendet.

<Ausdruck> muß ein Wert größer oder gleich Null sein.

Beispiel

```
DEFSNG lf1, lf2, lf3, lf4, lf5, lf6, lf7
lf1 = SQR (2)                  'Weist die Quadratwurzel von 2 der Variablen
                               'lf1 zu.
lf2 = SQR (lf4)                'Weist die Quadratwurzel von lf4 der Variablen
                               'lf2 zu.
lf3 = SQR (lf5 + lf6) * lf7    'Weist den Wert der Quadratwurzel von (lf5 +
                               'lf6), multipliziert mit lf7, der Variablen
                               'lf3 zu.
```

15.2 Trigonometrische Funktion

ACOS (Funktion) [Entspricht SLIM]

Funktion

Ermittelt einen Arkuskosinus.

Format

ACOS (<Ausdruck>)

Erklärung

Diese Anweisung ruft den Arkuskosinus des Wertes in <Ausdruck> ab.
Der abgerufene Wert wird in Grad angegeben; er kann im Bereich von 0 bis 180 liegen.

Wenn <Ausdruck> eine Real-Zahl mit doppelter Genauigkeit enthält, erhält der abgerufene Wert doppelte Genauigkeit. Andernfalls wird eine einfache Genauigkeit verwendet.

Verwandte Begriffe

SIN, TAN, ASIN, ATN, ATN2

Beispiel

```
DEFSNG lf1, lf2, lf3, lf4, lf5, lf6, lf7
lf1 = ACOS (0.5)               'Weist den Wert des Arkuskosinus von 0.5 der
                               'Variablen lf1 zu.
lf2 = ACOS (lf4/2)             'Weist den Wert des Arkuskosinus von (lf4/2) der
                               'Variablen lf2 zu.
lf3 = ACOS (lf5/lf6) * lf7     'Weist den Wert des Arkuskosinus von (lf5/lf6),
                               'multipliziert mit lf7, der Variablen lf3 zu.
```

ASIN (Funktion) [Entspricht SLIM]

Funktion

Ermittelt einen Arkussinus.

Format

ASIN (<Ausdruck>)

Erklärung

Diese Anweisung ruft den Arkussinus des Wertes in <Ausdruck> ab.
Der abgerufene Wert wird in Grad angegeben; er kann im Bereich von -90 bis 90 liegen.

Wenn <Ausdruck> eine Real-Zahl mit doppelter Genauigkeit enthält, erhält der abgerufene Wert doppelte Genauigkeit. Andernfalls wird eine einfache Genauigkeit verwendet.

Verwandte Begriffe

SIN, COS, TAN, ACOS, ATN, ATN2

Beispiel

```
DEFSNG lf1, lf2, lf3, lf4, lf5, lf6, lf7
lf1 = ASIN (0.5)               'Weist den Wert des Arkussinus von 0.5 der
                               'Variablen lf1 zu.
lf2 = ASIN (lf4/2)             'Weist den Wert des Arkussinus von (lf4/2) der
                               'Variablen lf2 zu.
lf3 = ASIN (lf5/lf6) * lf7     'Weist den Wert des Arkussinus von (lf5/lf6),
                               'multipliziert mit lf7, der Variablen lf3 zu.
```

ATN (Funktion) [Entspricht SLIM]

Funktion

Ermittelt einen Arkustangens.

Format

ATN (<Ausdruck>)

Erklärung

Diese Anweisung ruft den Arkustangens des Wertes in <Ausdruck> ab.
Der abgerufene Wert wird in Grad angegeben; er kann im Bereich von -90 bis 90 liegen.

Wenn <Ausdruck> eine Real-Zahl mit doppelter Genauigkeit enthält, erhält der abgerufene Wert doppelte Genauigkeit. Andernfalls wird eine einfache Genauigkeit verwendet.

Erklärung:	**Die ATN-Funktion empfängt das Verhältnis von 2 Seiten eines rechtwinkligen Dreiecks als Argument (Zahl) und gibt den entsprechenden Winkel zurück. Die beiden hier angegebenen Seiten schließen den rechten Winkel ein. Das Verhältnis der beiden Seiten ist der Wert der dem Winkel gegenüberliegenden Seite (Gegenkathede) zur an dem Winkel anliegenden Seite (Ankathede). Der Rückgabewert reicht von -π/2 bis π/2 (in Radianten). Multiplizieren Sie zur Umwandlung von Grad in Radianten die Gradangabe mit π/180. Multiplizieren Sie zur Umwandlung von Radianten in Grad die Radiantenangabe mit 180/π.**
Hinweis:	**Die ATN-Funktion ist eine trigonometrische Funktion, die die Umkehrung der Tan-Funktion darstellt. Die Tan-Funktion empfängt einen Winkel als Argument und gibt das Verhältnis der beiden Seiten, die den rechten Winkel eines rechtwinkligen Dreiecks einschließen, zurück. Beachten Sie den Unterschied zwischen der ATN-Funktion und dem Kotangens (1/Tangens) des reziproken Tangens.**

Verwandte Begriffe

SIN, COS, TAN, ASIN, ACOS, ATN2

Beispiel

```
DEFSNG lf1, lf2, lf3, lf4, lf5, lf6, lf7
lf1 = ATN(0,5)                  'Weist den Wert des Arkustangens von 0,5 der
                                'Variablen lf1 zu.
lf2 = ATN(lf4/2)                'Weist den Wert des Arkustangens von (lf4/2) der
                                'Variablen lf2 zu.
lf3 = ATN(lf5/lf6) * lf7        'Weist den Wert des Arkustangens von (lf5/lf6),
                                'multipliziert mit lf7, der Variablen lf3 zu.
```

ATN2 (Funktion) [Entspricht SLIM]

Funktion

Ermittelt den Arkustangens von Ausdruck 1, geteilt durch Ausdruck 2.

Format

ATN2 (<Ausdruck1>, <Ausdruck2>)

Erklärung

Diese Anweisung ruft den Wert des Arkustangens von <Ausdruck1>, dividiert durch <Ausdruck2>, ab.
Der abgerufene Wert wird in Grad angegeben; er kann im Bereich von -180 bis 180 liegen.

Wenn <Ausdruck> eine Real-Zahl mit doppelter Genauigkeit enthält, erhält der abgerufene Wert doppelte Genauigkeit. Andernfalls wird eine einfache Genauigkeit verwendet.

Die folgende Angabe kennzeichnet einen Wertebereich von ATN2.

	Erster Quadrant	**Zweiter Quadrant**	**Dritter Quadrant**	**Vierter Quadrant**
Unterer Grenzwert	0	90	-180	-90
Oberer Grenzwert	90	180	90	- 0

Verwandte Begriffe

SIN, COS, TAN, ASIN, ACOS, ATN

Beispiel

```
DEFSNG lf1, lf2, lf3, lf4, lf5, lf6, lf7, lf8
lf1 = ATN2(1, 2)             'Weist den Wert des Arkustangens von (1, 2) der
                             'Variablen lf1 zu.
lf2 = ATN2(lf4, lf5)         'Weist den Wert des Arkustangens von (lf4, lf5) der
                             'Variablen lf2 zu.
lf3 = ATN2(lf6, lf7) * lf8   'Weist den Wert des Arkustangens von (lf6, lf7),
                             'multipliziert mit lf8, der Variablen lf3 zu.
```

COS (Funktion) [Entspricht SLIM]

Funktion

Ermittelt einen Kosinus.

Format

COS (<Ausdruck>)

Erklärung

Diese Anweisung ruft den Kosinus des Wertes von <Ausdruck> ab.

Geben Sie <Ausdruck> in Grad an.

Wenn <Ausdruck> eine Real-Zahl mit doppelter Genauigkeit enthält, erhält der abgerufene Wert doppelte Genauigkeit. Andernfalls wird eine einfache Genauigkeit verwendet.
Wenn ein Argument in Radianten angegeben wird, fügen Sie RAD und die Konstante hinzu.

Erklärung:	**Die COS-Funktion empfängt einen Winkel als Argument und gibt das Verhältnis der beiden Seiten, die den rechten Winkel eines rechtwinkligen Dreiecks einschließen, zurück. Die beiden Seiten schließen den in dem Argument angegebenen Winkel ein. Das Verhältnis der beiden Seiten ist die Länge der angrenzenden Seite (Ankathede) zur Länge der Hypotenuse. Der Wertebereich der Rückgabewerte reicht von -1 bis 1. Zum Umwandeln von Grad in Radianten multiplizieren Sie die Gradangabe mit π/180. Zum Umwandeln von Radianten in Grad multiplizieren Sie die Radiantenangabe mit 180/π.**

Verwandte Begriffe

SIN, TAN, ASIN, ACOS, ATN, ATN2

Beispiel

```
DEFSNG lf1, lf2, lf3, lf4, lf5
lf1 = COS(0,78525)          'Weist den Wert des Kosinus von 0,78525 der
                            'Variablen lf1 zu.
lf2 = COS(lf4)              'Weist den Wert des Kosinus von lf4 der Variablen
                            'lf2 zu.
lf3 = COS(45) * lf5         'Weist den Wert des Kosinus von 45 Grad,
                            'mulitpliziert mit lf5, der Variablen lf3 zu.
```

SIN (Funktion) [Entspricht SLIM]

Funktion

Ermittelt einen Sinus.

Format

SIN (<Ausdruck>)

Erklärung

Ruft den Sinus des Wertes von <Ausdruck> ab.

Geben Sie <Ausdruck> in Grad an.

Wenn <Ausdruck> eine Real-Zahl mit doppelter Genauigkeit enthält, erhält der abgerufene Wert doppelte Genauigkeit. Andernfalls wird eine einfache Genauigkeit verwendet.
Wenn ein Argument in Radianten angegeben wird, geben Sie RAD nach der Konstanten an.

Erklärung:	**Die SIN-Funktion empfängt einen Winkel als Argument und gibt das Verhältnis der beiden Seiten, die den rechten Winkel eines rechtwinkligen Dreiecks einschließen, zurück. Die beiden hier angegebenen Seiten sind die dem Winkel gegenüberliegende Seite und die Hypotenuse. Das Verhältnis der beiden Seiten ist die Länge der gegenüberliegenden Seite (Gegenkathede) zur Länge der Hypotenuse. Der Wertebereich der Rückgabewerte reicht von -1 bis 1. Zum Umwandeln von Grad in Radianten multiplizieren Sie die Gradangabe mit π/180. Multiplizieren Sie zur Umwandlung von Radianten in Grad die Radiantenangabe mit 180/π.**

Verwandte Begriffe

COS, TAN, ASIN, ACOS, ATN, ATN2

Beispiel

```
DEFSNG lf1, lf2, lf3, lf4, lf5
lf1 = SIN(0.78525)          'Weist den Wert des Sinus von 0.78525 der Variablen
                            'lf1 zu.
lf2 = SIN(lf4)              'Weist den Wert des Sinus von lf4 der Variablen lf2
                            'zu.
lf3 = SIN(45) * lf5         'Weist den Wert des Sinus von 45 Grad, mulitpliziert
                            'mit lf5, der Variablen lf3 zu.
```

TAN (Funktion) [Entspricht SLIM]

Funktion

Ermittelt einen Tangens.

Format

TAN (<Ausdruck>)

Erklärung

Diese Anweisung ruft den Tangens des Wertes in <Ausdruck> ab.

Geben Sie <Ausdruck> in Grad an.

Wenn <Ausdruck> eine Real-Zahl mit doppelter Genauigkeit enthält, erhält der abgerufene Wert doppelte Genauigkeit. Andernfalls wird eine einfache Genauigkeit verwendet.
Wenn ein Argument in Radianten angegeben wird, geben Sie RAD nach der Konstanten an.

Erklärung: **Die TAN-Funktion empfängt einen Winkel als Argument und gibt das Verhältnis der beiden Seiten, die den rechten Winkel eines rechtwinkligen Dreiecks einschließen, zurück. Die beiden hier angegebenen Seiten schließen den rechten Winkel ein. Das Verhältnis der beiden Seiten ist der Wert der dem Winkel gegenüberliegenden Seite (Gegenkathede) zur an dem Winkel anliegenden Seite (Ankathede). Multiplizieren Sie zur Umwandlung von Grad in Radianten die Gradangabe mit π/180. Zum Umwandeln von Radianten in Grad multiplizieren Sie die Radiantenangabe mit 180/π.**

Verwandte Begriffe

SIN, COS, ASIN, ACOS, ATN, ATN2

Beispiel

```
DEFSNG lf1, lf2, lf3, lf4, lf5
lf1 = TAN(0,78525)      'Weist den Wert des Kosinus von 0,78525 der Variablen lf1
                        'zu.
lf2 = TAN(lf4)          'Weist den Wert des Tangens von lf4 der Variablen lf2 zu.
lf3 = TAN(45) * lf5     'Weist den Wert des Tangens von 45 Grad, mulitpliziert
                        'mit lf5, der Variablen lf3 zu.
```

15.3 Winkelumwandlung

DEGRAD (Funktion) [Entspricht SLIM]

Funktion

Wandelt die Einheit in Radianten um.

Format

DEGRAD (<Ausdruck>)

Erklärung

Diese Anweisung wandelt die Einheit des Wertes in <Ausdruck> von Grad in Radianten um.

Verwandte Begriffe

RADDEG

Beispiel

```
DEFSNG lf1, lf2, lf3
lf1 = DEGRAD(90)             'Weist den Wert von 90, umgewandelt in Radianten, der
                             'Variablen lf1 zu.
lf2 = SIN(DEGRAD(lf3 + 20))  'Weist den Wert des Vorzeichens von (lf3 + 20),
                             'umgewandelt in Radianten, der Variablen lf2 zu.
```

RAD (Funktion)

Funktion

Wandelt einen in Radianten angegebenen Wert in Grad um.

Format

<Numerischer Wert> RAD

Erklärung

Diese Anweisung wandelt einen in Radianten angegebenen <numerischen Wert> in Grad um und berechnet ihn.

Verwandte Begriffe

RADDEG

Beispiel

```
DIM lf1 As SINGLE
lf1 = 2RAD   'Wandelt Radiant 2 in Grad um und weist diesen Wert der Variablen
             'lf1 zu.
```

RADDEG (Funktion) [Entspricht SLIM]

Funktion

Wandelt die Einheit in Grad um.

Format

RADDEG (<Ausdruck>)

Erklärung

Diese Anweisung wandelt die Einheit des Wertes in <Ausdruck> von Radianten in Grad um.

Verwandte Begriffe

RAD, DEGRAD

Beispiel

```
DEFSNG lf1, lf2
lf1 = RADDEG(1,5705)    'Weist den Wert von 1,5705, umgewandelt in Grad, der
                        'Variablen lf1 zu.
lf2 = RADDEG(PI/2)      'Weist den Wert des Sinus von (/2), umgewandelt in Grad, der
                        'Variablen lf2 zu.
```

15.4 Geschwindigkeitsumwandlung

MPS (Funktion)

Funktion

Wandelt einen Ausdruck für die Geschwindigkeit um.

Format

MPS (<Ausdruck>)

Erklärung

Diese Anweisung wandelt den Geschwindigkeitswert in <Ausdruck> (Prozentwert) in den Wert für die Armendbewegungs- Geschwindigkeit (%) um.

<Ausdruck> wird in mm/sec angegeben.

Verwandte Begriffe

SPEED

Beispiel

```
SPEED MPS(50/2)         'Wandelt die interne Bewegungsgeschwindigkeit in
                        '50/2 (mm/sec) um.
                        'Im automatischen Betrieb ist SP100 = 1500 m/sec.
```

15.5 Zeitfunktion

SEC (Funktion)

Funktion

Wandelt einen in Sekunden ausgedrückten Wert in Millisekunden um.

Format

<Numerischer Wert> SEC

Erklärung

Diese Anweisung wandelt einen in Sekunden angegebenen <numerischen Wert> in Millisekunden um und berechnet ihn. Verwenden Sie diese Anweisung für Anweisungen mit Angaben in Millisekunden.

Beispiel

```
DELAY(10SEC) 'Wartet, bis 10 Sekunden abgelaufen sind.
```

15.6 Vektor

AVEC (Funktion)

Funktion

Extrahiert einen Anfahrvektor.

Format

AVEC (<Homogene Transformation>)

Erklärung

Diese Anweisung extrahiert einen Anfahrvektor aus Koordinaten des Typs Homogene Transformation.

Verwandte Begriffe

OVEC, PVEC

Beispiel

```
DEFTRN lt1, lt2, lt3
DEFVEC lv1, lv2
lv1 = AVEC(lt1)         'Weist den Anfahrvektor von lt1 der Variablen lv1 zu.
lv2 = AVEC(lt2 * lt3)   'Weist einen Anfahrvektor mit dem Wert (lt2 * lt3) der
                        'Variablen lv2 zu.
```

OVEC (Funktion)

Funktion

Extrahiert einen Orientierungsvektor.

Format

OVEC (<Homogene Transformation>)

Erklärung

Diese Anweisung extrahiert einen Orientierungsvektor aus Koordinaten des Typs Homogene Transformation.

Verwandte Begriffe

AVEC, PVEC

Beispiel

```
DEFTRN lt1, lt2, lt3
DEFVEC lv1, lv2
lv1 = OVEC(lt1)         'Weist den Orientierungsvektor von lt1 der Variablen lv1 zu.
lv2 = OVEC(lt2 * lt3)   'Weist einen Orientierungsvektor mit dem Wert (lt2 * lt3) der
                        'Variablen lv2 zu.
```

PVEC (Funktion)

Funktion

Extrahiert einen Positionsvektor.

Format

PVEC ({<Homogene Transformation>|<Positionstyp>})

Erklärung

Diese Anweisung extrahiert einen Positionsvektor aus Koordinaten des Typs Homogene Transformation oder aus Koordinaten des Typs Position.

Verwandte Begriffe

AVEC, OVEC, RVEC

Beispiel

```
                DIM lt1 As Trans
                DIM lv1 As Vector
                DIM lp1 As Position
                lv1 = PVEC(lt1)     'Weist den Positionsvektor von lt1 der Variablen
                                    'lv1 zu.

6 Achsen        lv1 = PVEC(lp1 + (100, 200, 0, 0, 0, 0) )
                                    'Weist den Positionsvektor lp1 +
                                    '(100, 200, 0, 0,0, 0) der Variablen lv1 zu.

4 Achsen        lv1 = PVEC(lp1 + (100, 200, 0, 0) )
                                    'Weist den Positionsvektor lp1+(100, 200, 0, 0)
                                    'der Variablen lv1 zu.
```

MAGNITUDE (Funktion)

Funktion

Ruft die Vektorgröße ab.

Format

MAGNITUDE (<Vektor>)

Erklärung

Diese Anweisung ruft die Vektorgröße eines Koordinatenwertes des Typs Vektor ab.

Verwandte Begriffe

DIST

Beispiel

```
DEFSNG lf1, lf2
DIM lv1 As Vector
lf1 = MAGNITUDE((10, 10, 10))     'Weist die Größe des Vektors (10, 10, 10) der
                                  'Variablen lf1 zu.
lf2 = MAGNITUDE(lv1)              'Weist die Größe des Vektors lv1 der Variablen
                                  'lf2 zu.
```

15.7 Transformation von Daten des Typs Stellung

J2P (Funktion)

Funktion

Wandelt Daten vom Typ Gelenk in Daten des Typs Position um.

Format

J2P (<Gelenk>)

Erklärung

Diese Anweisung wandelt die in <Gelenk> angegebenen Daten des Typs Gelenk in Daten des Typs Position um. Der abzurufende Wert entspricht den momentan eingestellten Werkzeug- und Arbeitskoordinatensystemen.

Verwandte Begriffe

J2T, P2J, P2T, T2J, T2P

Beispiel

```
                DEFPOS lp1, lp2
                DIM lj1 As Joint
                lp1 = J2P(lj1)      'Weist die lj1-Daten, umgewandelt in Daten des Typs
                                    'Position, der Variablen lp1 zu.

6 Achsen        lp2 = J2P((0, 0, 90, 0, 0, 0))
                                    'Weist die Daten (0, 0, 90, 0, 0, 0), umgewandelt
                                    'in Positionsdaten, der Variablen lp2 zu.

4 Achsen        lp2 = J2P((0, 0, 300, 0))
                                    'Weist die Daten (0, 0, 300, 0), umgewandelt
                                    'in Positionsdaten, der Variablen lp2 zu.
```

J2T (Funktion)

Funktion

Wandelt Daten vom Typ Gelenk in Daten des Typs Homogene Transformation um.

Format

J2T (<Gelenk>)

Erklärung

Diese Anweisung wandelt die in <Gelenk> angegebenen Daten des Typs Gelenk in Daten des Typs Homogene Transformation um. Der abzurufende Wert entspricht den momentan eingestellten Werkzeug- und Arbeitskoordinatensystemen.

Verwandte Begriffe

J2P, P2J, P2T, T2J, T2P

Beispiel

```
                DEFTRN lt1, lt2
                DIM lj1 As Joint
                lt1 = J2T(lj1)      'Weist die in Daten des Typs Homogene Transformation
                                    'umgewandelten Daten der Variablen lt1 zu.

6 Achsen        lt2 = J2T((0, 0, 90, 0, 0, 0))
                                    'Weist die in Daten (0, 0, 90, 0, 0, 0), umgewandelt
                                    'in   Daten   des   Typs   Homogene   Transformation,
                                    'der Variablen lt2 zu.

4 Achsen        lt2 = J2T((0, 0, 300, 0))
                                    'Weist die Daten (0, 0, 300, 0), umgewandelt in Daten
                                    'des Typs Homogene Transformation, der Variablen
                                    'lt2 zu.
```

P2J (Funktion)

Funktion

Wandelt Daten vom Typ Position in Daten des Typs Gelenk um.

Format

P2J (<Position>)

Erklärung

Diese Anweisung wandelt die als <Position> angegebenen Daten in Daten des Typs Gelenk um.
Wenn der FIG-Wert in den Positionsdaten -1 (nicht definiert) ist, wird die Transformation mit der aktuellen Armstellung ausgeführt. Der abzurufende Wert entspricht den momentan eingestellten Werkzeug- und Arbeitskoordinatensystemen.

Verwandte Begriffe

J2P, J2T, P2T, T2J, T2P, CURFIG

Beispiel

```
                DEFJNT lj1, lj2
                DIM lp1 As Point
                lj1 = P2J(lp1)      'Weist lp1-Daten, umgewandelt in Daten des Typs
                                    'Gelenk, mit der aktuellen Armstellung der
                                    'Variablen lj1 zu.

6 Achsen        lj2 = P2J((0, 0, 90, 0, 0, 180))
                                    'Weist die Daten (0, 0, 90, 0, 0, 180),
                                    'umgewandelt in Daten des Typs Gelenk, mit der
                                    'aktuellen Armstellung der Variablen lj2 zu.

4 Achsen        lj2 = P2J((100, 100, 300, 45))
                                    'Weist die Daten (100, 100, 300, 45), umgewandelt
                                    'in Daten des Typs Gelenk, mit der aktuellen
                                    'Armstellung der Variablen lj2 zu.
```

P2T (Funktion)

Funktion

Wandelt Daten des Typ Position in Daten des Typs Homogene Transformation um.

Format

P2T (<Position>)

Erklärung

Diese Anweisung wandelt die als <Position> angegebenen Daten in Daten des Typs Homogene Transformation um. Der abzurufende Wert entspricht den momentan eingestellten Werkzeug- und Arbeitskoordinatensystemen.

Verwandte Begriffe

J2P, J2T, P2J, T2J, T2P

Beispiel

```
                DEFTRN lt1, lt2
                DIM lp1 As Point
                Lt1 = P2T(lp1)      'Weist lp1-Daten, umgewandelt in Daten des Typs
                                    'Homogene Transformation, der Variablen lt1 zu.

6 Achsen        lt2 = P2T((350, 0, 450, 0, 0, 180))
                                    'Weist die Daten (350, 0, 450, 0, 0, 180), umgewandelt
                                    'in Daten des Typs Homogene Transformation, der
                                    'Variablen lt2 zu.

4 Achsen        lt2 = P2T((100, 100, 300, 45))
                                    'Weist die Daten (100, 100, 300, 45), umgewandelt in
                                    'Daten des Typs Homogene Transformation, der
                                    'Variablen lt2 zu.
```

T2J (Funktion)

Funktion

Wandelt Daten des Typs Homogene Transformation in Daten des Typs Gelenk um.

Format

T2J (<Homogene Transformation>)

Erklärung

Diese Anweisung wandelt die als <Homogene Transformation> angegebenen Daten in Daten des Typs Gelenk um.
Wenn “-1” (nicht definiert) für die <Armstellung> der homogenen Transformationsdaten angegeben ist, werden die Daten mit der aktuellen Roboter-Armstellung umgewandelt.
Der abzurufende Wert entspricht den momentan ausgewählten Werkzeug- und Arbeitskoordinatensystemen.

Verwandte Begriffe

J2P, J2T, P2J, P2T, T2P, CURFIG

Beispiel

```
DEFJNT lj1, lj2, lj3
DEFTRN lt1, lt2
DIM li1 As Integer
lj1 = T2J(lt1)    'Weist lt1-Daten, umgewandelt in Daten des Typs Gelenk,
                  'mit der aktuellen Armstellung der Variablen lj1 zu.
lj2 = T2J(lt2)    'Weist lt2-Daten, umgewandelt in Daten des Typs Gelenk,
                  'mit der Armstellung 0 der Variablen lj2 zu.
lj3 = T2J(350, 0, 450, 0, 1, 0, 0, 0, -1)
                  'Weist die Daten des Typs Homogene Transformation
                  '(350, 0, 450, 0, 1, 0, 0, 0, -1), umgewandelt in Daten des
                  'Typs Gelenk, mit der Armstellung li1 der Variablen lj3 zu.
```

T2P (Funktion)

Funktion

Wandelt Daten des Typs Homogene Transformation in Daten des Typs Position um.

Format

T2P (<Homogene Transformation>)

Erklärung

Diese Anweisung wandelt die als <Homogene Transformation> angegebenen Daten in Daten des Typs Position um. Der abzurufende Wert entspricht den momentan eingestellten Werkzeug- und Arbeitskoordinatensystemen.

Verwandte Begriffe

J2P, J2T, P2J, P2T, T2J

Beispiel

```
DEFPOS lp1, lp2
DIM lt1 As Trans
lp1 = T2P(lt1)    'Weist die lt1-Daten, umgewandelt in Daten des Typs Position,
                  'der Variablen lp1 zu.
lp2 = T2P(350, 0, 450, 0, 1, 0, 0, 0, -1)
                  'Weist die Daten des Typs Homogene Transformation (350, 0,
                  '450, 0, 1, 0, 0, 0, -1), umgewandelt in Daten des Typs
                  'Position, der Variablen lp2 zu.
```

TINV (Funktion)

Funktion

Berechnet eine inverse Matrix von Daten des Typs Homogene Transformation.

Format

TINV (<Homogene Transformation>)

Erklärung

Diese Anweisung invertiert die als <Homogene Transformation> angegebenen Daten.

Beispiel

```
DEFTRN lt1, lt2, Lt3
lt1 = TINV(350, 0, 450, 0, 1, 0, 0, 0, -1)
                  'Weist die inverse Matrix (350, 0, 450, 0, 1, 0, 0, 0,
                  '-1) der Variablen lt1 zu.
lt2 = TINV(lt3)   'Weist die inverse Matrix lt3 der Variablen lt2 zu.
```

15.8 Distanzextraktion

DIST (Funktion) [Entspricht SLIM]

Funktion

Gibt die Distanz zwischen zwei Punkten zurück.

Format

DIST (<Positionstyp1>,<Positionstyp2>)

Erklärung

Diese Anweisung ruft die Distanz zwischen <Position1> und <Position2> ab.

Beispiel

```
                DEFSNG lf1, lf2
                DEFPOS lp1, lp2, lp3

6 Achsen        lf1 = DIST((350, 0, 450, 0, 0, 180), lp1)
                                  'Weist die Distanz zwischen (350, 0, 450, 0, 0,
                                  '180) und lp1 der Variablen lf1 zu.

4 Achsen        lf1 = DIST((350, 0, 300, 45), lp1)
                                  'Weist die Distanz zwischen (350, 0, 300, 45)
                                  'und lp1 der Variablen lf1 zu.
                lf2 = DIST(lp2, lp3)
                                  'Weist die Distanz zwischen lp2 und lp3 der
                                  'Variablen lf2 zu.
```

15.9 Armstellungskomponente

FIG (Funktion)

Funktion

Extrahiert eine Armstellung.

Format

FIG ({<Position>|<Homogene Transformation>})

Erklärung

Diese Anweisung extrahiert eine Armstellung aus Positionsdaten und Daten des Typs Homogene Transformation.

Verwandte Begriffe

CURFIG, LETF, Roboter-Armstellung (Anhang 2)

Beispiel

```
                DIM li1 As Integer
                DIM lp1 As Point
                li1 = FIG(lp1)      'Weist die Armstellung von lp1 der Variablen li1 zu.

6 Achsen        li1 = FIG(lp1 + (0, 0, 100, 0, 0, 0) )
                                    'Weist die Armstellung von lp1 + (0,0,100,0,0,0)
                                    'der Variablen li1 zu.

4 Achsen        li1 = FIG(lp1 + (0, 0, 100,0) )
                                    'Weist die Armstellung von lp1 + (0, 0, 100,0) der
                                    'Variablen li1.
```

15.10 Winkelkomponente

JOINT (Funktion)

Funktion

Extrahiert einen Winkel aus Koordinaten des Typs Gelenk.

Format

JOINT (<Achsennummer>, <Gelenk>)

Erklärung

Extrahiert den Gelenkwinkel der in <Achsennummer> angegebenen Achse aus den in <Gelenk> angegebenen Gelenkkoordinaten.

Verwandte Begriffe

LETJ

Beispiel

```
                DEFJNT lj1, lj2
                DEFSNG lf1, lf2
                DIM li1 As Integer
                lf1 = JOINT(1, lj1)   'Weist den 1. Achsengelenkwinkel von lj1 der
                                      'Variablen lf1 zu.
                lf2 = JOINT(li1, lj2)'Weist den Achsengelenkwinkel mit den Werten von li1
                                      'und lj2 der Variablen lf2 zu.

6 Achsen        lf1 = JOINT(1, lj1 + (10, 10, 0, 0, 0, 0) )
                                      'Weist den 1. Achsengelenkwinkel von lj1 +
                                      '(10, 10, 0, 0, 0, 0) der Variablen lf1 zu.

4 Achsen        lf1 = JOINT(1, lj1 + (10, 10, 0, 0) )
                                      'Weist den 1. Achsengelenkwinkel von lj1 +
                                      '(10, 10, 0, 0) der Variablen lf1 zu.
```

15.11 Achsenkomponente

POSX (Funktion) [Entspricht SLIM]

Funktion

Extrahiert die X-Komponente.

Format

POSX ({<Position>|<Vektor>})

Erklärung

Diese Anweisung extrahiert die X-Komponente aus den Koordinaten des Typs Position oder Vektor.

Verwandte Begriffe

POSY, POSZ

Beispiel

```
                DIM lf1 As Single
                DIM lp1 As Position
                lf1 = POSX(lp1)     'Weist die X-Komponente von lp1 der Variablen
                                    'lf1 zu.

6 Achsen        lf1 = POSX(lp1 + (100, 100, 100, 0, 0, 0) )
                                    'Weist die X-Komponente von lp1 +
                                    '(100, 100, 100, 0, 0, 0) der Variablen lf1 zu.

4 Achsen        lf1 = POSX(lp1 + (100, 100, 100, 0) )
                                    'Weist die X-Komponente von lp1 +
                                    '(100, 100, 100, 0) der Variablen lf1 zu.
```

POSY (Funktion) [Entspricht SLIM]

Funktion

Extrahiert die Y-Komponente.

Format

POSY ({<Position>|<Vektor>})

Erklärung

Diese Anweisung extrahiert die Y-Komponente aus den Koordinaten des Typs Position oder Vektor.

Verwandte Begriffe

POSX, POSZ

Beispiel

```
                DIM lf1 As Single
                DIM lp1 As Position
                lf1 = POSY(lp1)     'Weist die Y-Komponente von lp1 der Variablen
                                    'lf1 zu.

6 Achsen        lf1 = POSY(lp1 + (100, 100, 100, 0, 0, 0) )
                                    'Weist die Y-Komponente von lp1 der Variablen
                                    'lp1 + (100, 100, 100, 0, 0, 0) zu.

4 Achsen        lf1 = POSY(lp1 + (100, 100, 100, 0) )
                                    'Weist die Y-Komponente von lp1 der Variablen
                                    'lp1 + (100, 100, 100, 0) zu.
```

POSZ (Funktion) [Entspricht SLIM]

Funktion

Extrahiert die Z-Komponente.

Format

POSZ ({<Position>|<Vektor>})

Erklärung

Diese Anweisung extrahiert die Z-Komponente aus den Koordinaten des Typs Position oder Vektor.

Verwandte Begriffe

POSX, POSY

Beispiel

```
                DIM lf1 As Single
                DIM lp1 As Position
                lf1 = POSZ(lp1)     'Weist die Z-Komponente von lp1 der Variablen
                                    'lf1 zu.

6 Achsen        lf1 = POSZ(lp1 + (100, 100, 100, 0, 0, 0) )
                                    'Weist die Z-Komponente von lp1 der Variablen
                                    'lp1 + (100, 100, 100, 0, 0, 0) zu.

4 Achsen        lf1 = POSZ(lp1 + (100, 100, 100, 0) )
                                    'Weist die Z-Komponente von lp1 der Variablen
                                    'lp1 + (100, 100, 100, 0) zu.
```

15.12 Rotationskomponente

POSRX (Funktion)

Funktion

Extrahiert die X-Achsen-Rotationskomponente.

Format

POSRX (<Position>)

Erklärung

Diese Anweisung extrahiert die X-Achsen-Rotationskomponente aus den als <Position> angegebenen Positionskoordinaten.

Verwandte Begriffe

POSRY, POSRZ

Beispiel

```
DIM lf1 As Single
DIM lp1 As Position
lf1 = POSRX(lp1)     'Weist die X-Achsen-Rotationskomponente von lp1 der
                     'Variablen lf1 zu.
```

POSRY (Funktion)

Funktion

Extrahiert die Y-Achsen-Rotationskomponente.

Format

POSRY (<Position>)

Erklärung

Diese Anweisung extrahiert die Y-Achsen-Rotationskomponente aus den als <Position> angegebenen Positionskoordinaten.

Verwandte Begriffe

POSRX, POSRZ

Beispiel

```
DIM lf1 As Single
DIM lp1 As Position
lf1 = POSRY(lp1)     'Weist die Y-Achsen-Rotationskomponente von lp1 der
                     'Variablen lf1 zu.
```

POSRZ (Funktion)

Funktion

Extrahiert die Z-Achsen-Rotationskomponente.

Format

POSRZ (<Position>)

Erklärung

Diese Anweisung extrahiert die Z-Achsen-Rotationskomponente aus den als <Position> angegebenen Positionskoordinaten.

Verwandte Begriffe

POSRX, POSRY

Beispiel

```
DIM lf1 As Single
DIM lp1 As Position
lf1 = POSRZ(lp1)     'Weist die Z-Achsen-Rotationskomponente von lp1 der
                     'Variablen lf1 zu.
```

POST (Funktion)

Funktion

Extrahiert die T-Achsen-Rotationskomponente.

Format

POST (<Position>)

Erklärung

Diese Anweisung extrahiert die T-Achsen-Rotationskomponente aus den als <Position> angegebenen Positionskoordinaten.

Beispiel

```
DIM lf1 As Single
DIM lp1 As Position
lf1 = POST(lp1)      'Weist die T-Achsen-Rotationskomponente von lp1 der
                     'Variablen lf1 zu.
```

15.13 Armstellungskomponente

RVEC (Funktion)

Funktion

Extrahiert eine Armstellung.

Format

RVEC (<Position>)

Erklärung

Diese Anweisung extrahiert eine Armstellung aus den Koordinaten des Typs Position.

Verwandte Begriffe

PVEC

Beispiel

```
DIM lp1 As Point
DIM lv1 As Vector
lv1 = RVEC(lp1)    'Weist die Armstellung von lp1 der Variablen lv1 zu.
```

15.14 Positionsfunktion

AREAPOS (Funktion)

Funktion

Gibt die Mittelpunktposition und die Richtung eines Quaders mit dem Positionstyp für einen Bereich zurück, für den eine Interferenzprüfung durchgeführt wird.

Format

AREAPOS (<Bereichsnummer>)

Erklärung

Dies Anweisung gibt die <Bereichsnummer> an, die mit dem AREA-Befehl deklariert wurde, und gibt die aktuelle Einstellung (Mittelpunktposition) mit dem Typ Position zurück.
<Bereichsnummer> Bereichsnummer (0 bis 7)

Verwandte Begriffe

AREA, SETAREA, RESETAREA, AREASIZE

Beispiel

```
DIM lp1 As Point
lp1 = AREAPOS(1)     'Weist die Mittelpunktsposition und die Richtung des Würfels
                     'für einen Bereich zu, in dem eine Interferenzprüfung mit
                     'lp1 ausgeführt wird.
```

AREASIZE (Funktion)

Funktion

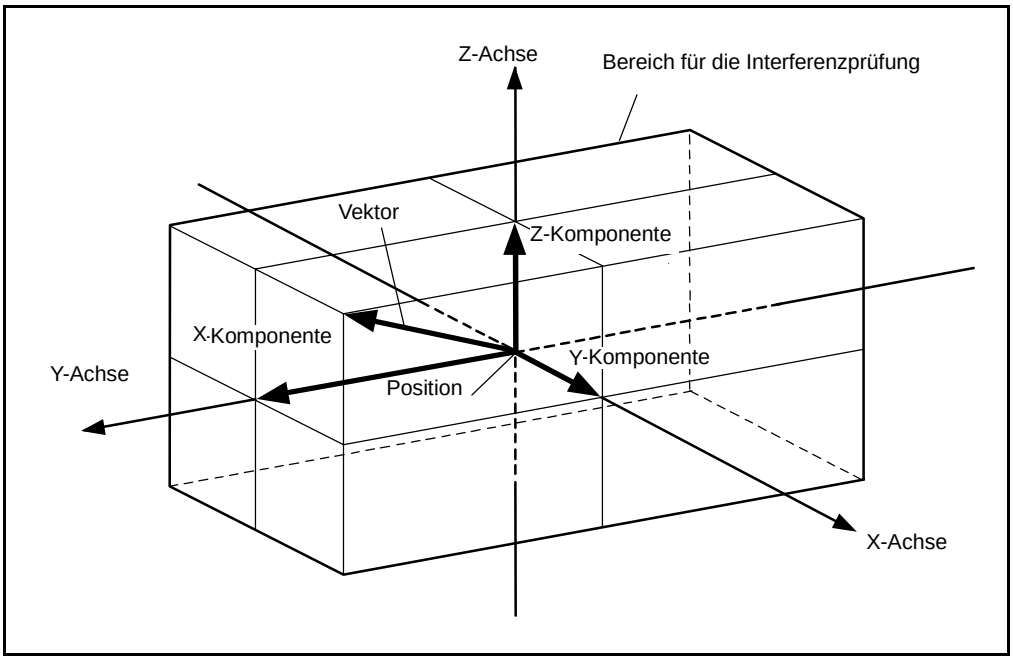
Gibt die Größe (jede Seitenlänge) eines Quaders zurück, der den Bereich für die Interferenzprüfung mit dem Vektortyp definiert.

Format

AREASIZE (<Bereichsnummer>)

Erklärung

Geben Sie die mit dem AREA-Befehl deklarierte <Bereichsnummer> an, und die Anweisung gibt die Größe (jede Seitenlänge) des Quaders zurück, der den Bereich mit dem Vektortyp definiert.
Die Länge der Seiten des Quaders ist die doppelte Länge der Vektoren X, Y und Z.



Verwandte Begriffe

AREA, SETAREA, RESETAREA, AREAPOS

Beispiel

```
DIM lv1 As Vector
lv1 = AREASIZE(1)       'Weist die Größe des Quaders zu, der den Bereich der
                        'Interferenzprüfung mit lv1 definiert.
```

TOOLPOS (Funktion)

Funktion

Gibt ein Werkzeugkoordinatensystem als Positionstyp zurück.

Format

TOOLPOS (<Werkzeugkoordinatensystemnummer>)

Erklärung

Geben Sie die mit dem TOOL-Befehl deklarierte <Werkzeugkoordinatensystemnummer> an, und diese Anweisung gibt eine Einstellung für ein Werkzeugkoordinatensystem mit dem Typ Position zurück. In diesem Fall wird -1 (nicht definiert) für eine FIG-Komponente der Positionsdaten angegeben.
<Werkzeugkoordinatensystemnummer> Werkzeugkoordinatensystemnummer (1 bis 63)

Verwandte Begriffe

TOOL, CHANGETOOL

Beispiel

```
DIM lp1 As Point
lp1 = TOOLPOS(1)       'Weist das Werkzeugkoordinatensystem der Variablen
                       'lp1 zu.
```

WORKPOS (Funktion)

Funktion

Gibt das Benutzerkoordinatensystem als Positionstyp zurück.

Format

WORKPOS (<Benutzerkoordinatensystemnummer>)

Erklärung

Geben Sie die mit dem WORK-Befehl deklarierte <Benutzerkoordinatensystemnummer> an, und diese Anweisung gibt eine Einstellung für ein Arbeitskoordinatensystem mit dem Typ Position zurück. In diesem Fall wird -1 (nicht definiert) für eine FIG-Komponente der Positionsdaten angegeben.
<Benutzerkoordinatensystemnummer> Benutzerkoordinatensystemnummer (1 bis 7)

Verwandte Begriffe

WORK, CHANGEWORK

Beispiel

```
DIM lp1 As Point
lp1 = WORKPOS(1)        'Weist das Benutzerkoordinatensystem der Variablen
                        'lp1 zu.
```

15.15 Zeichenfolgefunktion

ASC (Funktion)

Funktion

Führt eine Umwandlung in einen Zeichencode durch.

Format

ASC (<Zeichenfolge>)

Erklärung

Diese Anweisung wandelt das erste Zeichen von <Zeichenfolge> in einen Zeichencode um.

Anmerkung:	**Diese Funktion gibt nur den Wert des ersten Byte einer Zeichenfolge zurück. Auch wenn das erste Zeichen Kanji ist, wird daher das erste Byte des Shift-JIS-Code zurückgegeben.**

Verwandte Begriffe

CHR$, Zeichencodetabelle (Anhang 1)

Beispiel

```
DIM li1 As Integer
li1 = ASC( “ABCDEFGH” )          'Weist den Zeichencode 65(&H41) von A der
                                 'Variablen li1 zu.
```

BIN$ (Funktion) [Entspricht SLIM]

Funktion

Wandelt den Wert eines Ausdrucks in eine binäre Zeichenfolge um.

Format

BIN$ (<Ausdruck>)

Erklärung

Diese Anweisung wandelt den in <Ausdruck> angegebenen Wert in eine binäre Form um und wandelt den Wert in eine Zeichenfolge um.

Verwandte Begriffe

CHR$, HEX$, STR$, VAL

Beispiel

```
DIM li1 As Integer
DEFSTR ls1, ls2
ls1 = BIN$(20)          'Wandelt 20 in eine binäre Form um und weist diese Form
                        'der Variablen ls1 zu.
ls2 = BIN$(li1)         'Wandelt li1 in eine binäre Form um und weist diese Form
                        'der Variablen ls2 zu.
```

CHR$ (Funktion) [Entspricht SLIM]

Funktion

Wandelt einen ASCII-Code in ein Zeichen um.

Format

CHR$ (<Ausdruck>)

Erklärung

Diese Anweisung ruft ein Zeichen mit dem Zeichencode des in <Ausdruck> angegebenen Wertes ab.

Verwandte Begriffe

BIN$, HEX$, STR$, VAL, Zeichencodetabelle (Anhang 1)

Beispiel

```
DEFSTR ls1, ls2
ls1 = CHR$(49)          'Weist ein Zeichen mit dem Zeichencode 49 der Variablen ls1 zu.
ls2 = CHR$(&H4E)        'Weist ein Zeichen mit dem Zeichencode &H4E der Variablen
                        'ls2 zu.
```

SPRINTF$ (Funktion)

Funktion

Wandelt einen Ausdruck in ein angegebenes Format um und gibt ihn als Zeichenfolge zurück.

Format

SPRINTF$ (<Format>, <Ausdruck>)

Erklärung

<Format> umfaßt eine Zeichenfolge für die Ausgabe in der vorliegenden Form und eine Konvertierungsangabe zum Umwandeln und Ausgeben des Inhaltes von <Ausdruck> (als Teil einer Zeichenfolge).

Das erste Zeichen der Zeichenfolge für die Konvertierungsangabe ist %, gefolgt von einem oder mehreren Zeichen.

- Die Markierung 0 oder mehr kennzeichnet eine Zeichenfolge für die Umwandlung (keine bestimmte Reihenfolge). Die minimale Feldbreite (Option) zum Schreiben einer Zeichenfolge. Wenn die Anzahl der umgewandelten Zeichenwerte kleiner ist als die minimale Feldbreite, wird das Feld auf der linken Seite (bei linksbündigen Feldern auf der rechten Seite) mit Leerzeichen (Standardwert) aufgefüllt, bis die Mindesbreite des Feldes erreicht ist. Der Wert für die Feldbreite ist mit einem Sternchen (*) (weiter hinten in diesem Abschnitt beschrieben) oder mit einem ganzzahligen Wert angegeben.
- Genauigkeit (Option). Die Genauigkeitsoption gibt die Mindestanzahl von Zeichen, die bei der Umwandlung angezeigt werden, mit den Symbolen d, i, o, x und X an. Sie kennzeichnet mit den Symbolen e, E und f die Anzahl der Stellen nach dem Dezimalzeichen, die bei der Umwandlung angezeigt werden, und mit den Symbolen g und G die maximale effektive Anzahl an Stellen, die bei der Umwandlung angezeigt werden. Außerdem gibt diese Option die maximale Anzahl von Zeichen an, die bei der Umwandlung mit dem Umwandlungssymbol s aus einer Zeichenfolge geschrieben werden. Ein Punkt (.) wird als Genauigkeitswert verwendet, gefolgt von einem Sternchen (*) (weiter hinten in diesem Abschnitt beschrieben) oder einem ganzzahligen Wert (Option). Wenn nur ein Punkt angegeben wird, gilt die Genauigkeit 0. Ist die Genauigkeit mit anderen Umwandlungssymbolen angegeben, ist die Bewegung nicht angegeben.
- Symbol für die Umwandlungsangabe

  Wie eben beschrieben, kann ein Sternchen (*) für die Feldbreite und die Genauigkeit angegeben werden. Geben Sie bei einem Argument mit dem Typ Ganze Zahl den Wert für die Breite und Präzision an. Argumente, die die Feldbreite oder Genauigkeit oder beides angeben, müssen angegeben werden (Feldbreite vor Genauigkeit), bevor ein Argument (sofern vorhanden) umgewandelt werden kann. Wenn ein negativer Wert für die Feldbreite angegeben ist, wird er zur gleichen Angabe wie die positive Feldbreite nach einem negativen Umwandlungssymbol. Wenn für die Genauigkeit ein negativer Wert angegeben ist, hat dies die gleiche Wirkung, wie wenn die Genauigkeit nicht angegeben wird.

  Die Markierungen haben folgende Bedeutung.

- : Das Ergebnis der Umwandlung ist in dem Feld linksbündig angeordnet. (Wenn diese Markierung nicht angegeben ist, ist das Feld rechtsbündig angeordnet.)

+ : Bei einer Umwandlung mit Vorzeichen hat die erste Zahl immer ein Vorzeichen + oder - (wenn diese Markierung nicht angegeben ist, wird nur das Vorzeichen für einen negativen Wert hinzugefügt.)

<Leerzeichen> :
Wenn <Leerzeichen> und eine Markierung + am Anfang des Ergebnisses angegeben werden, wenn das erste Zeichen kein Vorzeichen bei einer Umwandlung mit Vorzeichen ist oder wenn nach einer Umwandlung mit Vorzeichen keine Zeichen generiert werden, wird <Leerzeichen> ignoriert.

: Die Umwandlung erfolgt mit dem “Ersatzformat.” Bei der Umwandlung mit dem Umwandlungssymbol o wird die Genauigkeit geändert, so daß die führende Ziffer des Ergebnisses Null wird. Bei der Umwandlung mit einem Umwandlungssymbol x (oder X) wird “0x” (oder 0X) hinzugefügt, so daß die erste Ziffer des Ergebnisses nicht Null ist. Bei der Umwandlung mit dem Umwandlungssymbol e, E, f, g oder G wird (auch wenn hinter dem Dezimalzeichen keine Stellen stehen) das Dezimalzeichen dem Ergebnis immer hinzugefügt. (Bei einer solchen Umwandlung wird das Dezimalzeichen normalerweise nur hinzugefügt, wenn hinter dem Dezimalzeichen noch Stellen angegeben sind.) Bei einer Umwandlung mit den Umwandlungssymbolen g und G wird das Zeichen o am Ende nicht gelöscht. Der Betrieb in einer anderen Umwandlung ist nicht definiert.

0 : Bei der Umwandlung mit den Umwandlungssymbolen d, i, o, x, X, e, E, f, g und G wird die Mindestfeldbreite durch Integrieren von 0 am Anfang (nach dem Vorzeichen oder der Basis) sichergestellt. Mit <Leerzeichen> gibt es keine Integration. Wenn eine Markierung 0 und eine Markierung - gleichzeitig angegeben werden, wird die Markierung 0 ignoriert. Wenn die Umwandlung der Genauigkeit mit den Umwandlungszeichen d, i, o, x und X angegeben ist, wird die Markierung 0 ignoriert. Die Operation mit anderen Umwandlungszeichen ist nicht definiert.

Die Umwandlungssymbole haben folgende Bedeutung.

d, i : Umwandlung des Wertes eines ganzzahligen Argumentes in eine Zeichenfolge von Dezimalzahlen mit einem Vorzeichen im Format “[-] dddd”. Die Mindestanzahl der angezeigten Stellen wird durch die Angabe der Genauigkeit festgelegt. Wenn ein umgewandelter Wert die Mindestanzahl von Stellen nicht erfüllt, wird der Anfang mit Nullen aufgefüllt. Die Standardgenauigkeit beträgt 1. Wenn der Wert Null mit der Genauigkeit Null umgewandelt wird, werden keine Zeichen ausgegeben.

o, x, X : Wandelt ein ganzzahliges Argument in eine Zeichenfolge einer Oktalzahl (o) oder einer Hexadezimalzahl (x oder X) ohne Vorzeichen in einem “dddd” oder Hexadezimalzahl (x oder X) Format um. Die Kleinbuchstaben (abcdef) werden für das Umwandlungssymbol x verwendet und die Großbuchstaben (ABCDEF) für das Umwandlungssymbol X. Die Mindestanzahl der angezeigten Zeichen wird durch die angegebene Genauigkeit festgelegt. Wenn ein Wert nach der Umwandlung die Mindestanzahl von Stellen nicht erfüllt, wird der Anfang mit Nullen aufgefüllt. Die Standardgenauigkeit beträgt 1. Wenn der Wert Null mit der Genauigkeit Null umgewandelt wird, werden keine Zeichen ausgegeben.

f : Wandelt ein Argument des Typs Real-Zahl in eine Zeichenfolge von Dezimalzahlen um, die im Format “[-] ddd.ddd” angezeigt werden. Die Anzahl der Stellen nach dem Dezimalzeichen wird durch die Angabe der Genauigkeit festgelegt. Wenn die Genauigkeit nicht angegeben ist, beträgt die Anzahl der Stellen nach dem Dezimalzeichen 6. Wenn die Genauigkeit 0 beträgt und kein Kennzeichen # angegeben ist, werden die Zeichen nach dem Dezimalzeichen nicht angezeigt. Wenn Zeichen nach dem Dezimalzeichen angezeigt werden, wird auch vor dem Dezimalzeichen mindestens ein Zeichen dargestellt. Der Wert wird auf eine geeignete Anzahl Stellen gerundet.

e, E : Wandelt ein Argument des Typs Real in ein Zeichen im Format “[-] d.ddde+/-dd” um. Eine Ziffer (nicht 0, wenn das Argument nicht 0 ist) wird vor dem Dezimalzeichen angezeigt. Die Anzahl der Stellen nach dem Dezimalzeichen wird entsprechend der angegebenen Genauigkeit festgelegt. Wenn die Genauigkeit nicht angegeben ist, beträgt die Anzahl der Stellen nach dem Dezimalzeichen 6. Wenn die Genauigkeit 0 beträgt und kein Kennzeichen # angegeben ist, werden die Zeichen nach dem Dezimalzeichen nicht ausgegeben. Der Wert wird auf eine geeignete Anzahl Stellen gerundet. Bei Verwendung des Umwandlungssymbols E wird der Exponent nicht als e, sondern als E dargestellt. Der Exponent wird immer mit 2 Stellen oder mehr dargestellt. Wenn der Wert 0 ist, wird der exponentielle Teil ebenfalls 0.

g, G : Umwandeln eines Argumentes des Typs Real in einen Typ f oder e (bei Auswahl von G wird E verwendet). Die gültige Anzahl von Stellen hängt von der Genauigkeit ab. Bei der Genauigkeit 0 beträgt die gültige Anzahl 1. Das verwendete Format wird von dem umzuwandelnden Wert festgelegt. Die Umwandlung in Typ e (oder E) wird nur verwendet, wenn der exponentielle Teil des Umwandlungsergebnisses kleiner ist als -4 bzw. größer oder gleich der Genauigkeit. Wenn eine 0 als letzte Ziffer hinter dem Dezimalzeichen steht, wird sie gelöscht. Zahlen mit einem Dezimalzeichen werden nur angezeigt, wenn hinter dem Dezimalzeichen eine Ziffer folgt.

C : Wandelt ein Argument des Typs Ganze Zahl in ein ASCII-Zeichen um und gibt die Zeichen nach der Umwandlung aus.

S : Systemreservierung

% : Gibt das Zeichen % aus. Das Argument wird nicht umgewandelt. Die Angabe der vollständigen Umwandlung lautet %%.

Es gibt keine Regelung für den Betrieb, wenn die Umwandlungsangabe falsch ist.
Wenn das Ergebnis der Umwandlung länger ist als die Feldbreite, wird es eventuell abgeschnitten; das Feld wird jedoch nie mit dem Umwandlungsergebnis verlängert.

Verwandte Begriffe

HEX$, STR$

Beispiel

```
S1 = SPRINTF$("% d",123)              'Weist "123" der Variablen S1 zu.
S2 = SPRINTF$("%-6.3f", 1.23)         'Weist "1.230 " der Variablen S2 zu.
S3 = SPRINTF$("%e",123.456#)          'Weist "1.234560e+002" der Variablen S3 zu.
S4 = SPRINTF$("%g",12345678#)         'Weist "1.23457e+007" der Variablen S4 zu.
S5 = SPRINTF$("%#x",128)              'Weist "0x80" der Variablen S5 zu.
```

HEX$ (Funktion) [Entspricht SLIM]

Funktion

Ruft einen Wert, der von einer dezimalen in eine hexadezimale Zahl umgewandelt wurde, als Zeichenfolge ab.

Format

HEX$ (<Ausdruck>)

Erklärung

Diese Anweisung wandelt den angegebenen Wert in <Expression> aus einer Dezimalzahl in eine Hexadezimalzahl um und wandelt den Wert in eine Zeichenfolge um.

Erklärung:	**Wenn ein Argument keine Ganze Zahl ist, wird es auf eine ganze Zahl abgerundent, die den Wert vor der Umwandlung nicht überschreitet.**

Verwandte Begriffe

BIN$, CHR$, STR$, VAL

Beispiel

```
DIM li1 As Integer
DEFSTR ls1, ls2
ls1 = HEX$(20)          'Wandelt 20 in eine Hexadezimalzahl um und weist sie der
                        'Variablen ls1 zu.
ls2 = HEX$(li1)         'Wandelt li1 in eine Hexadezimalzahl um und weist sie der
                        'Variablen ls2 zu.
```

LEFT$ (Funktion) [Entspricht SLIM]

Funktion

Extrahiert den linken Teil einer Zeichenfolge.

Format

LEFT$ (<Zeichenfolge>, <Anzahl Stellen>)

Erklärung

Diese Anweisung extrahiert eine Zeichenfolge für <Anzahl Stellen> von der linken Seite von <Zeichenfolge>.

Wenn die Zeichenfolge einen Nullwert enthält, wird der Nullwert zurückgegeben.
Wird in <Anzahl Stellen> ein Wert angegeben, der länger als die Zeichenfolge ist, wird die gesamte Zeichenfolge zurückgegeben.

Anmerkung: <Anzahl Stellen> wird als Bytezähler behandelt.

Verwandte Begriffe

LEN, MID$, RIGHT$

Beispiel

```
DIM li1 As Integer
DEFSTR ls1, ls2
ls1 = LEFT$( “abcdefg”, 3 )     'Weist 3 Zeichen “abc” von der linken Seite der
                                'Zeichenfolge in “abcdefg” der Variablen ls1 zu.
ls2 = LEFT$( ls1, li1 )         'Weist eine Zeichenfolge für den Wert der Anzahl
                                'von Zeichen in li1 von der linken Seite von ls1 der
                                'Variablen ls2 zu.
```

LEN (Funktion) [Entspricht SLIM]

Funktion

Ruft die Länge einer Zeichenfolge in Byte ab.

Format

LEN (<Zeichenfolge>)

Erklärung

Diese Anweisung ruft die Summe des Bytezählers für die Länge der in <Zeichenfolge> angegebenen Zeichenfolge ab.

Verwandte Begriffe

MID$, LEFT$, RIGHT$

Beispiel

```
DEFINT li1, li2
DIM ls1 As String
li1 = LEN( “abcdefg” ) 'Weist 7 Zeichen der Zeichenfolge “abcdefg” der Variablen li1 zu.
li2 = LEN( ls1 )       'Weist den Bytezähler ls1 der Variablen li2 zu.
```

MID$ (Funktion) [Entspricht SLIM]

Funktion

Extrahiert eine Zeichenfolge für die angegebene Anzahl Zeichen aus einer Zeichenfolge.

Format

MID$ (<Zeichenfolge >, <Startposition>[, <Anzahl Stellen>])

Erklärung

Diese Anweisung extrahiert eine Zeichenfolge für <Anzahl Stellen> aus <Startposition> von <Zeichenfolge>.
Wenn die Zeichenfolge einen Nullwert enthält, wird der Nullwert zurückgegeben.
Wird in <Anzahl Stellen> ein Wert angegeben, der länger als die Zeichenfolge ist, wird die gesamte Zeichenfolge zurückgegeben.
Wenn <Anzahl Stellen> nicht angegeben wird oder die Zeichenfolge weniger Zeichen hat als die angegebene Anzahl von Stellen, werden alle Zeichen nach der Startposition zurückgegeben.
Wird in <Startposition> ein Wert angegeben, der länger als die Zeichenfolge ist, wird die gesamte Zeichenfolge zurückgegeben.

Anmerkung: <Anzahl Stellen> wird als Bytezähler behandelt.

Verwandte Begriffe

LEN, LEFT$, RIGHT$

Beispiel

```
DEFSTR ls1, ls2
ls1 = MID$( “abcdefg”, 2, 3 )   'Weist die Zeichenfolge “bcd” der Variablen ls1 zu.
ls2 = MID$( ls1, 2, 2 )         'Weist die zwei Stellen ab dem zweiten Zeichen von ls1 der
                                'Variablen ls2 zu.
```

ORD (Funktion) [Entspricht SLIM]

Funktion

Führt eine Umwandlung in einen Zeichencode durch.

Format

ORD (<Zeichenfolge>)

Erklärung

Diese Anweisung wandelt das erste Zeichen in <Zeichenfolge> in einen Zeichencode um.

Anmerkung:	**Diese Funktion gibt nur den Wert des ersten Byte einer Zeichenfolge zurück. Auch wenn das erste Zeichen Kanji ist, wird daher das erste Byte des Shift-JIS-Code zurückgegeben. Wenn das Zeichen “ ” ist, wird 0 zurückgegeben.**

Verwandte Begriffe

CHR$, Zeichencodetabelle (Anhang 1)

Beispiel

```
DIM li1 As Integer
li1 = ORD( “ABCDEFGH” )          'Weist den Zeichencode 65(&H41) von A der
                                 'Variablen li1 zu.
```

RIGHT$ (Funktion) [Entspricht SLIM]

Funktion

Extrahiert den rechten Teil einer Zeichenfolge.

Format

RIGHT$ (<Zeichenfolge>, <Anzahl Stellen>)

Erklärung

Diese Anweisung extrahiert eine Zeichenfolge für <Anzahl Stellen> von der rechten Seite von <Zeichenfolge>.
Wenn die Zeichenfolge einen Nullwert enthält, wird der Nullwert zurückgegeben.
Wird in <Anzahl Stellen> ein Wert angegeben, der länger als die Zeichenfolge ist, wird die gesamte Zeichenfolge zurückgegeben.

Anmerkung: <Anzahl Stellen> wird als Bytezähler behandelt.

Verwandte Begriffe

LEN, MID$, LEFT$

Beispiel

```
DIM li1 As Integer
DEFSTR ls1, ls2
ls1 = RIGHT$( “abcdefg”, 3 )    'Weist die Zeichenfolge “efg” für 3 Zeichen
                                'von der rechten Seite in “abcdefg” der Variablen
                                'ls1 zu.
ls2 = RIGHT$( ls1, li1 )        'Weist eine Zeichenfolge für den Wert der Anzahl
                                'von Zeichen in li1 von der rechten Seite von ls1 der
                                'Variablen ls2 zu.
```

STRPOS (Funktion) [Entspricht SLIM]

Funktion

Ruft die Position einer Zeichenfolge ab.

Format

STRPOS (<Zeichenfolge 1>, <Zeichenfolge 2>)

Erklärung

Diese Anweisung ruft die Position von <Zeichenfolge 2> in <Zeichenfolge 1> ab.
Die Position wird als 1, 2, … von links gezählt.
Wird die Zeichenfolge nicht gefunden, wird Null zurückgegeben.

Verwandte Begriffe

Beispiel

```
DIM li1 As Integer
li1 = STRPOS(“abcdefg”, “bc”)   'Weist die Position (2) von “bc” in “abcdefg” der
                                'Variablen li1 zu.
```

STR$ (Funktion) [Entspricht SLIM]

Funktion

Wandelt einen Wert in eine Zeichenfolge um.

Format

STR$ (<Ausdruck>)

Erklärung

Diese Anweisung wandelt den in <Ausdruck> angegebenen Wert in eine Zeichenfolge um.

Erklärung:	**Wenn ein Wert in eine Zeichenfolge umgewandelt wird, gibt es immer eine Leerstelle zur Anzeige des Vorzeichens am Anfang des Rückgabewertes. Wenn der Wert positiv ist, wird am Anfang des Rückgabewertes der Str-Funktion ein Leerzeichen eingefügt. Dieses Leerzeichen gibt das Pluszeichen an. Die Str-Funktion erkennt nur einen Punkt (.) als gültiges Dezimalzeichen.**

Verwandte Begriffe

BIN$, CHR$, HEX$, VAL

Beispiel

```
DIM li1 As Integer
DEFSTR ls1, ls2
ls1 = STR$(20)          'Wandelt 20 in eine Zeichenfolge um und weist sie der Variablen
                        'ls1 zu.
ls2 = STR$(li1)         'Wandelt li1 in eine Zeichenfolge um und weist sie der Variablen
                        'ls2 zu.
```

VAL (Funktion) [Entspricht SLIM]

Funktion

Wandelt eine Zeichenfolge in einen numerischen Wert um.

Format

VAL (<Zeichenfolge>)

Erklärung

Diese Anweisung wandelt den in <Zeichenfolge> angegebenen Wert in einen numerischen Wert um.

Wenn das erste Zeichen von <Zeichenfolge> nicht +, -, & oder ein numerischer Wert ist, wird VAL 0.

Erklärung:	**Wenn das System ein anderes Zeichen findet als die Zahlen in der Zeichenfolge, stoppt die Val-Funktion den Lesevorgang. Die Val-Funktion interpretiert keine Symbole oder Buchstaben, die normalerweise als Teil eines numerischen Wertes betrachtet werden, beispielsweise das Yen-Symbol (\) oder Kommas (,) als Tausendertrennzeichen. Die Val-Funktion erkennt jedoch die Präfixe &H (Hexadezimalzahl) und &B (Binärzahl). Leerzeichen, Tabulatorzeichen und Zeilenvorschubzeichen in der Zeichenfolge eines Argumentes werden ignoriert.**

Verwandte Begriffe

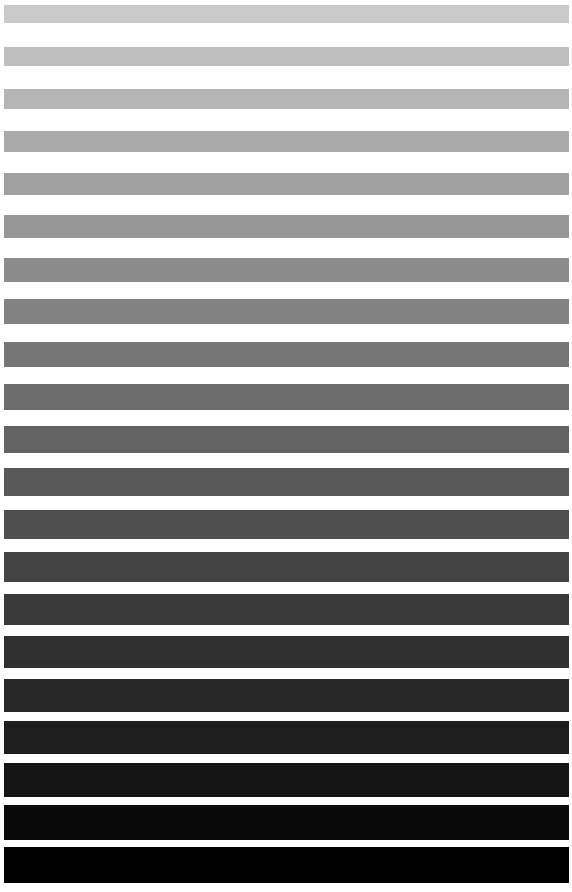
BIN$, CHR$, HEX$, STR$

Beispiel

```
DEFINT li1, li2, li3
li1 = VAL(“&B100”)      'Wandelt “&B100” in einen numerischen Wert um
                        '(4 in dezimal) und weist ihn der Variablen li1 zu.
li2 = VAL(“&H20”)       'Wandelt “&H20” in einen numerischen Wert um
                        '(32 in dezimal) und weist ihn der Variablen li2 zu.
li3 = VAL(“-30”)        'Wandelt “-30” in einen numerischen Wert um (-30 in dezimal) und
                        'weist ihn der Variablen li3 zu.
```

Kapitel 16

Konstanten



Dieses Kapitel enthält eine Erläuterung zu den in PAC vorbereiteten Konstanten.

16.1 Integrierte Konstanten

OFF (Integrierte Konstante)

Funktion

Stellt den Wert OFF (0) ein.

Format

OFF

Erklärung

Diese Anweisung stellt den Wert OFF (0) in einem Ausdruck ein.

Verwandte Begriffe

ON

Beispiel

```
1F I1 = TRUE THEN          'Setzt den Booleschen Wert auf wahr (1).
  I1 = ON                  'Stellt ON (1) für die Variable des Typs Ganze Zahl ein.
ELSEIF I1 = FALSE THEN     'Setzt den Booleschen Wert auf wahr (1).
  I1 = OFF                 'Stellt OFF (0) für die Variable des Typs Ganze Zahl ein.
ELSE
  D1 = PI                  'Weist der Variablen des Typs Real zu.
ENDIF
```

ON (Integrierte Konstante)

Funktion

Stellt den Wert ON (1) ein.

Format

ON

Erklärung

Diese Anweisung stellt den Wert ON (1) in einem Ausdruck ein.

Verwandte Begriffe

OFF

Beispiel

```
1F I1 = TRUE THEN          'Setzt den Booleschen Wert auf wahr (1).
  I1 = ON                  'Stellt ON (1) für die Variable des Typs Ganze Zahl ein.
ELSEIF I1 = FALSE THEN     'Setzt den Booleschen Wert auf wahr (1).
  I1 = OFF                 'Stellt OFF (0) für die Variable des Typs Ganze Zahl ein.
ELSE
  D1 = P                   'Weist der Variablen des Typs Real zu.
ENDIF
```

PI (Integrierte Konstante)

Funktion

Stellt einen Wert ein.

Format

PI

Erklärung

Diese Anweisung gibt einen Wert mit doppelter Genauigkeit zurück.

Beispiel

```
1F I1 = TRUE THEN          'Setzt den Booleschen Wert auf wahr (1).
  I1 = ON                  'Stellt ON (1) für die Variable des Typs Ganze Zahl ein.
ELSEIF I1 = FALSE THEN     'Setzt den Booleschen Wert auf wahr (1).
  I1 = OFF                 'Stellt OFF (0) für die Variable des Typs Ganze Zahl ein
ELSE
  D1 = PI                  'Weist der Variablen des Typs Real zu.
ENDIF
```

FALSE (Integrierte Konstante)

Funktion

Setzt einen Booleschen Wert auf falsch (0).

Format

FALSE

Erklärung

Diese Anweisung setzt einen Booleschen Wert in einer Anweisung auf falsch (0).

Verwandte Begriffe

TRUE

Beispiel

```
1F I1 = TRUE THEN          'Setzt den Booleschen Wert auf wahr (1).
  I1 = ON                  'Stellt ON (1) für die Variable des Typs Ganze Zahl ein.
ELSEIF I1 = FALSE THEN     'Setzt den Booleschen Wert auf wahr (1).
  I1 = OFF                 'Stellt OFF (0) für die Variable des Typs Ganze Zahl ein.
ELSE
  D1 = PI                  'Weist der Variablen des Typs Real zu.
ENDIF
```

TRUE (Integrierte Konstante)

Funktion

Setzt einen Booleschen Wert auf wahr (1).

Format

TRUE

Erklärung

Diese Anweisung setzt einen Booleschen Wert auf wahr (1).

Verwandte Begriffe

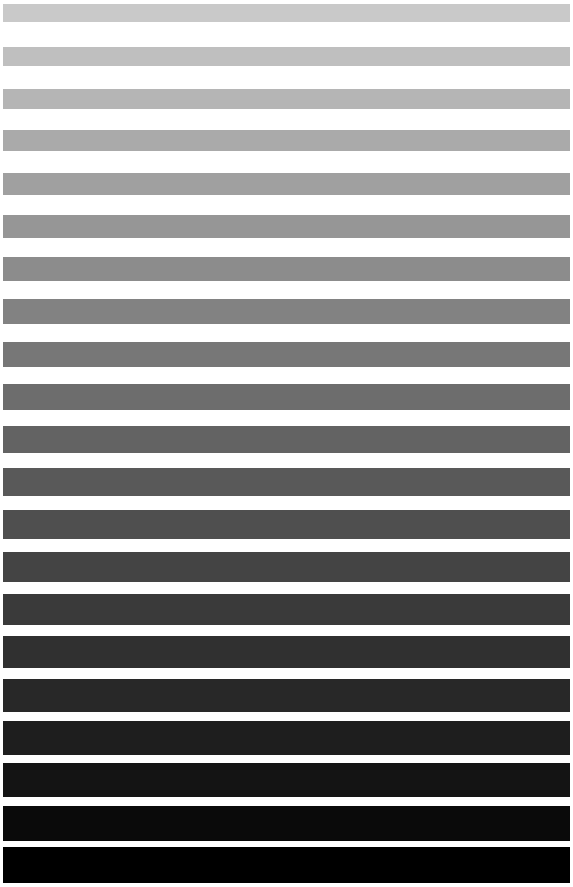
FALSE

Beispiel

```
1F I1 = TRUE THEN          'Setzt den Booleschen Wert auf wahr (1).
  I1 = ON                  'Stellt ON (1) für die Variable des Typs Ganze Zahl ein.
ELSEIF I1 = FALSE THEN     'Setzt den Booleschen Wert auf wahr (1).
  I1 = OFF                 'Stellt OFF (0) für die Variable des Typs Ganze Zahl ein.
ELSE
  D1 = PI                  'Weist der Variablen des Typs Real zu.
ENDIF
```

Kapitel 17

Steuerung von Zeit/Datum



Dieses Kapitel enthält eine Erläuterung der Befehle zur Steuerung von Datum, Uhrzeit, verstrichener Zeit und zeitabhängiger Unterbrechungen.

17.1 Zeit/Datum

DATE$ (Systemvariable) [Entspricht SLIM]

Funktion

Ruft das aktuelle Datum ab.

Format

DATE$

Erklärung

Diese Anweisung speichert das aktuelle Datum im folgenden Format: “jjjj/mm/tt”(Jahr/Monat/Tag).

Verwandte Begriffe

TIME$

Beispiel

```
DIM ls1 As String
ls1 = DATE$            'Weist das aktuelle Datum der Variablen ls1 zu.
```

TIME$ (Systemvariable) [Entspricht SLIM]

Funktion

Ruft die aktuelle Uhrzeit ab.

Format

TIME$

Erklärung

Diese Anweisung speichert die aktuelle Uhrzeit im folgenden Format: “hh:mm:ss” (Stunde: Minute: Sekunde).
Die Zeit wird im 24-Stunden-Format angezeigt.

Verwandte Begriffe

DATE$

Beispiel

```
DIM ls1 As String
ls1 = TIME$            'Weist die aktuelle Uhrzeit der Variablen ls1 zu.
```

TIMER (Systemvariable) [Entspricht SLIM]

Funktion

Ruft die abgelaufene Zeit ab.

Format

TIMER

Erklärung

Diese Anweisung ruft die abgelaufene Zeit in Millisekunden ab dem Einschalten des Controllers (0) ab.

Anmerkung:	**Wenn die abgelaufene Zeit 2147483647 Millisekunden übersteigt, wird die abgelaufene Zeit ab -214148,3648 Millisekunden angezeigt.**

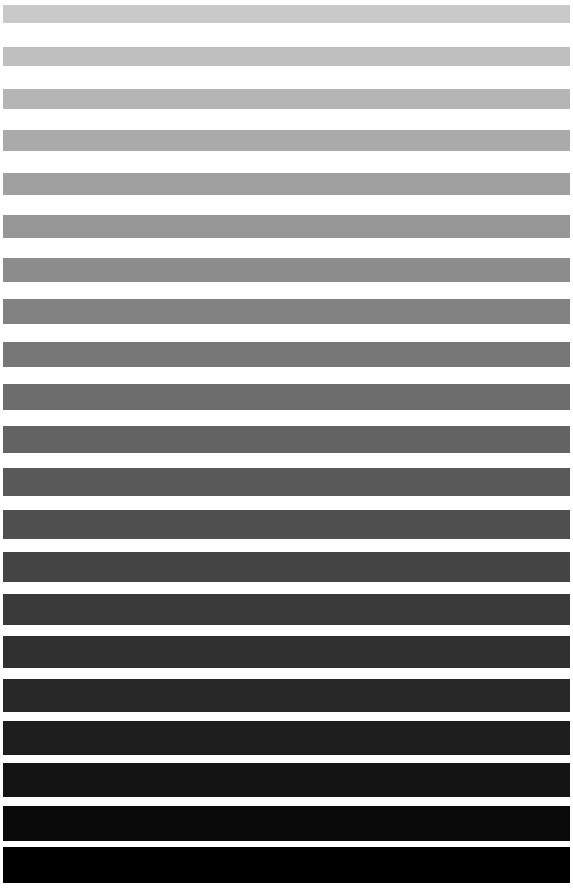
Verwandte Begriffe

Beispiel

```
DEFINT li1, li2, li3
li1 = TIMER             'Weist die abgelaufene Zeit ab der Referenzzeit der
                        'Variablen li1 zu.
li2 = TIMER + li3       'Weist den Wert der abgelaufenen Zeit ab der Referenzzeit
                        'plus der Zeit li3 der Variablen li2 zu.
```

Kapitel 18

Fehlersteuerungen



Wenn ein Fehler auftritt, können Sie die Ursache des Fehlers überprüfen oder eine Prozedur zur Wiederherstellung programmieren. Dieses Kapitel enthält eine Erläuterung der zur Verarbeitung von Fehlern erforderlichen Befehle.

18.1 Fehlerinformationen

ERL (Systemvariable)

Funktion

Ruft die Zeilennummer ab, in der ein Fehler auftrat.

Format

ERL

Erklärung

Dieses Programm speichert die Zeilennummer des Programmes, an der ein Fehler auftrat.
Der anfängliche Status ist relativ instabil; auch wenn Sie den Inhalt von ERL lesen können, bevor der Fehler auftritt, so hat dieser Inhalt keine Bedeutung.
ERL enthält keine Zeilennummern zu Fehlern in Programmen, die im Multitasking-Betrieb ausgeführt werden.

Anmerkungen:	**Dieser Befehl hat keine Bedeutung, wenn "Fehlerunterbrechung Prozeßcode löschen" auf ON (1) in [Programm] von [Projekteinstellung] des PAC Manager in WINCAPSII eingestellt ist.** **Die gleiche Einstellung wurde im Programmiergerät vorbereitet.**

Verwandte Begriffe

ERR, ON ERROR GOTO

Beispiel

```
DIM Li1 As INTEGER
DIM Li2% (2, 3)
ON ERROR GOTO *Error1    'Verzweigt zur Verarbeitungsroutine *Error1, wenn der
                         'Fehler auftritt.
FOR Lil = 0 TO 3
  Li2 (0, Li1) = 0       'Es tritt ein Fehler auf, wenn Li1 gleich 3 ist; in diesem Fall
                         'wird zu der Verarbeitungsroutine '*Error1 verzweigt.
NEXT Li1
*R_Label:
END

*Errorl:                 'Deklariert die Kennung *Errorl.
 S1 = "Line"+STR$(ERL)+ ":" +ERRMSG$(ERR)
                         'Bearbeitet die Fehlermeldung.
 PRINTMSG S1, 2, "Error"'Gibt die Fehlerinformationen aus.
 IF ERR = &H3B3E THEN
   RESUME NEXT           'Kehrt zu der Zeile nach der Zeile zurück, in der
                         'der Fehler auftrat.
 ELSE
   RESUME *R_Label       'Kehrt zurück zu der Zeile mit *R_Label.
ENDIF
```

ERR (Systemvariable)

Funktion

Ruft die Nummer eines aufgetretenen Fehlers ab.

Format

ERR

Erklärung

Dieses Programm speichert die Nummer eines aufgetretenen Fehlers.
ERR enthält keine Zeilennummern zu Fehlern in Programmen, die im Multitasking-Betrieb ausgeführt werden.

Anmerkungen:	**Dieser Befehl hat keine Bedeutung, wenn "Fehlerunterbrechung Prozeßcode löschen" auf ON (1) in [Programm] von [Projekteinstellung] des PAC Manager in WINCAPSII eingestellt ist.** **Die gleiche Einstellung wurde im Programmiergerät vorbereitet.**

Verwandte Begriffe

ERL, ERRMSG$, ON ERROR GOTO

Beispiel

```
DIM Li1 As INTEGER
DIM Li2% (2, 3)
ON ERROR GOTO *Error1    'Verzweigt zur Verarbeitungsroutine *Error1,
                         'wenn der Fehler auftrifft.
FOR Lil = 0 TO 3
  Li2 (0, Li1) = 0       'Wenn Li1 gleich 3 ist, tritt ein Fehler auf; in diesem Fall
                         'wird zu der Verarbeitungsroutine *Error1 verzweigt.
NEXT Li1
*R_Label:
END

*Errorl:                 'Deklariert die Kennung *Errorl.
 S1= "Line" +STR$(ERL)+ ":" +ERRMSG$(ERR)
                         'Bearbeitung der Fehlermeldung.
 PRINTMSG S1,2, "error"  'Gibt die Fehlerinformationen aus.
 IF ERR = &H3B3E THEN
   RESUME NEXT           'Kehrt zu der Zeile nach der Zeile zurück, in der
                         'der Fehler auftrat.
 ELSE
   RESUME *R_Label       'Kehrt zu der Zeile mit *R_Label zurück.
ENDIF
```

ERRMSG$ (Funktion)

Funktion

Legt eine Fehlermeldung fest.

Format

ERRMSG$ (<Arithmetischer Ausdruck>)

Erklärung

Diese Anweisung legt die entsprechende Fehlermeldung für die Fehlernummer, die mit <Numerischer Ausdruck> angegeben wurde, fest.

Verwandte Begriffe

ERL, ERR, ON ERROR GOTO

Beispiel

```
Bsp. 1:  DIM ls1 As String
         ls1 = ERRMSG$(&H600C)   'Weist eine Fehlermeldung mit einer Fehlernummer
                                 '&H600C der Variablen ls1 zu.
Bsp. 2:  DIM Li1 As INTEGER
         DIM Li2% (2, 3)
         ON ERROR GOTO *Error1   'Verzweigt zur Verarbeitungsroutine *Error1, wenn
                                 'der Fehler auftritt.
         FOR Lil = 0 TO 3
            Li2 (0, Li1) = 0     'Wenn Li1 gleich 3 ist, tritt ein Fehler auf; in
                                 'diesem Fall wird zu der Verarbeitungsroutine
                                 *Error1 verzweigt.
         NEXT Li1
         *R_Label:
         END

         *Errorl:                'Deklariert die Kennung *Errorl.
          S1= “Line” +STR$(ERL)+ “:” +ERRMSG$(ERR)
                                 'Bearbeitet die Fehlermeldung.
          PRINTMSG S1,2, “error” 'Gibt die Fehlerinformationen aus.
          IF ERR = &H3B3E THEN
            RESUME NEXT          'Kehrt zu der Zeile nach der Zeile zurück, in der
                                 'der Fehler auftrat.
          ELSE
            RESUME *R_Label      'Kehrt zurück zu der Zeile mit *R_Label.
         ENDIF
```

18.2 Fehlerunterbrechung

ON ERROR GOTO (Anweisung)

Funktion

Unterbricht die Verarbeitung, wenn ein Fehler auftritt.

Format

ON ERROR {GOTO|GO TO}<Kennungsname>

Erklärung

Diese Anweisung definiert eine Routine zur Unterbrechungsverarbeitung und führt sie aus, wenn ein Fehler auftritt. Wenn diese Unterbrechung auftritt, wird das Programm mit der in <Kennungsname> angegebenen Fehlerverarbeitungsroutine ausgeführt. Führen Sie zur Rückkehr aus der Fehlerprozeßroutine in das Programm die Anweisung RESUME aus.
Wenn bei der Ausführung der Fehlerverarbeitungsroutine ein Fehler auftritt, wird das laufende Programm gestoppt.

Anmerkungen:	**Dieser Befehl hat keine Bedeutung, wenn "Fehlerunterbrechung Prozeßcode löschen" auf ON (1) in [Programm] von [Projekteinstellung] des PAC Manager in WINCAPSII eingestellt ist.** **Die gleiche Einstellung wurde im Programmiergerät vorbereitet.**

Verwandte Begriffe

ERR, ERL, RESUME

Beispiel

```
DIM Li1 As INTEGER
DIM Li2% (2, 3)
ON ERROR GOTO *Error1     'Wenn ein Fehler auftritt, verzweigt das Programm zur
                          'Verarbeitungsroutine Error1.
FOR Lil = 0 TO 3
  Li2 (0, Li1) = 0        'Wenn Li1 gleich 3 ist, tritt ein Fehler auf, und das
                          'Programm verzweigt zu der Verarbeitungsroutine *Error1.
NEXT Li1
*R_Label:
END

*Errorl:                  'Deklariert die Kennung *Errorl
 S1= "Line" +STR$(ERL)+ ":" b+ERRMSG$(ERR)
                          'Bearbeitet die Fehlermeldung.
 PRINTMSG S1,2, "error"   'Gibt die Fehlerinformationen aus.
 IF ERR = &H3B3E THEN
   RESUME NEXT            'Kehrt zu der Zeile nach der Zeile zurück, in der
                          'der Fehler auftrat.
 ELSE
   RESUME *R_Label        'Kehrt zu der Zeile mit *R_Label zurück.
ENDIF
```

RESUME (Anweisung)

Funktion

Kehrt von einer Routine zur Unterbrechungsverarbeitung zurück.

Format

RESUME [NEXT|<Kennungsname>]

Erklärung

Diese Anweisung kehrt zu dem Programm zurück, bevor es von einer Routine zur Unterbrechungsverarbeitung, wie in der Anweisung ON ERROR GOTO definiert, verzweigt wurde.
Mit RESUME NEXT kehrt das Programm zu der Zeile nach der Zeile zurück, in der ein Fehler auftrat.
Mit RESUME <Kennungsname> übergibt das Programm die Steuerung an die angegebene Kennungszeile.
Wenn Sie keine der Argumentoptionen in der RESUME-Anweisung angeben, kehrt das Programm zu der Zeile zurück, in der der Fehler auftrat.

Anmerkungen (1): Dieser Befehl hat keine Bedeutung, wenn "Fehlerunterbrechung Prozeßcode löschen" auf ON (1) in [Programm] von [Projekteinstellung] des PAC Manager in WINCAPSII eingestellt ist.
Die gleiche Einstellung wurde im Programmiergerät vorbereitet.
(2): Wenn das Programm zu der Zeile zurückkehrt, in der der Fehler auftrat, ohne daß die Ursache des Fehlers beseitigt wird, führt das Programm eventuell eine Endlosschleife aus.
(3): Sie können diese Anweisung nicht außerhalb der Routine zur Unterbrechungsverarbeitung ausführen.

Verwandte Begriffe

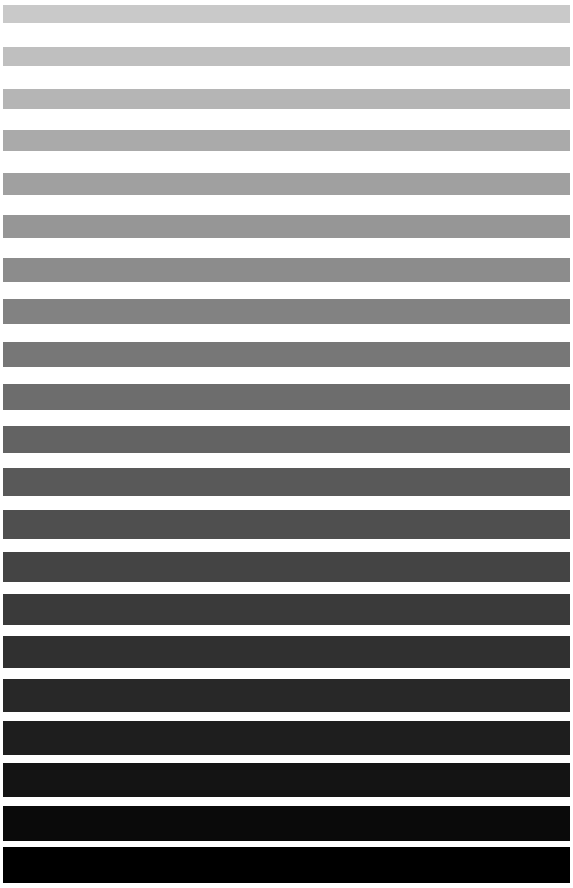
ON ERROR GOTO

Beispiel

```
DIM Li1 As INTEGER
DIM Li2% (2, 3)
ON ERROR GOTO *Error1     'Wenn ein Fehler auftritt, verzweigt das Programm zur
                          'Verarbeitungsroutine Error1.
FOR Lil = 0 TO 3
  Li2 (0, Li1) = 0        'Wenn Li1 gleich 3 ist, tritt ein Fehler auf, und das
                          'Programm verzweigt zu der Verarbeitungsroutine *Error1.
NEXT Li1
*R_Label:
END

*Errorl:                  'Deklariert die Kennung *Errorl
 S1= "Line" +STR$(ERL)+ ":" b+ERRMSG$(ERR)
                          'Bearbeitet die Fehlermeldung.
 PRINTMSG S1,2, "error"   'Gibt die Fehlerinformationen aus.
 IF ERR = &H3B3E THEN
   RESUME NEXT            'Kehrt zu der Zeile nach der Zeile zurück, in der der
                          'Fehlerauftrat.
 ELSE
   RESUME *R_Label        'Kehrt zu der Zeile mit *R_Label zurück.
ENDIF
```

Kapitel 19

Systeminformationen



Dieses Kapitel enthält eine Erläuterung der erforderlichen Befehle zum Abrufen von Informationen zum Robotersystem.

19.1 System

GETENV (Funktion)

Funktion

Ruft die Werte für die Umgebungseinstellung des Systems ab.

Format

GETENV (<Tabellennummer>, <Elementnummer>)

Erklärung

Diese Anweisung ruft die Werte der Umgebungseinstellung des Systems ab, die mit <Tabellennummer> und <Elementnummer> angegeben wurden. Die Werte für die Elemente der Umgebungseinstellung sind in Anhang 3 “Umgebungseinstellwerte”, beschrieben

Verwandte Begriffe

VER$, Umgebungseinstellwerte (Anhang 3)

Beispiel

```
DIM li1 As Integer
li1 = GETENV (9, 1)  'Weist den Inhalt der Umgebungseinstellwerte von
                     'Nummer 9 und 1 der Variablen li1 zu.
```

LETENV (Anweisung)

Funktion

Definiert die Werte für die Umgebungseinstellung des Systems.

Format

LETENV <Tabellennummer>, <Elementnummer>, <Einstellungswert>

Erklärung

Diese Anweisung stellt die mit <Tabellennummer> und <Elementnummer> angegebenen Werte der Umgebungseinstellung des Systems auf den <Einstellungswert> ein.
Die Werte für die Elemente der Umgebungseinstellung sind in Anhang 3 “Umgebungseinstellwerte”, beschrieben.
Zu Beginn der Ausführung kann ein Fehler auftreten, wenn Sie eine Elementnummer oder einen Einstellungswert angeben, der den Einstellungsbereich überschreitet.

Anmerkungen: **Die einfachste Möglichkeit festzustellen, ob eine Einstellung gültig ist, ist das Öffnen des WINCAPSII Manager, der die Tabelle für <Tabellennummer> steuert, und die Überprüfung der Tabellenanzeige. Wenn eine Spalte der Eigenschaft kein Symbol “*,” hat, kann das Element in dieser Spalte gesetzt werden. Sie müssen sich hierzu jedoch mit der Kennwortstufe <Programmierer> anmelden.**

Anmerkung: **Wenn Sie die Werte der Einstellungsumgebung ändern, sollten Sie den Controller neu starten.**

Verwandte Begriffe

VER$, GETENV

Beispiel

```
LETENV 6, 22, 5600   'Stellt 5600 für die Werte der Umgebungseinstellwert von Nummer 6
                     'und 22 ein.
```

VER$ (Funktion)

Funktion

Ruft die Version jedes Moduls ab.

Format

VER$ (<Ausdruck>)

Erklärung

Diese Anweisung speichert die Version jedes Moduls entsprechend dem Wert von <Ausdruck> mit einer Zeichenfolge.
Siehe Anhang 9 “Versionstabelle.”

Verwandte Begriffe

GETENV, Versionstabelle (Anhang 9)

Beispiel

```
DEFSTR ls1, ls2, ls3
DIM li1 As Integer
li1 = 1
ls1 = VER$ (1)       'Weist die Version der Hauptplatine der Variablen ls1 zu.
ls2 = VER$ (2)       'Weist die ROM-Version auf der DIO-Karte der Variablen
                     'ls2 zu.
ls3 = VER$ (li1)     'Weist die Versionsinformationen zum Wert von li1
                     'der Variablen ls3 zu.
```

In diesem Beispiel werden Daten in ls1 und ls2 im folgenden Format als Zeichenfolge gespeichert (die nachfolgenden Daten waren die Daten zum Zeitpunkt der Veröffentlichung der Version).

ls1	1.000	07/30/1998
ls2	V1.00	08/03/1998

19.2 Protokoll

ANMERKUNG: Das Steuerprotokoll wurde ab Ver. 1.20 erheblich erweitert. Für weitere Informationen zu Ver. 1.20 oder neuer, siehe Abschnitt 10.7 “Neues Steuerprotokoll” in Kapitel 10 des WINCAPSII Handbuch.

STARTLOG (Anweisung)

Funktion

Startet die Aufzeichnung des Servo-Steuerprotokolls.

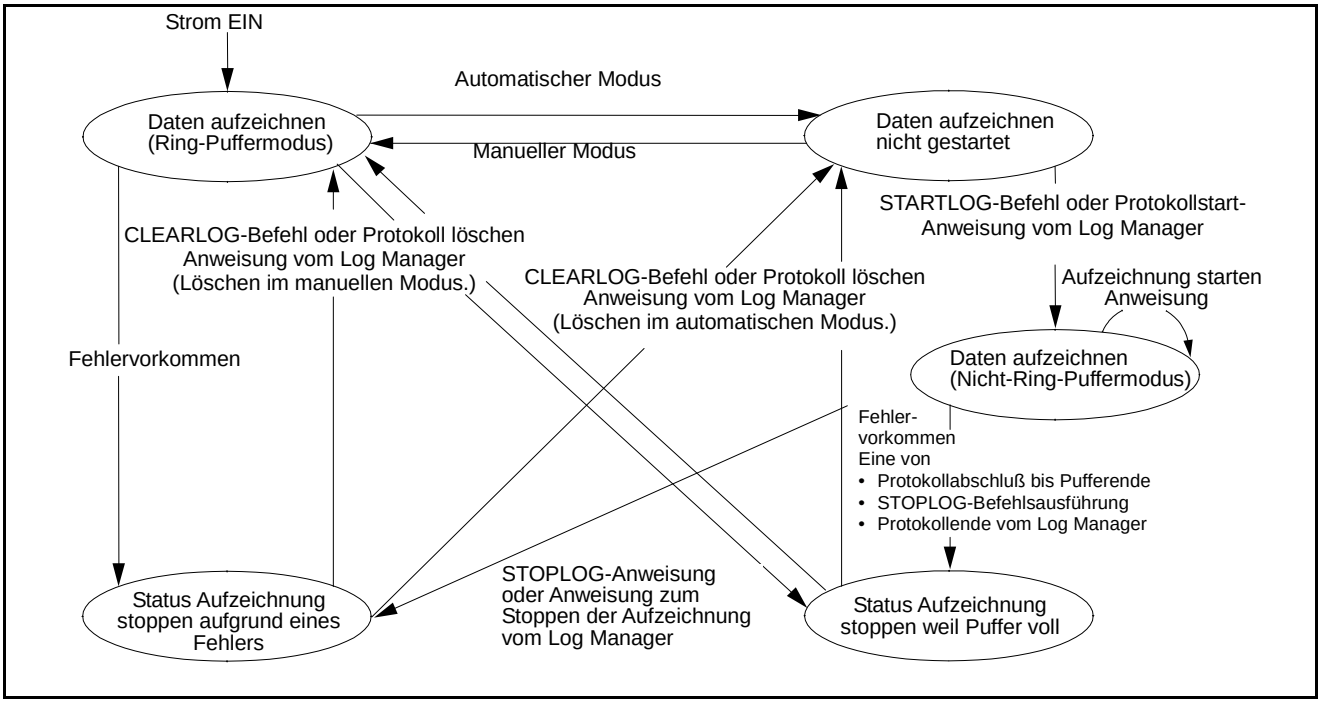
Format

STARTLOG

Erklärung

Verwenden Sie den STARTLOG-Befehl zum Starten der Aufzeichnung des Servo-Steuerprotokolls im Programm.
Zum Sicherstellen der Aufzeichnung des Servo-Steuerprotokolls führen Sie vorher den CLEARLOG-Befehl aus oder löschen Sie das Servo-Protokoll im Log Manager, bevor Sie den STARTLOG-Befehl aufrufen. Nach dem Starten der Aufzeichnung des Servo-Steuerprotokolls ist keine weitere Aufzeichnung möglich, auch wenn Sie den STARTLOG-Befehl ausführen.

Im manuellen Modus und im Programmierprüfmodus startet das System automatisch die Aufzeichnung des Servo-Steuerprotokolls.
Im automatischen Modus beginnt das System mit der Aufzeichnung des Servo-Steuerprotokolls mit Anweisungen zum Starten der Aufzeichnung im Programm oder durch Starten der Aufzeichnung im Log Manager.
Wenn ein Fehler auftritt oder der Puffer voll ist (10 Sekunden), wird die Aufzeichung des Servo-Steuerprotokolls gestoppt. Im manuellen Modus bewahrt das System das Protokoll der letzten 10 Sekunden auf, bis das Programm angewiesen wird, die Aufzeichung zu stoppen.



Protokoll-Betriebsstatus Transition-Diagramm

Verwandte Begriffe

CLEARLOG, STOPLOG

Beispiel

```
STARTLOG            'Beginnt die Aufzeichnung des Servo-Steuerprotokolls.
```

CLEARLOG (Anweisung)

ANMERKUNG: Das Steuerprotokoll wurde ab Ver. 1.20 erheblich erweitert. Für weitere Informationen zu Ver. 1.20 oder neuer, siehe Abschnitt 10.7 “Neues Steuerprotokoll” in Kapitel 10 des WINCAPSII Handbuch.

Funktion

Initialisiert die Aufzeichnung des Servo-Steuerprotokolls.

Format

CLEARLOG

Erklärung

Verwenden Sie den CLEARLOG-Befehl zur Initialisierung des Servo-Steuerprotokolls im Programm.
Wenn der Aufzeichnungspuffer des Servo-Protokolls voll ist oder die Programmausführung wegen eines Fehlers gestoppt wird, müssen Sie den Inhalt des Protokolls löschen (initialisieren), bevor Sie die Aufzeichnung erneut starten. Siehe “Protokoll-Betriebsstatus Transition-Diagramm” auf der vorigen Seite.
Zum fehlerfreien Aufzeichnen des Servo-Steuerprotokolls rufen Sie vor dem STARTLOG-Befehl einen CLEARLOG-Befehl auf. Nach dem Starten der Aufzeichnung ist keine weitere Aufzeichnung möglich, auch wenn Sie den STARTLOG-Befehl ausführen.

Verwandte Begriffe

STARTLOG, STOPLOG

Beispiel

```
CLEARLOG            'Initialisiert die Aufzeichnung des Servo-Steuerprotokolls.
```

STOPLOG (Anweisung)

ANMERKUNG: Das Steuerprotokoll wurde ab Ver. 1.20 erheblich erweitert. Für weitere Informationen zu Ver. 1.20 oder neuer, siehe Abschnitt 10.7 “Neues Steuerprotokoll” in Kapitel 10 des WINCAPSII Handbuch.

Funktion

Stoppt die Aufzeichnung des Servo-Steuerprotokolls.

Format

STOPLOG

Erklärung

Verwenden Sie den STOPLOG-Befehl zum Stoppen des Servo-Steuer-protokolls im Programm.
Wenn im automatischen Modus der Puffer voll ist, nachdem die Aufzeichnung gestartet wurde, wird die Aufzeichung automatisch gestoppt. Sie brauchen daher im Programm keinen STOPLOG-Befehl auszuführen.

Verwandte Begriffe

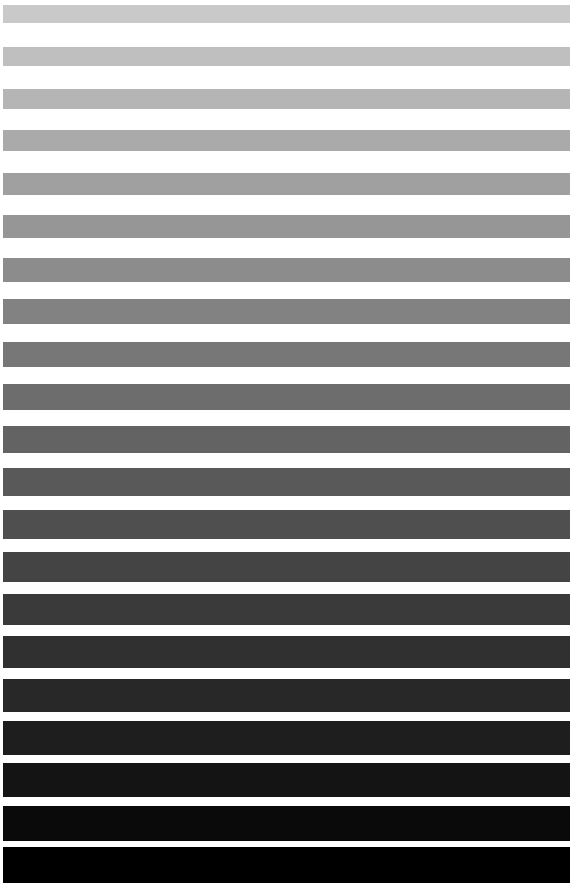
STARTLOG, CLEARLOG

Beispiel

```
STOPLOG             'Stoppt die Aufzeichnung des Servo-Steuerprotokolls.
```

Kapitel 20

Präprozessor



Wenn Sie mit dem PAC Manager ein Programm erstellen, können Sie die in diesem Kapitel beschriebenen Präprozessorbefehle verwenden.
Der PAC Manager ersetzt automatisch die angegebenen Zeichenfolgen oder Dateien entsprechend den Definitionen der Präprozessorbefehle und kompiliert sie anschließend.
Durch Verwendung von Präprozessorbefehlen können Sie das Bibliotheksprogramm verwenden oder ein leicht lesbares Programm beschreiben.

20.1 Symbolkonstanten · Makrodefinitionen

#define (Präprozessoranweisung)

Funktion

Ersetzt eine angegebene Konstante oder einen Makronamen in dem Programm durch eine angegebene Zeichenfolge.

Format

#define <Symbolkonstante> <Zeichenfolge>
oder
#define <Makroname (Argument)> <Argument in Zeichenfolge>

Erklärung

Diese Anweisung ersetzt <Symbolkonstante> oder <Makroname> in dem Programm durch eine angegebene Zeichenfolge. Im Fall eines Makronamens wird dieser durch die bereits enthaltenen Argumente ersetzt.
<Symbolkonstante> oder Zeichenfolgen von <Makronamen> in “ ” (doppelten Anführungszeichen) werden nicht ersetzt.
Sie müssen die #define-Anweisung in einer Zeile angeben.
Sie müssen 1 oder mehrere Leerzeichen zwischen <Symbolkonstante> und <Zeichenfolge> einfügen.
Setzen Sie kein Leerzeichen zwischen einen Makronamen und die Klammern eines Arguments.
Sie können <Symbolkonstante> und <Makroname> neu definieren; Sie müssen sie jedoch mindestens ein Mal mit #undef ungültig machen. Es gelten die zuletzt definierten Angaben.
<Symbolkonstante> und <Makroname> dürfen maximal 64 Zeichen lang sein.
Sie können maximal 2048 Makronamen in einem Programm verwenden. Für die Anzahl der Argumente für Makrofunktionen gibt es keine Begrenzung.

Verwandte Begriffe

#undef

Beispiel

```
#DEFINE NAME “Denso Corporation”
                        'Weist “Denso Corporation” der Symbolkonstanten NAME zu.
#DEFINE mAREA(Radius) PI * POW(Radius, 2)
                        'Deklariert mAREA(Radius) als Makrofunktion.

S1 = NAME               'Weist “Denso Corporation” der Variablen S1 zu.
D1 = mAREA(10)          'Weist den Berechnungswert PI*POW(10,2) der Variablen
                        'D1 zu.
```

#undef (Präprozessoranweisung)

Funktion

Macht eine mit #define oder einer Makrodefinition definierte Symbolkonstante ungültig.

Format

#undef {<Symbolkonstante>|<Makroname>}

Erklärung

Diese Anweisung macht eine mit #define definierte Symbolkonstante oder Makrodefinition im Programm ungültig. Geben Sie nur den Makronamen für Makros mit einem Argument an.

Verwandte Begriffe

#define

Beispiel

```
#UNDEF NAME     'Macht NAME, mit #DEFINE in einer Symboldefinition oder einer
                'Makrodefinition definiert, ungültig.
```

#error (Präprozessoranweisung)

Funktion

Erzwingt das Generieren eines Kompilierungsfehlers bei Ausführung des Befehls #error.

Format

#error [<Meldung>]

Erklärung

Verwenden Sie diese Anweisung, wenn Sie absichtlich einen Fehler generieren wollen, um ein Programm zu testen.
Wenn Sie eine Zeile mit dem Befehl #error kompilieren, tritt ein Fehler auf.
Zu diesem Zeitpunkt erscheint eine Fehlermeldung mit dem in <Meldung> definierten Inhalt.

Beispiel

```
#error “Fehler aufgetreten”    'Bei der Ausführung der Kompilierung erscheint
                               'die Meldung “Fehler aufgetreten”
```

20.2 Datei einbinden

#include (Präprozessoranweisung)

Funktion

Bindet das Präprozessorprogramm ein.

Format

#include “[Pfad] Dateiname”

#include <[Pfad] Dateiname>

Erklärung

Diese Anweisung bindet die Präprozessor-Programdatei an der Position der #include-Anweisung ein. Wenn im Fall von “ ” der Pfad der Datei nicht angegeben ist, sucht das System zunächst im aktuellen Verzeichnis und anschließend im Systemverzeichnis nach der Datei. Bei < > wird nur das Systemverzeichnis durchsucht. Wenn der Pfad mit einem vollständigen Pfadnamen angegeben ist, wird nur das angegebene Verzeichnis durchsucht.
Sie können die #include-Anweisung für eine Datei angeben, die mit einer #include-Anweisung angegeben ist. Sie können hierbei bis zu acht Ebenen verschachteln.
Für die Angabe stehen die Dateierweiterungen H und PAC zur Verfügung.

Beispiel

```
#include “samp1.h”     'Erweitert die Datei samp1.h in dieser Zeile.
```

20.3 Optimierung

#pragma optimize (Präprozessoranweisung)

Funktion

Gibt die auszuführende Optimierung für jedes Programm an.

Format

#pragma optimize (“<Optionsliste>”, {on/off})

Erklärung

Diese Anweisung gibt die auszuführende Optimierung für jedes Programm an. Geben Sie diese Anweisung in der Zeile vor der PROGRAM-Deklarationsanweisung oder in einer gültigen Anweisungszeile an.
Geben Sie für <Optionsliste> eine beliebige Anzahl Parameter aus der folgenden Tabelle an.

Parameter	Optimierungstyp
A	Löscht den Array-Untersuchungscode.
C	Löscht den Code für die Berechnung der Zykluszeit.

Anmerkungen: **Durch eine effektive Verwendung der Optimierungsoption können Sie die Ausführungsgeschwindigkeit einer Nicht-Aktionsanweisung um 10% bis 20% steigern. Sie finden die Programmeinstellungstabelle, wenn Sie [Datei] und anschließend [Projekteinstellung] im PAC Manager auswählen. Hier gibt es außerdem eine Option zur Programmoptimierung.**

Beispiel

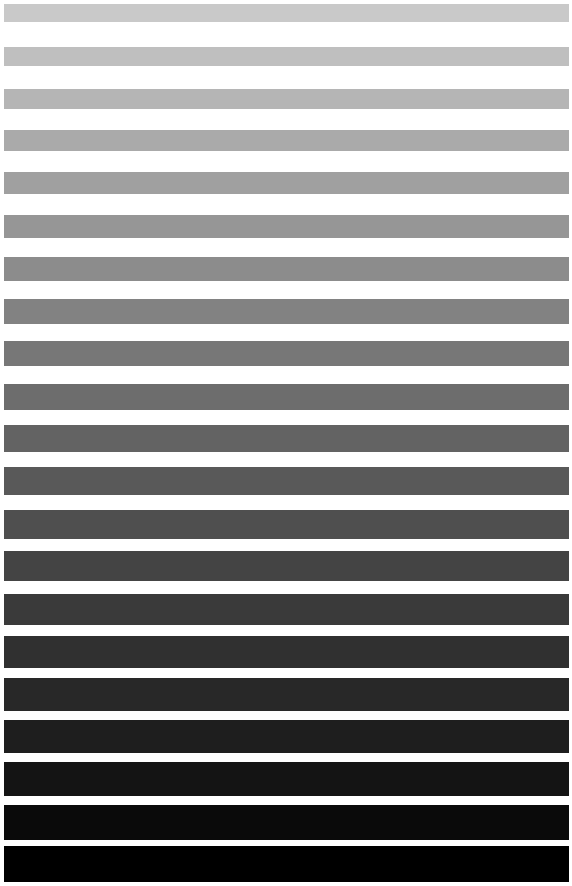
```
#pragma optimize( “ac”, ON )   'Löscht den Array-Prüfungscode und macht den
                               'Berechnungscode für die Zykluszeit gültig.
```

Anmerkung

Dieser Befehl hat Priorität über den unter “Projekteinstellung” gesetzten Status des PAC Manager. Wenn daher der Code für die Array-Prüfung gelöscht wird, prüft das System den Bereich der Array-Indexzahlen nicht, obwohl der Fehlerprozeß mit ON ERROR GOTO gesetzt wurde.

Kapitel 21

Bildverarbeitungssteuerung (Option)



Dieses Kapitel listet alle Befehle auf, die zur Verwendung mit der µBildverarbeitungskarte und µVision-21 zur Verfügung stehen.

Vorsichtsmaßnahmen bei der Verwendung von Bildverarbeitungsbefehlen.

(1) Zur Verwendung von Bildverarbeitungsbefehlen ist eine optionale µBildverarbeitungskarte erforderlich.

(2) Einem Bildverarbeitungsbefehl muß ein Befehl für die Priorität der Bildverarbeitung (TAKEVIS) vorangestellt werden.

Beispiel:
```
TAKEVIS
CAMIN 1
VISPLNOUT 0
    :
```

(3) Wenn bei der Ausführung eines Bildverarbeitungsbefehls (mit Ausnahme der Kamera-Eingabebefehle) ein Fehler auftritt, erscheint die Fehlermeldung bei der Ausführung des nächsten Bildverarbeitungsbefehls.

Beispiel:
```
WINDMAKE R,1,100,10,0,2
VISPRJ 1,100,100,1,128     An dieser Stelle tritt ein
                           "Fensterformfehler" auf.
WINDDISP 1                 Bei der Ausführung dieses
    .                      Befehls wird die Fehlermeldung
    .                      angezeigt.
    .
```

(4) Bei der Ausführung eines TAKEVIS-Befehls während der Kommunikation zwischen dem WINCAPSII Vision Manager und der µBildverarbeitungskarte tritt ein Fehler auf.

21.1 Kompatibilität mit der konventionellen µVision-15

Die Funktionen der konventionellen µVision-15 werden mit Ausnahme der Befehle zur Ausgabe (parallele E/A, RS232C) übernommen. Es gibt jedoch einen Unterschied im Format der Anweisungen. Wenn Sie ein Programm bearbeiten, das auf ein konventionelles Programm verweist, schlagen Sie in der entsprechenden Tabelle weiter unten nach.

	µVision-15	µBildverarbeitungskarte	Beschreibung	Seite
Bild-Ein-/Ausgabe	VIN	CAMIN	Kamerabilder erfassen	21-3
	AOUT	VISCAMOUT	Kamerabildanzeige	21-7
	VOUT	VISPLNOUT	Prozeßfensteranzeige	21-8
		VISOVERLAY	Overlay-Einstellung	21-9
Fenster-einstellung	WNDALN	WINDMAKE	Stellt die Fensterdaten ein.	21-14
	WNDCIR			
	WNDDPT			
	WNDELP			
	WNDRCT			
	WNDSCT			
	HWIND			
	CLRWND	WINDCLR	Löscht die eingestellten Fenstereinstellungsdaten.	21-19
	WNDCPY	WINDCOPY	Kopiert die eingestellten Fensterdaten.	21-20
	WDISP	WINDDISP	Fensteranzeige	21-23
Zeichnen	SCREEN	VISSCREEN	Stellt die Zeichnungsbedingungen ein.	21-24
	CLS	VISCLS	Löscht die Anzeigen.	21-27
	PUTP	VISPUTP	Zeichnet Punkte.	21-29
	LINE	VISLINE	Zeichnet Linien (Länge und Winkel).	21-30
	LNDPT	VISPTP	Zeichnet Linien (2 Punkt).	21-31
	RECT	VISRECT	Zeichnet Rechtecke.	21-32
	CIRCLE	VISCIRCLE	Zeichnet Kreise.	21-33
	ELIPSE	VISELLIPSE	Zeichnet Ellipsen.	21-34
	SECT	VISSECT	Zeichnet Sektoren.	21-35
	PAINT	VISRECT	Zeichnet Rechtecke.	21-32
		VISCIRCLE	Zeichnet Kreise.	21-33
		VISELLIPSE	Zeichnet Ellipsen.	21-34

	μVision-15	μBildverarbeitungs-karte	Beschreibung	Seite
Zeichnen	CROSS	VISCROSS	Zeichnet Kreuzsymbole.	14-36
	CRSALN			
	LOC	VISLOC	Findet Positionen für die Zeichenanzeige.	14-37
	PRINT	VISDEFCHAR	Zeicheneinstellung für Zeichnung	14-39
		VISPRINT	Zeichnet Zeichen.	14-40
Bildverarbeitung	GETP	VISGETP	Ruft die Helligkeit der angegebenen Koordinaten ab.	14-42
	HIST	VISHIST	Führt Histogramm aus.	14-43
	HCLR			
		VISREFHIST	Ruft Ergebnisse von Histogrammen ab.	14-44
	LEVEL	VISLEVEL	Berechnet Binärcodestufen.	14-45
	BINA	VISBINA	Binärverarbeitung	14-47
	RBINA	VISBINAR	Binärcodeanzeige	14-49
	AREA	VISMEASURE	Berechnet Funktionen (Bereich, Schwerpunkt, Hauptachsenwinkel etc.).	14-55
	BIGT			
	CENTR			
	MOMENT			
	STRT			
	PROJ	VISPROJ	Projektionsprozeß	14-58
	EDGE	VISEDGE	Mißt Kanten.	14-60
Suchfunktion	CORN	SHCORNER	Sucht Ecken.	14-86
	CIRC	SHCIRCLE	Sucht Kreise.	14-89

21.2 Bildeingabe und -ausgabe

CAMIN (Anweisung)

Funktion

Speichert ein Abbild von der Kamera im Bildspeicher (Prozeßfenster).

Format

CAMIN <Kameranummer>[, <Speichernummer> [, <Tabellennummer>]]

Erklärung

<Kameranummer>	Gibt die Kameranummer (1 oder 2) an.
<Speichernummer>	Gibt die Nummer des Speichers (Prozeßfenster) (0 bis 3) an. Wird diese Option nicht angegeben, gilt der Standardwert 0.
<Tabellennummer>	Gibt die Nummer der zu speichernden Nachschlagetabelle von 0 bis 15 an. Wenn diese Option nicht angegeben wird, gilt der Standardwert 0.

Anmerkung (1):	**Wenn keine Kamera angeschlossen ist oder wenn wegen einer Fehlfunktion keine Eingabe verfügbar ist, tritt ein Fehler auf.**
Anmerkung (2):	**Wenn beim Speichern des Bildes die Tabellennummer nicht 0 ist, wird die Tabelle geändert. In diesem Fall kann die Anzeige durcheinander geraten; dies ist jedoch kein Fehler.**
Anmerkung (3):	**Nach der Ausführung wird die Nummer der Tabelle automatisch auf Null zurückgesetzt.**
Anmerkung (4):	**Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.**

Verwandte Begriffe

CAMMODE, CAMLEVEL, VISDEFTABLE

Beispiel

```
CAMIN 1,0,0      'Wandelt ein Bild von Kamera Nummer 1 mit Tabellennummer 0
                 '(mit der gleichen Helligkeit wie vom Kamerabild) um und
                 'speichert es im Speicher 0.
DELAY 2000       'Stoppt 2 Sekunden lang.
CAMIN 1          'Das gleiche Ergebnis kann mit CAMIN 1,0,0 abgerufen werden.
DELAY 2000       'Stoppt 2 Sekunden lang.
I1 = 1           '
I2 = 0           '
I3 = 3           '
CAMIN I1,I2,I3   'Wandelt ein Bild von Kamera Nummer 1 mit Tabelle 3
                 '(invertiert) um und speichert es im Speicher 0.
VISPLNOUT 0      'Gibt ein Bild in Speicher 0 auf dem Monitor
                 'als feststehendes Bild aus.
```

CAMMODE (Anweisung)

Funktion

Stellt die zum Speichern eines Kamerabildes verwendete Funktion ein.

Format

CAMMODE <Kameranummer>, <Funktion>, <Speichermethode>

Erklärung

<Kameranummer> Gibt die Kameranummer (1 oder 2) an.

<Funktion> Gibt die Funktion der Kamera (0 oder 1) an.

0: Normal (bei der normalen Einstellung)
1: Rücksetzungsfunktion. (Nachdem der Kamerabetrieb zurückgesetzt wurde, wird ein Kamerabild gespeichert.)
Weiter Informationen, siehe Abschnitt 2.5.2 “Allgemeine Informationen zur Kamera” in Kapitel 2 des Installations- und Wartungshandbuches.

<Speichermethode> Geben Sie die Speichermethode an (0 oder 1).

0: Frame lesen
Speichert ein Bild von der Kamera für 1 Frame. Die Auflösung in vertikaler Richtung entspricht dem Maximum.
1: Feld lesen
Bei Verwendung der Schließfunktion bei einer Feldverschlußkamera stellen Sie diese Option ein.
Speichern Sie ein Bild ohne Verzerrung und praktisch ohne Verzögerung (1/60 Sekunden) zwischen den Feldern.
Die Auflösung in Kamerarichtung wird dadurch jedoch halbiert.

Anmerkung (1):	**Wenn Sie diese Anweisung nicht setzen, wird der anfänglich eingestellte Wert verwendet.**
Anmerkung (2):	**Diese Anweisung ändert die anfängliche Einstellung nicht. Wenn Sie das System erneut starten und dabei die Stromversorgung ausgeschaltet ist, gehen die eingestellten Werte verloren.**
Anmerkung (3):	**Überprüfen Sie die Kameraverbindung. Wenn die Kamera fehlerhaft ist, überprüfen Sie den Status von VISSTATUS. Wenn der Status von VISSTATUS (0) gleich -1 ist, ist die Kamera fehlerhaft; falls die Kamera nicht fehlerhaft ist, wird 0 zurückgegeben.**
Anmerkung (4):	**Zur Ausführung dieser Anweisung ist eine Bildverarbeitungskarte (Option) erforderlich.**

Verwandte Begriffe

CAMIN, VISSTATUS

Beispiel

```
CAMMODE 1,0,0      'Stellt die Funktion der Kamera 1 auf normal ein und die
                   'Speichermethode auf frame.
I1 = VISSTATUS(0)  'Bei normal ist I1 = 0
IF I1 = 0 THEN
    CAMIN 1        'Wandelt ein Bild von Kamera Nummer 1 mit Tabelle 0 um
                   '(mit der gleichen Helligkeit wie vom Kamerabild) und
                   'speichert es in Speicher 0.
    VISPLNOUT 0    'Gibt ein Bild (unbewegtes Bild) in Speicher 0 auf dem
                   'Monitor aus.
    VISLOC 10,10   'Gibt die anzuzeigende Position an.
    VISPRINT “Normal lesen”
                   'Zeigt Zeichen auf dem Bildschirm an.
ELSE
    VISLOC 10,10   'Gibt die anzuzeigende Position an.
    VISPRINT “Kamera-Fehlfunktion”
                   'Zeigt Zeichen auf dem Bildschirm an.
END IF
```

CAMLEVEL (Anweisung)

Funktion

Stellt die Eingangsstufe für Kamerabilder ein.

Format

CAMLEVEL <Kameranummer>, <Untere Stufe>, <Obere Stufe>

Erklärung

<Kameranummer>	Gibt die Kameranummer (1 oder 2) an.
<Untere Stufe>	Gibt den unteren Grenzwert zum Lesen der Kamerabilder an (0 bis 93).
<Obere Stufe>	Gibt den oberen Grenzwert zum Lesen der Kamerabilder an (7 bis 100).

Anmerkung (1): Diese Anweisung setzt den unteren Grenzwert des maximalen Eingabebereiches auf 0% und den oberen Grenzwert auf 100%.

Anmerkung (2): Der Bereich zwischen dem eingestellten unteren Grenzwert und dem oberen Grenzwert ist in 256 Intensitätsstufen unterteilt.

Anmerkung (3): Für die Einstellung muß gelten: 0 ≤ unterer Grenzwert < oberer Grenzwert ≤ 100 und oberer Grenzwert – unterer Grenzwert ≥ 7.

Anmerkung (4): Sie erhalten detailliertere Bildinformationen, wenn Sie diesen Befehl zum Anpassen eines dunklen oder hellen Bildes verwenden oder zum teilweisen Vergrößern der Auflösung oder Helligkeit.

Anmerkung (5): Wenn Sie diese Anweisung nicht setzen, werden die anfänglich eingestellten Werte verwendet.

Anmerkung (6): Diese Anweisung ändert die anfängliche Einstellung nicht. Wenn Sie das System erneut starten und dabei die Stromversorgung ausgeschaltet ist, gehen die eingestellten Werte verloren.

Anmerkung (7): Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.

Anmerkung (8): Dieser Befehl wird gültig bei der Ausführung des nächsten Kameraeingangsbefehls (CAMIN).

Verwandte Begriffe

CAMIN, VISCAMOUT

Beispiel

```
CAMLEVEL 1,0,100    'Stellt die Stufe des Eingangsbereiches für Kamera 1 auf das
                    'Maximum ein.
VISCAMOUT 1         'Zeigt ein Bild (dynamisches Bild) von der Kamera auf
                    'dem Monitor an.
CAMIN 1             'Wandelt ein Bild von Kamera Nummer 1 mit Tabelle 0 um
                    '(mit der gleichen Helligkeit wie vom Kamerabild) und
                    'speichert es in Speicher 0.
DELAY 2000          'Stoppt 2 Sekunden lang.
CAMLEVEL 1,30,80    'Setzt den unteren Grenzwert auf 30% und den oberen Grenzwert
                    'auf 80% für Kamera 1.
VISCAMOUT 1         'Zeigt ein Bild (dynamisches Bild) von der Kamera auf
                    'dem Monitor an.
CAMIN 1
```

VISCAMOUT (Anweisung)

Funktion

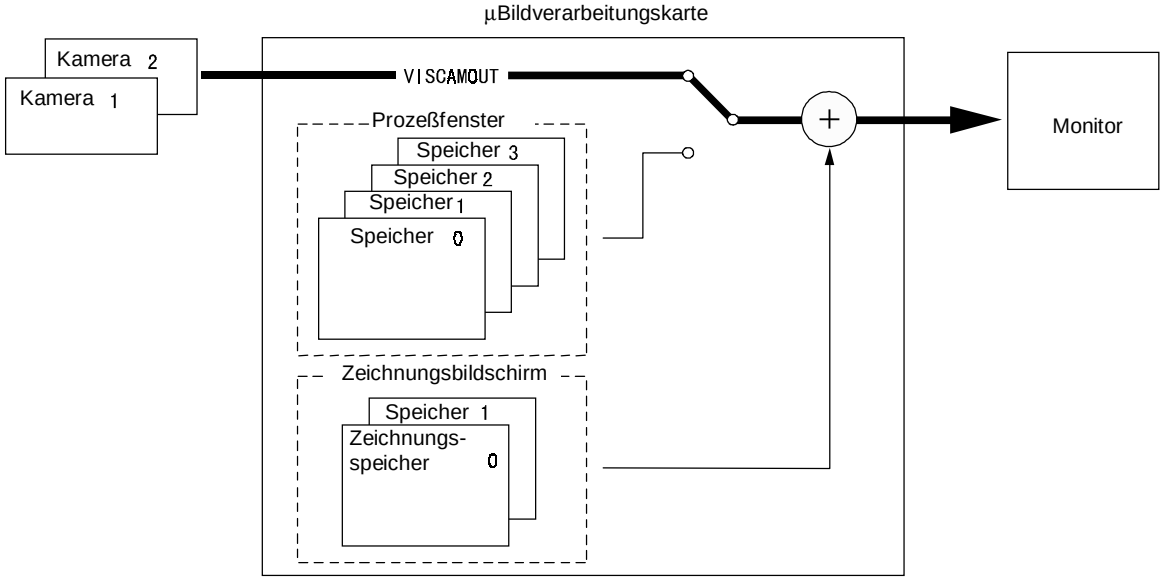
Zeigt ein Bild von der Kamera auf dem Monitor an.

Format

VISCAMOUT <Kameranummer>[, <Tabellennummer>]

Erklärung

<Kameranummer> Gibt die Kameranummer (1 oder 2) an.
<Tabellennummer> Gibt die Nummer der anzuzeigenden Nachschlagetabelle (0~15) an. Wird diese Option nicht angegeben, gilt der Standardwert 1.



Anmerkung: Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.

Verwandte Begriffe

VISDEFTABLE, VISPLNOUT, VISOVERLAY, CAMMODE

Beispiel

```
VISCAMOUT 1,1      'Wandelt ein Bild (dynamisches Bild) von Kamera 1 mit
                   'Tabelle 1 (0~175, 70% Helligkeit komprimiert) um und zeigt
                   'es auf dem Monitor an.
VISCAMOUT 1        'Es wird das gleiche Ergebnis erzielt wie mit VISCAMOUT 1,1.
```

VISPLNOUT (Anweisung)

Funktion

Zeigt ein Abbild im Speicher auf dem Monitor an.

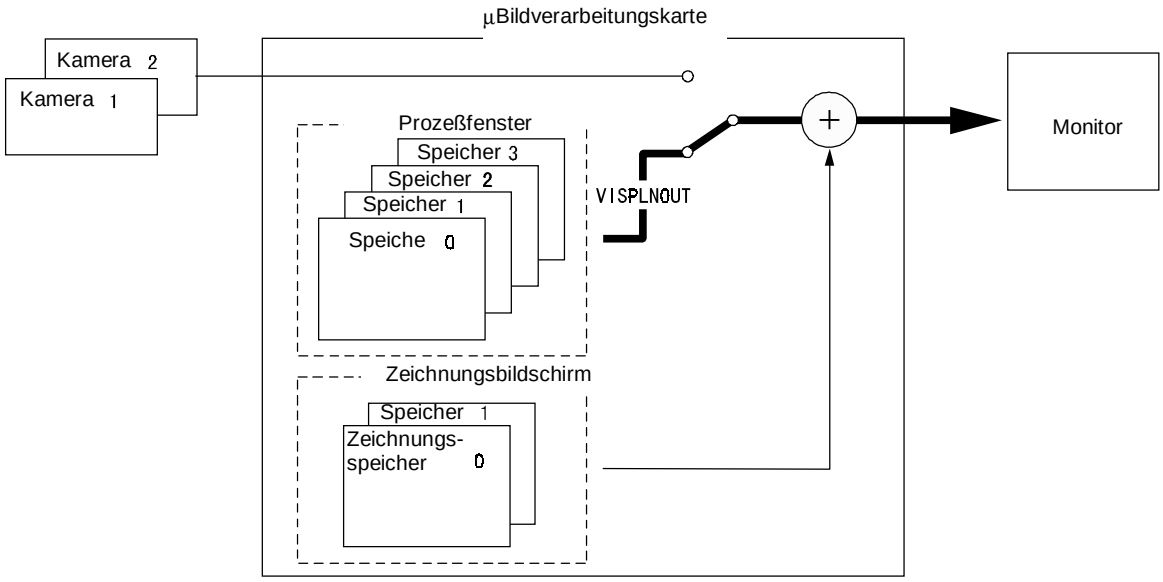
Format

VISPLNOUT <Speichernummer>[, <Tabellennummer>]

Erklärung

<Speichernummer> Gibt die Nummer des Speichers (Prozeßfenster) (0 bis 3) an.

<Tabellennummer> Gibt die Nummer der anzuzeigenden Nachschlagetabelle (0~15) an. Wird diese Option nicht angegeben, gilt der Standardwert 1.



Verwandte Begriffe

VISDEFTABLE, VISCAMOUT, VISOVERLAY

Beispiel

```
VISPLNOUT 0,1      'Wandelt ein Bild (unbewegtes Bild) in Speicher 0 mit
                   'Tabelle 1 (0~175, 70% Helligkeit komprimiert) um und zeigt
                   'es auf dem Monitor an.
VISPLNOUT 0        'Es wird das gleiche Ergebnis erzielt wie mit VISPLNOUT 0,1.
```

VISOVERLAY (Anweisung)

Funktion

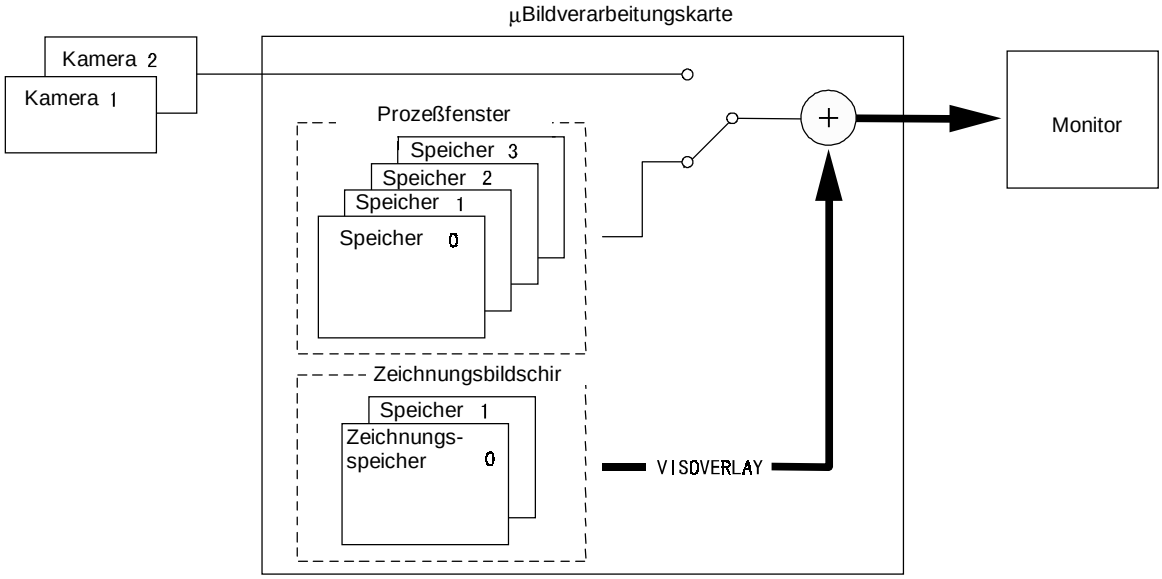
Zeigt Zeichenbildschirm-Informationen auf dem Monitor an.

Format

VISOVERLAY <Nummer>

Erklärung

<Nummer> Stellt den Zeichnungsbildschirm (0 bis 3) ein.
0: Zeigt keinen Zeichnungsbildschirm an.
1: Zeigt den Zeichnungsbildschirm 0 an.
2: Zeigt den Zeichnungsbildschirm 1 an.
3: Zeigt die Zeichnungsbildschirme 0 und 1 gleichzeitig an.



Verwandte Begriffe

VISCAMOUT, VISPLNOUT

Beispiel

```
VISOVERLAY 3            'Stellt den zu zeichnenden Zielbildschirm ein.
VISSCREEN 1,0           '
VISCLS 0                '
VISLOC 10,10            'Gibt die anzuzeigende Position an.
VISPRINT “Zeichnen auf dem Zeichnungsbildschirm 0”
                        'Zeichnet Zeichen auf dem Bildschirm.
VISSCREEN 1,1           'Gibt den zu zeichnenden Zielbildschirm an.
VISLOC 10,11            'Gibt die anzuzeigende Position an.
VISPRINT “Zeichnen auf dem Zeichnungsbildschirm 0”
                        'Zeichnet Zeichen auf dem Bildschirm.
VISOVERLAY 0            'Stoppt die Anzeige des Zeichnungsbildschirms.
DELAY 5000              'Stoppt 5 Sekunden lang.
VISOVERLAY 1            'Zeigt den Zeichnungsbildschirm 0 auf dem Monitor an.
DELAY 5000              'Stoppt 5 Sekunden lang.
VISOVERLAY 2            'Zeigt den Zeichnungsbildschirm 1 auf dem Monitor an.
DELAY 5000              'Stoppt 5 Sekunden lang.
VISOVERLAY 3            'Zeigt die Zeichungsbildschirme 0 und 1 auf dem
                        'Monitor an.
```

VISDEFTABLE (Anweisung)

Funktion

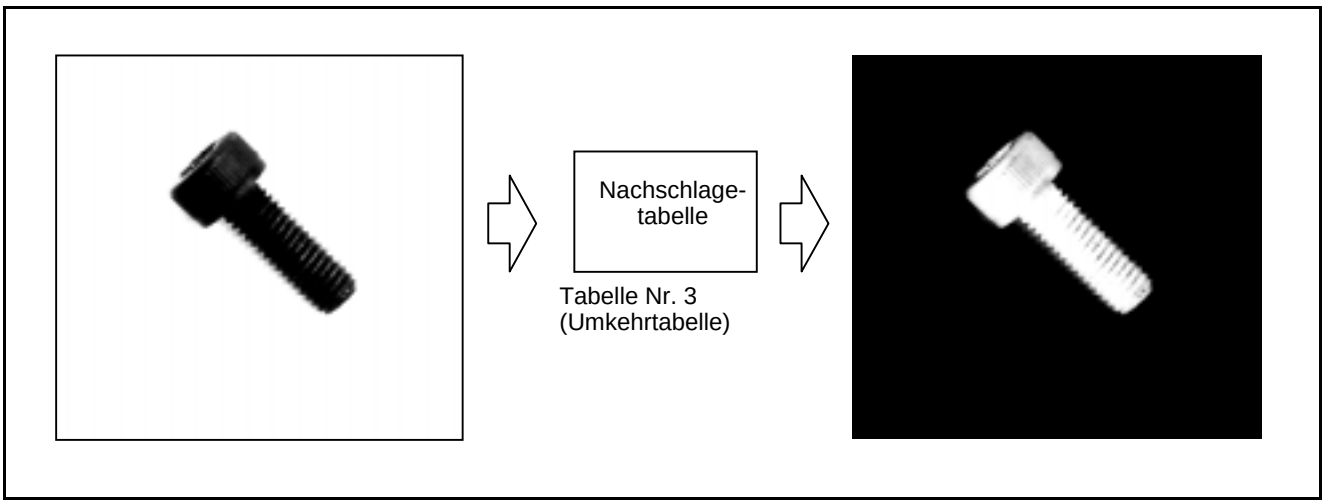
Liest Abbilder von der Kamera ein und stellt die Nachschlage-Tabellendaten für die Abbildausgabe ein.

Format

VISDEFTABLE <Tabellennummer>, <Eingabewert>, <Ausgabewert>

Erklärung

<Tabellennummer>	Gibt die Nummer der Nachschlagetabelle (5 bis 15) an.
<Eingabewert>	Gibt den Eingabewert der Tabelle (0 bis 255) an.
<Ausgabewert>	Gibt den Ausgabewert der Tabelle (0 bis 255) an.



Beispiel der Umwandlung mit der Nachschlagetabelle

Anmerkung (1): Wenn Sie diese Anweisung nicht setzen, verwendet die Anweisung die anfänglichen Einstellungswerte.

Anmerkung (2): Die Ausführung dieser Anweisung dauert ca. 2 bis 3 Sekunden.

Anmerkung (3): Die als hilfreich betrachteten Tabellen werden vorab für die Tabellenummern 0 bis 4 gesetzt. Mit diesem Befehl können Sie keine Bearbeitung vornehmen.
Tabelle 0: Normal (0 bis 255)
Tabelle 1: 70% Helligkeit komprimiert (0 bis 175)
Tabelle 2: γ Kompensierung
Tabelle 3: Umgekehrt
Tabelle 4: 70% Helligkeit komprimiert und umgekehrt

Anmerkung (4): Zur Ausführung dieser Anweisung ist eine μBildverarbeitungskarte (Option) erforderlich.

Anmerkung (5): Schalten Sie während der Ausführung dieser Anweisung den Controller nicht aus. Wenn Sie den Controller ausschalten, erkennt er in der nächsten Einschaltsequenz nicht, daß er nicht normal beendet wurde und initialisiert die Bildverarbeitungsinformationen neu.

Verwandte Begriffe

CAMIN, VISCAMOUT, VISPLNOUT, VISREFTABLE

Beispiel

```
VISSCREEN 1,0,1     '
VISCLS 0            '
VISCAMOUT 1,1       'Wandelt ein Bild (dynamisches Bild) von Kamera 1 mit Tabelle 1
                    '(0~175, 75% Helligkeit komprimiert) um und zeigt es
                    'auf dem Monitor an.
DELAY 5000          'Stoppt 5 Sekunden lang.
VISDEFCHAR 4,4,2    '
FOR I1 = 0 TO 255   '
    VISLOC 10,10    '
    VISPRINT I1     '
    VISDEFTABLE 5,I1,255-I1
                    'Stellt Tabellennummer 5 ein.
NEXT I1             '
VISCAMOUT 1,5       'Kehrt ein Bild (dynamisches Bild) von Kamera 1 um und zeigt
                    'es auf dem Monitor an.
```

VISREFTABLE (Funktion)

Funktion

Verweist auf Daten in der Nachschlagetabelle.

Format

VISREFTABLE (<Tabellennummer>, <Eingabewert>)

Erklärung

<Tabellennummer>	Gibt die Nummer der anzuzeigenden Nachschlagetabelle (0 bis15) an.
<Eingabewert>	Gibt den Eingabewert der Tabelle (0 bis 255) an.

Verwandte Begriffe

VISDEFTABLE

Beispiel

```
VISSCREEN 1,0,1                '
VISCLS 0                       '
FOR I1 = 0 TO 255              '
    I2 = VISREFTABLE(1,I1)     'Ruft Daten aus Tabelle 1 ab.
    VISLOC 10,10               'Gibt die anzuzeigende Position an.
    VISDEFCHAR 1,1,2           'Gibt die anzuzeigenden Zeichen an.
    VISPRINT “Data”;I1;“=”;I2  'Zeigt das Fenster an.
NEXT I1                        '
```

21.3 Fenstereinstellung

WINDMAKE (Anweisung)

Funktion

Zeichnet einen Bereich für die Abbildverarbeitung.

Format

WINDMAKE <Fensterform>, <Fensternummer>, <Parameter 1>, <Parameter 2>, …

Linienfenster (2-Punkt-Angabe):

WINDMAKE P, <Fensternummer>, <Startpunkt X-Koordinate>, <Startpunkt Y-Koordinate>, <Endpunkt X-Koordinate>, <Endpunkt Y-Koordinate>

Linienfenster (Längen- und Winkelangabe):

WINDMAKE L, <Fensternummer>, <Länge>[, <Winkel>]

Kreisfenster:

WINDMAKE C, <Fensternummer>, <Radius>

Ellipsenfenster:

WINDMAKE E, <Fensternummer>, <Breite>, <Höhe>

Sektorfenster:

WINDMAKE S, <Fensternummer>, <Außendurchmesser>, <Innendurchmesser>, <Startwinkel>, <Endwinkel>, <Partitionswinkel>[, <Modus>]

Rechteckfenster:

WINDMAKE R, <Fensternummer>, <Breite>, <Höhe> [, <Winkel>[, <Modus>]]

Erklärung

<Fensterform> Kennzeichnet die Fensterform.

- P: Linienfenster Punkt-zu-Punkt (2-Punkt-Angabe)
- L: Linienfenster Linie (Längen- und Winkelangabe)
- C: Kreisfenster Kreis
- E: Ellipsenfenster Ellipse
- S: Sektorfenster Abschnitt
- R: Rechteckfenster Rechteck

<Fensternummer> Gibt die Fensternummer (0 bis 511) an.

<Parameter> Gibt den Wert für die einzelnen Fensterformen an.

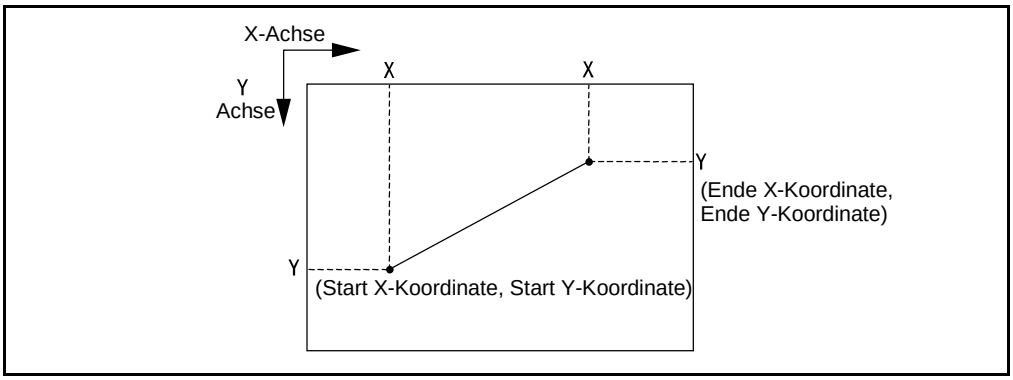
Linienfenster (2-Punkt-Angabe):

<Startpunkt X-Koordinate> Gibt die X-Koordinate des Linienstartpunktes an (0 bis 511).

<Startpunkt Y-Koordinate> Gibt die Y-Koordinate des Linienstartpunktes an (0 bis 479).

<Endpunkt X-Koordinate> Gibt die X-Koordinate des Linienendpunktes an (0 bis 511).

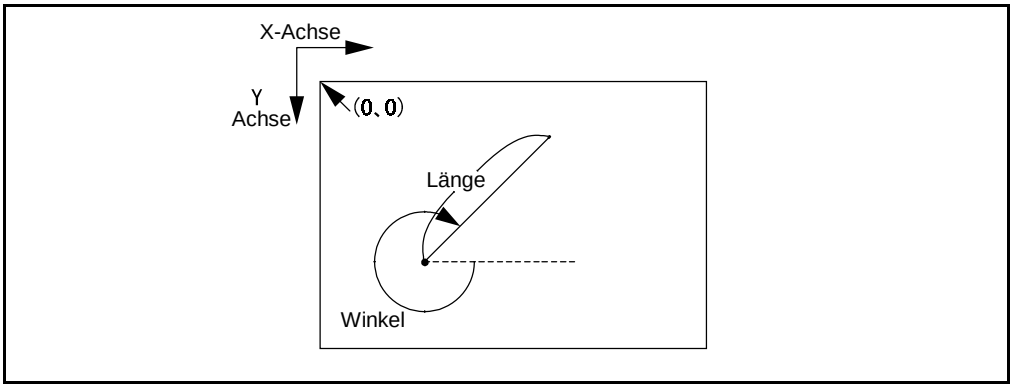
<Endpunkt Y-Koordinate> Gibt die Y-Koordinate des Linienendpunktes an (0 bis 479).



Linienfenster (Längen- und Winkelangabe):

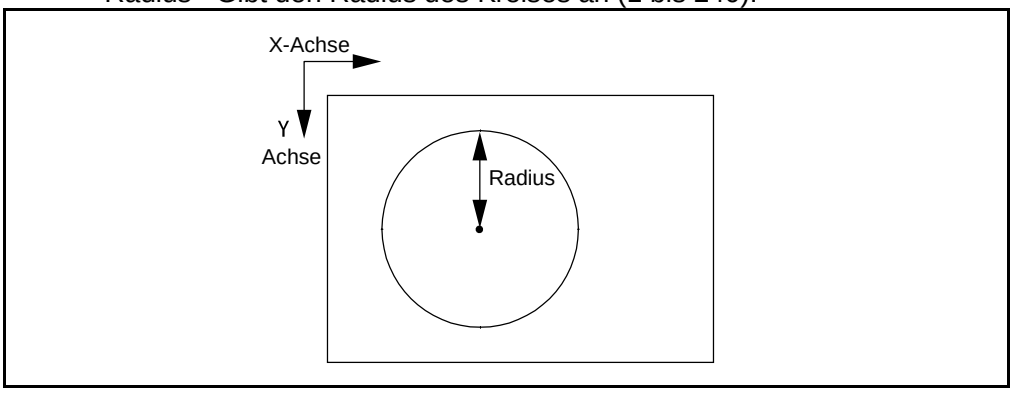
<Länge> Gibt die Länge der Linie von 1 bis 512 an.

<Winkel> Gibt den Winkel der Linie an. Wird diese Option nicht angegeben, gilt der Standardwert 0 Grad.



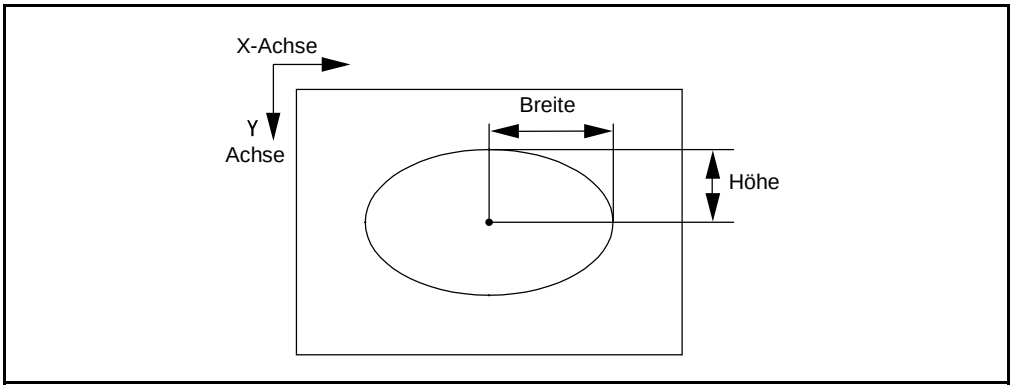
Kreisfenster:

<Radius> Gibt den Radius des Kreises an (1 bis 240).



Ellipsenfenster:

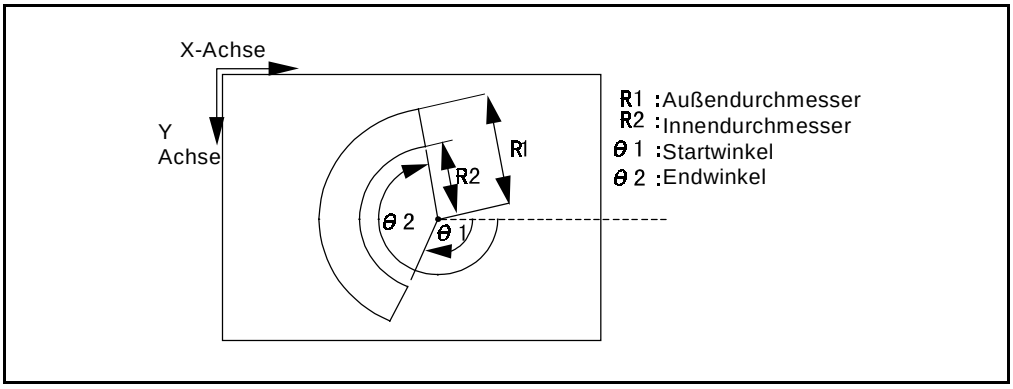
<Breite> Gibt die Breite einer Ellipse an (1 bis 256).
<Höhe> Gibt die Höhe einer Ellipse an (1 bis 240).



Sektorfenster:

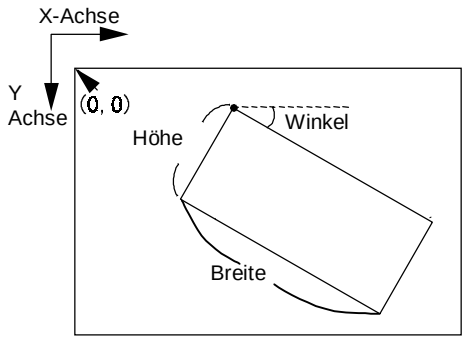
<Außendurchmesser> Gibt den Außendurchmesser eines Sektors (Außendurchmesser > Innendurchmesser) an (1 bis 9999).
<Innendurchmesser> Gibt den Innendurchmesser eines Sektors (Außendurchmesser > Innendurchmesser) an (1 bis 9999).
<Startwinkel> Gibt den Startwinkel eines Sektors an (-720 bis 720).
<Endwinkel> Gibt den Endwinkel eines Sektors an (-720 bis 720).
<Partitionswinkel> Gibt die Auflösung für die Verarbeitung an (0,1 bis 360).
<Modus> Gibt den Zeichnungsmodus an (0 bis 2).

0: Gibt die VISPROJ und VISEDGE Scan-Richtung zum Meridian an.
1: Gibt die VISPROJ und VISEDGE Scan-Richtung zur Peripherie eines Kreises an.
2: Gibt keine VISPROJ und VISEDGE Scan-Richtung an.
Wenn diese Option nicht angegeben wird, gilt die Angabe 0.



Rechteckfenster:

<Breite> Gibt die Breite eines Rechteckes an (1 bis 512).
<Höhe> Gibt die Höhe eines Rechteckes an (1 bis 480).
<Winkel> Gibt den Winkel der Linie an. Wenn diese Option nicht angegeben wird, gilt die Angabe 0 Grad (-720 bis 720).
<Modus> Gibt den Zeichnungsmodus an (0 bis 2).
0: Gibt die VISPROJ und VISEDGE Scan-Richtung zur Breite an.
1: Gibt die VISPROJ und VISEDGE Scan-Richtung zur Höhe an.
2: Gibt keine VISPROJ und VISEDGE Scan-Richtung an.
Wird diese Option nicht angegeben, gilt der Standardwert 0.



Anmerkung (1): Diese Anweisung ändert die anfängliche Einstellung nicht. Beim Neustarten des Systems mit ausgeschalteter Stromversorgung werden die Anfangswerte wiederhergestellt.

Anmerkung (2): Wenn eine dauerhafte Änderung erforderlich ist, löschen Sie die Daten mit dem WINCAPSII VisManager.

Anmerkung (3): Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.

Verwandte Begriffe

VISHIST, VISBINA, VISFILTER, VISMASK, VISMEASURE, VISPROJ, VISEDGE, VISREADQR, BLOB, SHMODEL, SHCORNER, SHCIRCLE

Beispiel

```
VISSCREEN 1,0,1                    '
VISCLS 0                           '
VISCAMOUT 1                        '
CAMIN 1                            '
VISPLNOUT 0                        '
WINDMAKE P,1,50,100,100,150        'Erstellt ein Linienfenster (2-Punkt-Angabe).
WINDMAKE L,2,100,45                'Erstellt ein Linienfenster (Längen- und
                                   'Winkelangabe).
WINDMAKE C,3,50                    'Erstellt ein Kreisfenster.
WINDMAKE E,4,50,100                'Erstellt ein Ellipsenfenster.
WINDMAKE S,5,100,80,90,300,1,2     'Erstellt ein Sektorfenster.
WINDMAKE R,6,100,50,45,2           'Erstellt ein Rechteckfenster.
VISMEASURE 1,100,100,1,1,128       '
WINDDISP                               '
VISMEASURE 2,150,150,1,1,128           '
WINDDISP 2                             '
VISMEASURE 3,200,200,1,1,128           '
WINDDISP 3                             '
VISMEASURE 4,250,250,1,1,128           '
WINDDISP 4                             '
VISMEASURE 5,300,300,1,1,128           '
WINDDISP 5                             '
VISMEASURE 6,350,350,1,1,128           '
WINDDISP 6                             '
```

WINDCLR (Anweisung)

Funktion

Löscht eine eingestellte Fensterinformation.

Format

WINDCLR <Fensternummer>

Erklärung

<Fensternummer> Gibt die Fensternummer (0 bis 511) an.

Anmerkung (1):	**Diese Anweisung ändert die anfängliche Einstellung nicht. Beim Neustarten des Systems mit ausgeschalteter Stromversorgung werden die Anfangswerte wiederhergestellt.**
Anmerkung (2):	**Wenn Daten dauerhaft gelöscht werden müssen, löschen Sie sie mit dem WINCAPSII VisManager.**
Anmerkung (3):	**Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.**

Verwandte Begriffe

WINDMAKE

Beispiel

```
VISSCREEN 1,0,1                         '
VISCLS 0                                '
VISCAMOUT 1                             '
WINDMAKE R,1,50,100,0,2                 '
FOR I1 = 0 TO 7                         '
    I2 = WINDREF(1,I1)                  'Ruft Daten von Fenster Nummer 1 ab.
    VISLOC 0,I1                         'Gibt die anzuzeigende Position an.
    VISPRINT “Data”;I1;“=”;I2           'Zeigt das Fenster an.
NEXT I1                                 '
WINDCLR 1                               '
VISLOC 0,9                              '
VISPRINT “Nach dem Löschen der Fensterinformationen”
                                        '
FOR I1 = 0 TO 7                         '
    I2 = WINDREF(1,I1)                  'Ruft Daten von Fenster Nummer 1 ab.
    VISLOC 0,10+I1                      'Gibt die anzuzeigende Position an.
    VISPRINT “Data”;I1;“=”;I2           'Zeigt das Fenster an.
NEXT I1                                 '
```

WINDCOPY (Anweisung)

Funktion

Kopiert Fensterdaten.

Format

WINDCOPY <Nummer des Kopierquellfensters>, <Nummer des Kopierzielfensters>

Erklärung

<Nummer des Kopierquellfensters>	Gibt die Nummer des Fensters an, aus dem Daten kopiert werden (0 bis 511).
<Nummer des Kopierzielfensters>	Gibt die Nummer des Fensters an, in das Daten kopiert werden (0 bis 511).

Anmerkung (1):	**Diese Anweisung ändert die anfängliche Einstellung nicht. Beim Neustarten des Systems mit ausgeschalteter Stromversorgung gehen die kopierten Einstellungen verloren.**
Anmerkung (2):	**Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.**

Verwandte Begriffe

WINDMAKE

Beispiel

```
VISSCREEN 1,0,1                                 '
VISCLS 0                                        '
VISCAMOUT 1                                     '
WINDCLR 2                                       '
WINDMAKE R,1,50,100,0,2                         '
VISLOC 0,0                                      '
VISPRINT “Informationen zum Kopierquellfenster” '
FOR I1 = 0 TO 7                                 '
    I2 = WINDREF(1,I1)                          'Ruft Daten von Fenster Nummer 1 ab.
    VISLOC 0,1+I1                               'Gibt die anzuzeigende Position an.
    VISPRINT “Data”;I1;“=”;I2                   'Zeigt das Fenster an.
NEXT I1                                         '
VISLOC 0,9                                      '
VISPRINT “Fensterinformationen vor dem Kopieren”'
FOR I1 = 0 TO 7                                 '
    I2 = WINDREF(2,I1)                          'Ruft Daten von Fenster Nummer 1 ab.
    VISLOC 0,10+I1                              'Gibt die anzuzeigende
                                                'Position an.
    VISPRINT “Data”;I1;“=”;I2                   'Zeigt das Fenster an.
NEXT I1                                         '
WINDCOPY 1,2                                    '
VISLOC 0,18                                     '
VISPRINT “Fensterinformationen nach dem Kopieren”'
FOR I1 = 0 TO 7                                 '
    I2 = WINDREF(2,I1)                          'Ruft Daten von Fenster Nummer 1 ab.
    VISLOC 0,19 + I1                            'Gibt die anzuzeigende Position an.
    VISPRINT “Data”;I1;“=”;I2                   'Zeigt das Fenster an.
NEXT I1                                         '
```

WINDREF (Funktion)

Funktion

Ruft Fensterinformationen ab.

Format

WINDREF (<Fensternummer>, <Element>)

Erklärung

Diese Anweisung gibt die Fensternummer mit <Fensternummer> (0 bis 511) an. Sie gibt die Nummer des <Element>s (0 bis 7) an, aus dem Daten abgerufen werden sollen.

Elementnummer 0: Existenz der Fenstereinstellung — Rückgabewert Vorhanden = 0, Nicht Vorhanden = -1
Elementnummer 1: Fensterform — Rückgabewert (Siehe nachfolgende Tabelle.)
Elementnummer 2: Fenster-Referenzpunkt X-Koordinaten
Elementnummer 3: Fenster-Referenzpunkt Y-Koordinaten
Elementnummer 4~9: Fenster-Einstellungsdaten — Rückgabewert für alle Einstellungsdaten (siehe folgende Tabelle.)

	Element						
Fensterform	1	4	5	6	7	8	9
Linie (2-Punkt-Angabe)	0	Startpunkt X-Koordinate	Startpunkt Y-Koordinate	Endpunkt X-Koordinate	Endpunkt Y-Koordinate	-1	-1
Linie (Längen- und Winkelangabe)	1	Länge	Winkel	-1	-1	-1	-1
Kreis	2	Radius	-1	-1	-1	-1	-1
Ellipse	3	Breite	Höhe	-1	-1	-1	-1
Sektor	4	Außendurchmesser	Innendurchmesser	Startwinkel	Endwinkel	Partitions-winkel	Modus
Rechteck	5	Breite	Höhe	Winkel	Modus	-1	-1

Anmerkung (1): Wenn keine Angabe gemacht wird, gilt der Standardwert -1 als Rückgabewert.
Anmerkung (2): Die abgerufenen Daten sind nicht die anfänglich eingestellten Daten, sondern die aktuell eingestellten Daten.
Anmerkung (3): Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.

Verwandte Begriffe

WINDMAKE

Beispiel

```
VISSCREEN 1,0,1                '
VISCLS 0                       '
VISCAMOUT 1                    '
WINDMAKE R,1,50,100,0,2        '
FOR I1 = 0 TO 7                '
    I2 = WINDREF(1,I1)         'Ruft Daten von Fenster Nummer 1 ab.
    VISLOC 0,I1                'Gibt die Anzeigeposition an.
    VISPRINT "Data";I1;"=";I2  'Zeigt das Fenster an.
NEXT I1                        '
```

WINDDISP (Anweisung)

Funktion

Zeichnet ein angegebenes Fenster.

Format

WINDDISP <Fensternummer>

Erklärung

Diese Anweisung gibt die Fensternummer mit <Fensternummer> (0 bis 511) an.

Anmerkung (1):	**Bei den Fensterinformationen gibt es einen Referenzpunkt zur Verarbeitung des Fensters. Diese Daten werden mit jeder Anweisung zur Bildverarbeitung festgelegt. Wenn Sie daher diese Anweisung nach der Ausführung von Prozeßanweisungen ausführen, wird das Fenster an einer richtigen Position angezeigt.**
Anmerkung (2):	**Gleich nach dem Festlegen eines neuen Fensters wird der Fenster-Referenzpunkt auf X = 0 und Y = 0 gesetzt.**
Anmerkung (3):	**Ein zu zeichnender Bildschirm wird zu dem mit VISSCREEN eingestellten.**
Anmerkung (4):	**Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.**

Verwandte Begriffe

WINDMAKE, VISSCREEN

Beispiel

```
VISOVERLAY 1                'Zeigt nur den Zeichungsbildschirm 0 auf dem
                            'Monitor an.
VISSCREEN 1,0,1             'Zeichnet sofort auf dem Zeichnungsbildschirm 0.
WINDMAKE R,1,200,200,0,2    'Erstellt ein Rechteckfenster.
WINDDISP 1                  'Zeigt das Fenster (mit dem Referenzpunkt
                            'X=0 and Y=0) an.
VISHIST 1,100,100           'Führt die Verarbeitung des Histogramms aus.
WINDDISP 1                  'Zeigt das Fenster (mit dem Referenzpunkt
                            'X=100 and Y=100) an.
```

21.4 Zeichnen

VISSCREEN (Anweisung)

Funktion

Gibt einen Zeichnungsbildschirm an.

Format

VISSCREEN <Zeichnungsobjekt>, <Bildschirmnummer>[, <Zeichnungsmodus>]

Erklärung

<Zeichnungsobjekt>	Gibt den Speicher bzw. den Overlay-Speicher (Bildschirm) an (0 oder 1). 0: Speicher (Prozeßfenster) 1: Overlay-Speicher (Nur Zeichnungsbildschirm)
<Bildschirmnummer>	Bildschirmnummer auf jedem Bildschirm Speicher (Prozeßfenster): 4 Bildschirme von 0 bis 3 Overlay-Speicher (Nur Zeichnungsbildschirm): 2 Bildschirme, 0 und 1.
<Zeichnungsmodus>	Gibt die Anzeige-Zeichnungsmethode an (0 oder 1). 0: Sofortanzeige OFF 1: Sofortanzeige ON Wenn diese Option nicht angegeben wird, gilt der Standardwert 1.

Anmerkung (1):	**Wenn Sie das System einschalten, werden VISSCREEN 1, 0, 1 als Anfangseinstellungen festgelegt.**
Anmerkung (2):	**Wenn die Option zum sofortigen Zeichnen ON ist und das System die Zeichnungsanweisung ausführt, wird das Ergebnis der Zeichnung auf dem Bildschirm angezeigt. Ist die Sofortanzeige jedoch OFF, wird der Bildschirm nach jeder Ausführung von Anweisungen entsprechend dem Zeichnungsobjekt von VISPLNOUT oder VISOVERLAY neu aufgebaut.**
Anmerkung (3):	**Wenn die Option zum sofortigen Zeichnen ON ist, braucht das System bei der Ausführung der Zeichenanweisung ca. 0,02 Sekunden zum Neuaufbau. Wenn Sie das Zeichnen häufig wiederholen, steigt die erforderliche Zeit daher rasch an. Durch Einstellen des sofortigen Zeichnens auf OFF und Anzeigen aller Objekte auf einmal können Sie Verarbeitungszeit sparen.**
Anmerkung (4):	**Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.**

Verwandte Begriffe

VISBRIGHT, VISCLS, VISPUTP, VISLINE, VISPTP, VISRECT, VISCIRCLE, VISELLIPSE, VISSECT, VISCROSS, VISLOC, VISDEFCHAR, VISPRINT

Beispiel

```
VISSCREEN 1,0,1              'Zeichnet sofort den Zeichnungsbildschirm 0.
VISCLS 0                     '
FOR I1 = 100 TO 200 STEP 2   '
    VISRECT I1,I1,200,200    'Zeichnet ein Rechteck mit einer Breite von 200 und einer
                             'Höhe von 200.
NEXT I1                      '
VISCLS 0                     'Löscht den Bildschirm.
VISSCREEN 1,0,0              'Zeichnet nur auf dem Zeichnungsbildschirm 0.
FOR I1 = 100 TO 200 STEP 2   '
    VISRECT I1,I1,200,200    'Zeichnet ein Rechteck mit einer Breite von 200 und einer
                             'Höhe von 200.
NEXT I1                      '
VISOVERLAY 1                 'Zeigt nur den Zeichungsbildschirm 0 auf dem Monitor an
                             '(Neuaufbau).
```

VISBRIGHT (Anweisung)

Funktion

Gibt einen Helligkeitswert für die Zeichnung an.

Format

VISBRIGHT <Helligkeitswert>

Erklärung

<Helligkeitswert> Gibt einen Helligkeitswert (0 bis 255) für die Zeichnung an.

Anmerkung (1):	**Wenn sich ein Zeichnungsobjekt im Speicher (Prozeßfenster) befindet, wird es mit der angegebenen Helligkeit gezeichnet. Befindet sich das Objekt im Overlay-Speicher (Nur-Zeichnen-Speicher), wird die folgende Helligkeit verwendet. Helligkeitswert 128 bis 255 : 255 (Weiß) Helligkeitswert 1 bis 127 : 0 (Schwarz) Helligkeitswert 0 : Durchsichtig (transparent)**
Anmerkung (2):	**Wenn Sie das System einschalten, wird VISBRIGHT 255 als Standardwert verwendet.**
Anmerkung (3):	**Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.**

Verwandte Begriffe

VISSCREEN, VISCLS, VISPUTP, VISLINE, VISPTP, VISRECT, VISCIRCLE, VISELLIPSE, VISSECT, VISCROSS, VISLOC, VISDEFCHAR, VISPRINT

Beispiel

```
VISPLNOUT 0                    'Zeigt Speicher 0 (Prozeßfenster) an.
VISSCREEN 0,0,1                'Zeichnet sofort auf das Prozeßfenster 0.
VISCLS 0                       '
FOR I1 = 10 TO 200 STEP 5      '
    VISBRIGHT I1               '
    VISRECT 100+I1,100,200,200 'Zeichnet ein Rechteck mit einer Breite von 200 und einer
                               'Höhe von 200.
NEXT I1                        '
DELAY 500                      'Stoppt 0,5 Sekunden lang.
VISCLS 0                       'Löscht den Bildschirm.
VISSCREEN 1,0,1                'Zeichnet sofort auf dem Zeichnungsbildschirm 0.
FOR I1 = 10 TO 200 STEP 5      '
    VISBRIGHT I1               '
    VISRECT 100+I1,100,200,200 'Zeichnet ein Rechteck mit einer Breite von 200 und einer
                               'Höhe von 200.
NEXT I1                        '
```

VISCLS (Anweisung)

Funktion

Füllt (bereinigt) eine angegebene Anzeige, die in einem Modus mit einer angegebenen Helligkeit definiert wurde.

Format

VISCLS [<Modus>[, <Helligkeitswert>]]

Erklärung

<Modus> (0~3) Wenn dieser Wert nicht angegeben ist, wird 2 verwendet.

0: Füllt den angegebenen Bildschirm mit der VISSCREEN-Anweisung mit der angegebenen Helligkeit.
1: Füllt das gesamte Prozeßfenster mit der angegebenen Helligkeit.
2: Füllt den gesamten Bildschirm mit der angegebenen Helligkeit.
3: Füllt alle Bildschirme mit der angegebenen Helligkeit.

<Helligkeitswert> Gibt den Helligkeitswert für die Beschichtung (Löschen) an. (0 bis 255). Wenn diese Option nicht angegeben wird, gilt der Standardwert 0.

Anmerkung (1):	**Wenn sich ein Zeichnungsobjekt im Speicher (Prozeßfenster) befindet, wird es mit der angegebenen Helligkeit gelöscht. Befindet sich das Objekt im Overlay-Speicher (Nur-Zeichnen-Speicher), wird die folgende Helligkeit verwendet.** **Helligkeitswert 128 bis 255 : 255 (Weiß)** **Helligkeitswert 1 bis 127 : 0 (Schwarz)** **Helligkeitswert 0 : Durchsichtig (transparent)**
Anmerkung (2):	**Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.**

Verwandte Begriffe

VISSCREEN

Beispiel

```
VISSCREEN 1,0,1              '
VISCLS 0                     '
VISSCREEN 1,1,1              '
VISCLS 0                     '
VISPLNOUT 0                  'Zeigt Speicher 0 (Prozeßfenster) an.
VISSCREEN 0,0,1              'Zeichnet sofort auf das Prozeßfenster 0.
FOR I1 = 10 TO 200 STEP 5    '
    VISCLS 0,I1              'Füllt den Bildschirm mit der angegebenen Helligkeit von I1.
NEXT I1                      '
DELAY 500                    'Stoppt 0,5 Sekunden lang.
VISCLS 0                     'Füllt den Bildschirm mit der angegebenen Helligkeit 0.
VISRECT 200,200,200,200      'Zeichnet ein Rechteck mit einer Breite von 200 und einer
                             'Höhe von 200.
DELAY 500                    'Stoppt 0,5 Sekunden lang.
VISCLS 0                     'Füllt ein Rechteck auf dem Zeichnungsbildschirm.
VISCAMOUT 1                  '
VISSCREEN 1,0,1              'Zeichnet sofort auf dem Zeichnungsbildschirm 0.
VISRECT 100,100,200,200      'Zeichnet ein Rechteck mit einer Breite von 200 und einer
                             'Höhe von 200.
DELAY 500                    'Stoppt 0,5 Sekunden lang.
VISCLS 0                     'Füllt ein Rechteck auf dem Zeichnungsbildschirm.
VISCLS 0,127                 'Füllt den Zeichnungsbildschirm schwarz aus.
DELAY 500                    'Stoppt 0,5 Sekunden lang.
VISCLS 0,255                 'Füllt den Zeichnungsbildschirm weiß aus.
DELAY 500                    'Stoppt 0,5 Sekunden lang.
VISCLS 0                     'Setzt den Zeichnungsbildschirm auf transparent.
```

VISPUTP (Anweisung)

Funktion

Zeichnet einen Punkt auf dem Bildschirm.

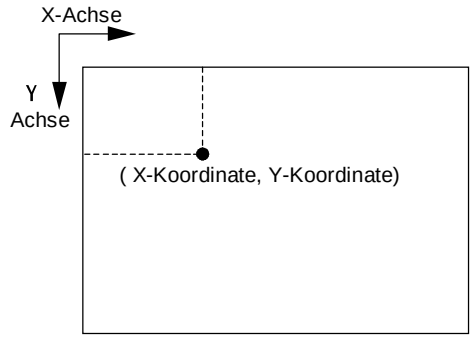
Format

VISPUTP <X-Koordinate>, <Y-Koordinate>

Erklärung

<X-Koordinate> Gibt die X-Koordinate zum Zeichnen eines Punktes an.
<Y-Koordinate> Gibt die Y-Koordinate zum Zeichnen eines Punktes an.

Anmerkung (1):	**Die Werte der X- und Y-Koordinaten werden nicht geprüft. Auch wenn die Koordinaten außerhalb des zulässigen Bereiches liegen, tritt kein Fehler auf.**
Anmerkung (2):	**Der Objektbildschirm ist der mit VISSCREEN angegebene Bildschirm.**
Anmerkung (3):	**Der Helligkeitswert für die Zeichnung ist die mit VISBRIGHT angegebene Helligkeit.**
Anmerkung (4):	**Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.**



Verwandte Begriffe

VISSCREEN, VISBRIGHT

Beispiel

```
VISPLNOUT 0                  'Zeigt Speicher 0 (Prozeßfenster) an.
VISSCREEN 0,0,1              'Zeichnet sofort auf das Prozeßfenster 0.
VISCLS 0                     'Löscht den Bildschirm.
VISBRIGHT 255                'Stellt den Helligkeitswert für die Zeichnung ein.
FOR I1 = 10 TO 200 STEP 5    '
    VISPUTP 100+I1,100       'Zeichnet einen Punkt an den Koordinaten
                             '(100+I1,100).
NEXT I1                      '
```

VISLINE (Anweisung)

Funktion

Zeichnet eine Linie auf dem Bildschirm.

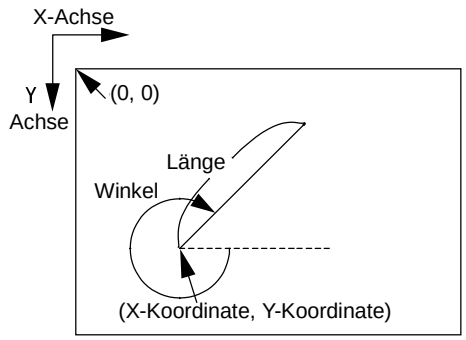
Format

VISLINE<X-Koordinate>, <Y-Koordinate>, <Länge> [, <Winkel>]

Erklärung

<X-Koordinate> Gibt die X-Koordinate zum Zeichnen einer Linie an.
<Y-Koordinate> Gibt die Y-Koordinate zum Zeichnen einer Linie an.
<Länge> Gibt die Länge der zu zeichnenden Linie an.
<Winkel> Gibt den Winkel der zu zeichnenden Linie an. Wenn diese Option nicht angegeben wird, gilt der Standardwert 0 Grad.

Anmerkung (1): Die Werte der X- und Y-Koordinaten, der Länge und der Winkel werden nicht geprüft. Auch wenn diese Werte außerhalb des zulässigen Bereiches liegen, tritt kein Fehler auf.
Anmerkung (2): Der Objektbildschirm ist der mit VISSCREEN angegebene Bildschirm.
Anmerkung (3): Der Helligkeitswert für die Zeichnung ist die gleiche Helligkeit, die mit VISBRIGHT angegeben wurde.
Anmerkung (4): Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.



Verwandte Begriffe

VISSCREEN, VISBRIGHT

Beispiel

```
VISPLNOUT 0                    'Zeigt Speicher 0 (Prozeßfenster) an.
VISSCREEN 0,0,1                'Zeichnet sofort auf das Prozeßfenster 0.
VISCLS 0                       'Löscht den Bildschirm.
VISBRIGHT 255                  'Stellt den Helligkeitswert für die
                               'Zeichnung ein.
FOR I1 = 0 TO 360 STEP 5       '
    VISLINE 256,256,100,I1     'Zeichnet eine Linie mit der Länge 100.
NEXT I1                        '
```

VISPTP (Anweisung)

Funktion

Zeichnet eine Linie als Verbindung zwischen zwei Punkten auf dem Bildschirm.

Format

VISPTP <Startpunkt X-Koordinate>, <Startpunkt Y-Koordinate>, <Endpunkt X-Koordinate>, <Endpunkt Y-Koordinate>

Erklärung

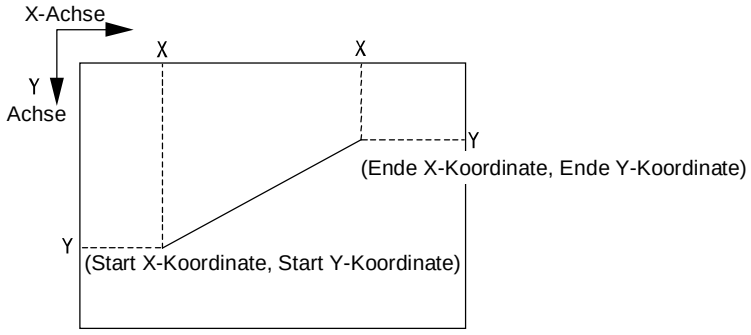
<Startpunkt X-Koordinate>	Gibt die X-Koordinate des Startpunktes der zu zeichnenden Linie an.
<Startpunkt Y-Koordinate>	Gibt die Y-Koordinate des Startpunktes der zu zeichnenden Linie an.
<Endpunkt X-Koordinate>	Gibt die X-Koordinate des Endpunktes der zu zeichnenden Linie an.
<Endpunkt Y-Koordinate>	Gibt die Y-Koordinate des Endpunktes der zu zeichnenden Linie an.

Anmerkung (1): Die Werte der X- und Y-Koordinaten werden nicht geprüft. Auch wenn diese Werte außerhalb des zulässigen Bereiches liegen, tritt kein Fehler auf.

Anmerkung (2): Der Objektbildschirm ist der mit VISSCREEN angegebene Bildschirm.

Anmerkung (3): Der Helligkeitswert für die Zeichnung ist die gleiche Helligkeit, die mit VISBRIGHT angegeben wurde.

Anmerkung (4): Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.



Verwandte Begriffe

VISSCREEN, VISBRIGHT

Beispiel

```
VISPLNOUT 0                  'Zeigt Speicher 0 (Prozeßfenster) an.
VISSCREEN 0,0,1              'Zeichnet sofort auf das Prozeßfenster 0.
VISCLS 0                     'Löscht den Bildschirm.
VISBRIGHT 255                'Stellt den Helligkeitswert für die Zeichnung
                             'ein.
FOR I1 = 200 TO 300          '
    VISPTP 100,100,200,I1    'Zeichnet eine Linie, die zwei Punkte verbindet.
NEXT I1                      '
```

VISRECT (Anweisung)

Funktion

Zeichnet ein Rechteck auf dem Bildschirm.

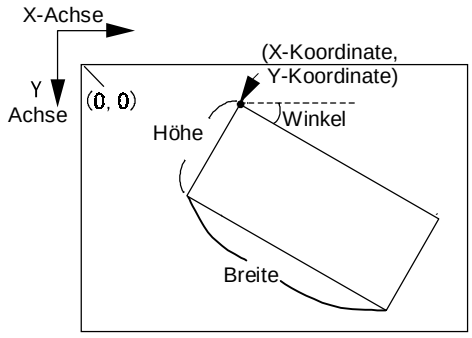
Format

VISRECT <X-Koordinate>, <Y-Koordinate>, <Breite>, <Höhe>[, <Modus>[, <Winkel>]]

Erklärung

<X-Koordinate> Gibt die X-Koordinate des zu zeichnenden Rechteckes an.
<Y-Koordinate> Gibt die Y-Koordinate des zu zeichnenden Rechteckes an.
<Breite> Gibt die Breite des zu zeichnenden Rechteckes an.
<Höhe> Gibt die Höhe des zu zeichnenden Rechteckes an.
<Modus> Gibt den Modus für das Zeichnen an.
0: Zeichnet nur einen Umriß des Rechteckes.
1: Zeichnet einen vollständig beschichteten Bildschirm.
Wenn diese Option nicht angegeben wird, gilt der Standardwert 0.
<Winkel> Gibt den Winkel der zu zeichnenden Linie an. Wenn diese Option nicht angegeben wird, gilt der Standardwert 0 Grad.

Anmerkung (1): Die Werte der X- und Y-Koordinaten, der Länge und der Winkel werden nicht geprüft. Auch wenn die Koordinaten außerhalb des zulässigen Bereiches liegen, tritt kein Fehler auf.
Anmerkung (2): Der Objektbildschirm ist der mit VISSCREEN angegebene Bildschirm.
Anmerkung (3): Der Helligkeitswert für die Zeichnung ist die gleiche Helligkeit, die mit VISBRIGHT angegeben wurde.
Anmerkung (4): Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.



Verwandte Begriffe

VISSCREEN, VISBRIGHT

Beispiel

```
VISPLNOUT 0                          'Zeigt Speicher 0 (Prozeßfenster) an.
VISSCREEN 0,0,1                      'Zeichnet sofort auf das Prozeßfenster 0.
VISCLS 0                             'Löscht den Bildschirm.
VISBRIGHT 255                        'Stellt den Helligkeitswert für die Zeichnung ein.
FOR I1 = 0 TO 360 STEP 5             '
    VISRECT 256,256,100,100,0,I1     'Zeichnet ein Rechteck mit einer Breite von 100 und einer Höhe
'von 100.
NEXT I1                              '
VISRECT 200,200,50,50,1,45           'Zeichnet ein beschichtetes Rechteck mit einer Höhe von 50 und
'einer Breite von 50.
```

VISCIRCLE (Anweisung)

Funktion

Zeichnet einen Kreis auf dem Bildschirm.

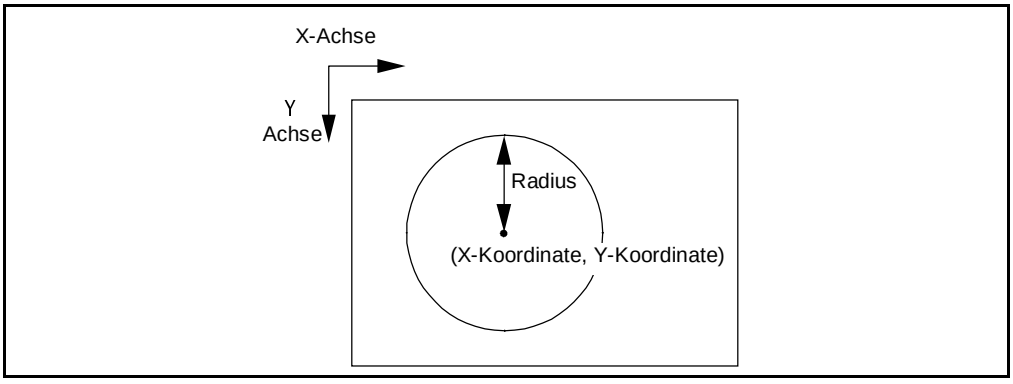
Format

VISCIRCLE <X-Koordinate>, <Y-Koordinate>, <Radius> [, <Modus>]

Erklärung

<X-Koordinate> Gibt die X-Koordinate des zu zeichnenden Kreises an.
<Y-Koordinate> Gibt die Y-Koordinate des zu zeichnenden Kreises an.
<Radius> Gibt den Radius des zu zeichnenden Kreises an.
<Modus> Gibt den Modus für das Zeichnen an.
0: Zeichnet nur einen Umriß eines Kreises
1: Zeichnet einen vollständig ausgefüllten Kreis.
Wenn diese Option nicht angegeben wird, gilt der Standardwert 0.

Anmerkung (1):	**Die Werte der X- und Y-Koordinaten sowie der Radius werden nicht geprüft. Auch wenn diese Werte außerhalb des zulässigen Bereiches liegen, tritt kein Fehler auf.**
Anmerkung (2):	**Der Objektbildschirm ist der mit VISSCREEN angegebene Bildschirm.**
Anmerkung (3):	**Der Helligkeitswert für die Zeichnung ist die gleiche Helligkeit, die mit VISBRIGHT angegeben wurde.**
Anmerkung (4):	**Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.**



Verwandte Begriffe

VISSCREEN, VISBRIGHT

Beispiel

```
VISPLNOUT 0             'Zeigt Speicher 0 (Prozeßfenster) an.
VISSCREEN 0,0,1         'Zeichnet sofort auf das Prozeßfenster 0.
VISCLS 0                'Löscht den Bildschirm.
VISBRIGHT 255           'Stellt den Helligkeitswert für die Zeichnung ein.
VISCIRCLE 255,255,100   'Zeichnet einen Kreis mit dem Radius 100.
VISCIRCLE 200,200,50,1  'Zeichnet einen ausgefüllten Kreis mit dem Radius 50.
```

VISELLIPSE (Anweisung)

Funktion

Zeichnet eine Ellipse auf dem Bildschirm.

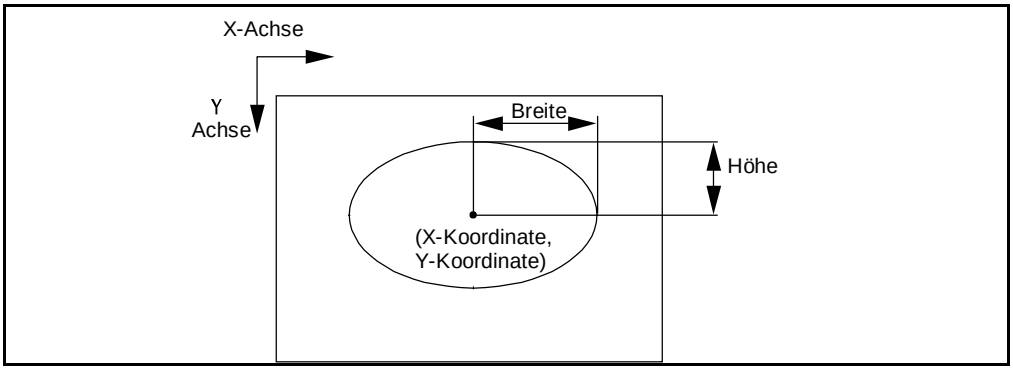
Format

VISELLIPSE <X-Koordinate>, <Y-Koordinate>, <Breite>, <Höhe> [, <Modus>]

Erklärung

<X-Koordinate> Gibt die X-Koordinate der zu zeichnenden Ellipse an.
<Y-Koordinate> Gibt die Y-Koordinate der zu zeichnenden Ellipse an.
<Breite> Gibt die Breite der zu zeichnenden Ellipse an.
<Höhe> Gibt die Höhe der zu zeichnenden Ellipse an.
<Modus> Gibt den Modus für das Zeichnen an.
0: Zeichnet nur einen Umriß einer Ellipse.
1: Zeichnet eine vollständig ausgefüllte Ellipse.
Wenn diese Option nicht angegeben wird, gilt der Standardwert 0.

Anmerkung (1): Die Werte der X- und Y-Koordinaten sowie der Radius werden nicht geprüft. Auch wenn diese Werte außerhalb des zulässigen Bereiches liegen, tritt kein Fehler auf.
Anmerkung (2): Der Objektbildschirm ist der mit VISSCREEN angegebene Bildschirm.
Anmerkung (3): Der Helligkeitswert für die Zeichnung ist die gleiche Helligkeit, die mit VISBRIGHT angegeben wurde.
Anmerkung (4): Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.



Verwandte Begriffe

VISSCREEN, VISBRIGHT

Beispiel

```
VISPLNOUT 0                    'Zeigt Speicher 0 (Prozeßfenster) an.
VISSCREEN 0,0,1                'Zeichnet sofort auf das Prozeßfenster 0.
VISCLS 0                       'Löscht den Bildschirm.
VISBRIGHT 255                  'Stellt den Helligkeitswert für die Zeichnung ein.
VISELLIPSE 255,255,100,50      'Zeichnet eine Ellipse mit einer Breite von 50 und einer
                               'Höhe von 100.
VISELLIPSE 255,255,50,100,1    'Zeichnet eine beschichtete Ellipse mit einer Breite von 50
                               'und einer Höhe von 100.
```

VISSECT (Anweisung)

Funktion

Zeichnet einen Sektor auf dem Bildschirm.

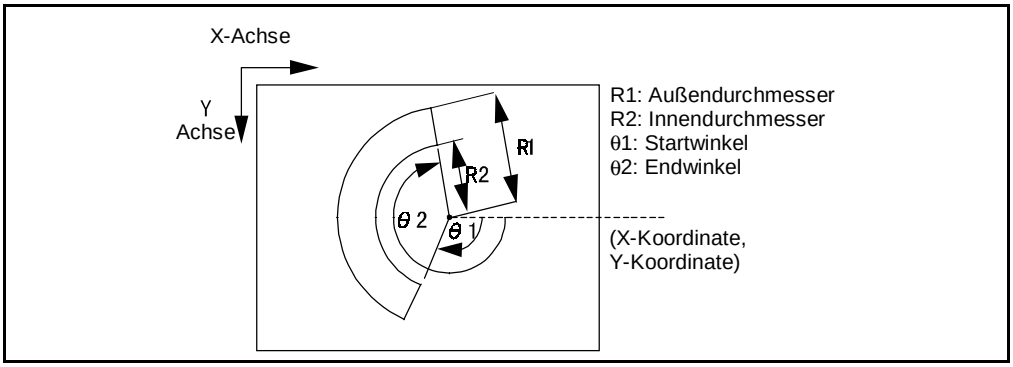
Format

VISSECT <X-Koordinate>, <Y-Koordinate>, <Außendurchmesser>, <Innendurchmesser>, <Startwinkel>, <Endwinkel>

Erklärung

<X-Koordinate> Gibt die X-Koordinate des zu zeichnenden Sektors an.
<Y-Koordinate> Gibt die Y-Koordinate des zu zeichnenden Sektors an.
<Außendurchmesser> Gibt den Außendurchmesser des zu zeichnenden Sektors an.
<Innendurchmesser> Gibt den Innendurchmesser des zu zeichnenden Sektors an.
<Startwinkel> Gibt den Startwinkel des zu zeichnenden Sektors an.
<Endwinkel> Gibt den Endwinkel des zu zeichnenden Sektors an.

Anmerkung (1): Die angegebenen Werte werden nicht geprüft. Auch wenn diese Werte außerhalb des zulässigen Bereiches liegen, tritt kein Fehler auf.
Anmerkung (2): Der Objektbildschirm ist der mit VISSCREEN angegebene Bildschirm.
Anmerkung (3): Der Helligkeitswert für die Zeichnung ist die gleiche Helligkeit, die mit VISBRIGHT angegeben wurde.
Anmerkung (4): Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.



Verwandte Begriffe

VISSCREEN, VISBRIGHT

Beispiel

```
VISPLNOUT 0                    'Zeigt Speicher 0 (Prozeßfenster) an.
VISSCREEN 0,0,1                'Zeichnet sofort auf das Prozeßfenster 0.
VISCLS 0                       'Löscht den Bildschirm.
VISBRIGHT 255                  'Stellt den Helligkeitswert für die Zeichnung ein.
VISSECT 255,255,100,50,100,260 'Zeichnet einen Sektor.
```

VISCROSS (Anweisung)

Funktion

Zeichnet ein Kreuzsymbol auf dem Bildschirm.

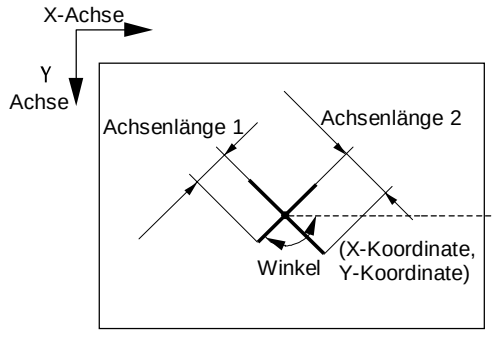
Format

VISCROSS <X-Koordinate>, <Y-Koordinate>[, <Achsenlänge 1>[, <Achsenlänge 2> [, <Winkel>]]]

Erklärung

<X-Koordinate> Gibt die X-Koordinate des zu zeichnenden Kreuzsymbols an.
<Y-Koordinate> Gibt die Y-Koordinate des zu zeichnenden Kreuzsymbols an.
<Achsenlänge 1> Gibt die Achsenlänge des zu zeichnenden Kreuzsymbols an. Wenn diese Option nicht angegeben wird, gilt der Standardwert 10.
<Achsenlänge 2> Gibt die Achsenlänge des zu zeichnenden Kreuzsymbols an. Wenn diese Option nicht angegeben wird, gilt der Standardwert 10.
<Winkel> Gibt den Winkel des zu zeichnenden Kreuzsymbols an. Wenn diese Option nicht angegeben wird, gilt der Standardwert 0 Grad.

Anmerkung (1): Die angegebenen Werte werden nicht geprüft. Auch wenn diese Werte außerhalb des zulässigen Bereiches liegen, tritt kein Fehler auf.
Anmerkung (2): Der Objektbildschirm ist der mit VISSCREEN angegebene Bildschirm.
Anmerkung (3): Der Helligkeitswert für die Zeichnung ist die gleiche Helligkeit, die mit VISBRIGHT angegeben wurde.
Anmerkung (4): Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.



Verwandte Begriffe

VISSCREEN, VISBRIGHT

Beispiel

```
VISPLNOUT 0                    'Zeigt Speicher 0 (Prozeßfenster) an.
VISSCREEN 0,0,1                'Zeichnet sofort auf das Prozeßfenster 0.
VISCLS 0                       'Löscht den Bildschirm.
VISBRIGHT 255                  'Stellt den Helligkeitswert für die Zeichnung ein.
VISCROSS 255,255,10,20,45      'Zeichnet ein Kreuzsymbol.
```

VISLOC (Anweisung)

Funktion

Kennzeichnet die Anzeigeposition der Zeichen.

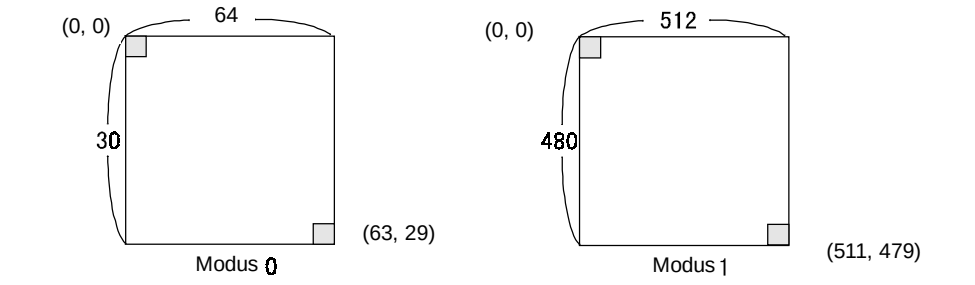
Format

VISLOC <X-Position>, <Y-Position>[, <Modus>]

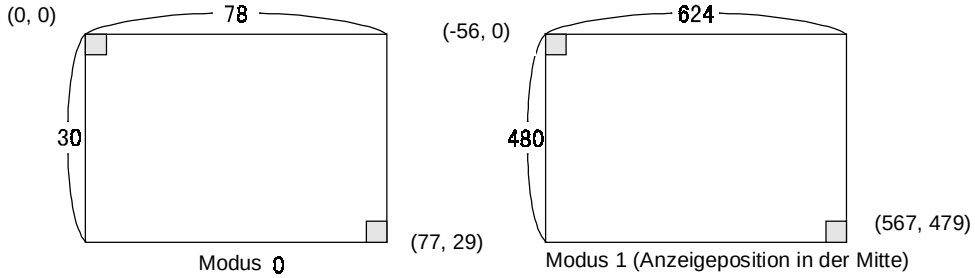
Erklärung

<X-Koordinate> Gibt die X-Koordinate des zu zeichnenden Zeichens an.
<Y-Koordinate> Gibt die Y-Koordinate des zu zeichnenden Zeichens an.
<Modus> Gibt den Koordinatenmodus der X- und Y-Positionsdaten an.
Wenn diese Option nicht angegeben wird, gilt der Standardwert 0.
0: Angabe mit Spalte und Linie.
1: Angabe mit dem XY-Koordinatensystem.
Die anzugebenden Werte werden nicht geprüft. Auch wenn diese Werte außerhalb des Zeichnungsbereiches liegen, tritt kein Fehler auf. Wenn sie außerhalb des zulässigen Bereiches liegen, wird die Anweisung ungültig, und es wird der zuvor eingestellte Wert verwendet.

Anmerkung (1): Wenn Sie das System einschalten, wird VISLOC 0, 0, 0 als Anfangseinstellung verwendet.
Anmerkung (2): Wenn Sie Modus 0 einstellen, merkt sich das System nach dem Zeichnen die Position der Zeichen.
Anmerkung (3): Wenn die Zeichen auf der rechten Seite über das Ende der Zeile hinausgehen, fügt das System automatisch weitere Zeilen ein.
Anmerkung (4): Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.



Wenn der Verarbeitungsbildschirm ein Objekt ist.



Wenn der Zeichnungsbildschirm ein Objekt ist.

Verwandte Begriffe

VISPRINT

Beispiel

```
VISPLNOUT 0                    'Zeigt Speicher 0 (Prozeßfenster) an.
VISSCREEN 0,0,1                'Zeichnet sofort auf das Prozeßfenster 0.
VISCLS 0                       'Löscht den Bildschirm.
VISBRIGHT 255                  'Stellt den Helligkeitswert für die Zeichnung ein.
VISLOC 10,10                   'Gibt die Position zum Anzeigen von Zeichen an.
VISPRINT “Testmodus 0”         '
VISLOC 10,10,1                 'Gibt die Position zum Anzeigen von Zeichen an.
VISPRINT “Testmodus 1”         '
VISPRINT “Testmodus 1 Fortsetzung”
                               '
VISPRINT “******** Zeilenvorschub ********”
```

VISDEFCHAR (Anweisung)

Funktion

Gibt die Größe der Zeichen und die Anzeigemethode an.

Format

VISDEFCHAR <Breitenangabe>, <Längenangabe>, <Anzeigemethode>

Erklärung

<Breitenangabe> Gibt die Zeichenbreite an (1 bis 4).
<Längenangabe> Gibt die Länge der Zeichen an (1 bis 4).
<Anzeigemethode> Gibt die Anzeigemethode für die Zeichen an (0 bis 3).
0: Weiß
1: Schwarz
2: Umrißzeichen auf einem weißen Hintergrund
3: Umrißzeichen auf einem schwarzen Hintergrund

Anmerkung (1): Wenn Sie das System einschalten, wird VISDEFCHAR 1, 1, 0 als Anfangseinstellung verwendet.

Anmerkung (2): Die Standardgröße für Zeichen ist 16 (Länge) x 16 (Breite) Punkte (VISDEFCHAR 1, 1, *).

VISDEFCHAR 1,1,1

VISDEFCHAR 2,2,1

Anmerkung (3): Wenn Sie weiß oder schwarz setzen, werden nur Zeichen angezeigt, und der Hintergrund bleibt.

Anmerkung (4): Für Zeichen auf einem schwarzen Hintergrund wird der Hintergrund auf schwarz gesetzt und die Zeichen auf weiß. Für Zeichen auf einem weißen Hintergrund wird der Hintergrund auf weiß gesetzt und die Zeichen auf schwarz. Schwer lesbare Zeichen sind somit besser zu erkennen.

Anmerkung (5): Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.

Verwandte Begriffe

VISPRINT

Beispiel

```
VISPLNOUT 0            'Zeigt Speicher 0 (Prozeßfenster) an.
VISSCREEN 0,0,1        'Zeichnet sofort auf das Prozeßfenster 0.
VISCLS 128             'Löscht den Bildschirm.
VISLOC 10,10           'Gibt die Anzeigeposition an.
VISDEFCHAR 1,1,0       'Gibt die Zeichengröße und die Anzeigemethode an.
VISPRINT “Größe 1”     '
VISLOC 10,11           'Gibt die Anzeigeposition an.
VISDEFCHAR 2,2,1       'Gibt die Zeichengröße und die Anzeigemethode an.
VISPRINT “Größe 2”     '
VISLOC 10,13           'Gibt die Anzeigeposition an.
VISDEFCHAR 2,2,2       'Gibt die Zeichengröße und die Anzeigemethode an.
VISPRINT “Umrißzeichen auf einem weißen Hintergrund”
VISDEFCHAR 2,2,3       'Gibt die Zeichengröße und die Anzeigemethode an.
VISPRINT “Umrißzeichen auf einem schwarzen Hintergrund”
```

VISPRINT (Anweisung)

Funktion

Zeigt Zeichen und Abbildungen auf dem Bildschirm an.

Format

VISPRINT <Meldung>[<Trennzeichen><Meldung> …]

Erklärung

<Meldung> Gibt Zeichen, Variablen und Konstanten innerhalb von “” an.

<Trennzeichen> Verwendet ein Komma (,) oder Semikolon (;).
Komma: Erstellt Leerzeichen zwischen Meldungen.
Semikolon: Keine Leerzeichen zwischen Meldungen.
Wenn Sie ein Komma oder Semikolon am Ende von <Meldung> angeben, wird kein Zeilenvorschub generiert, und die nächste VISPRINT-Anzeige beginnt in dieser Zeile.

Verwandte Begriffe

VISSCREEN, VISBRIGHT, VISLOC, VISDEFCHAR

Beispiel

```
VISPLNOUT 0                    'Zeigt Speicher 0 (Prozeßfenster) an.
VISSCREEN 0,0,1                'Zeichnet sofort auf das Prozeßfenster 0.
VISCLS 0                       'Löscht den Bildschirm.
VISLOC 10,10                   'Gibt die Anzeigeposition an.
VISDEFCHAR 1,1,0               'Gibt die Zeichengröße und die Anzeigemethode an.
I1= 10                         '
F1 = 0.999                     '
VISPRINT “Ganzahl-Variable I1=”;I1,“Real-Variable F1=”;F1
```

21.5 Bildverarbeitung

VISWORKPLN (Anweisung)

Funktion

Gibt den zu verarbeitenden Speicher (Prozeßfenster) an.

Format

VISWORKPLN <Speichernummer>

Erklärung

<Speichernummer> Gibt die Nummer des zu verarbeitenden Speichers (0 bis 3) an. Wird diese Option nicht angegeben, gilt der Standardwert 0.

Anmerkung (1): Wenn Sie das System einschalten, wird VISWORKPLN 0 als Anfangseinstellung verwendet.
Anmerkung (2): Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.

Verwandte Begriffe

VISGETP, VISHIST, VISBINA, VISMEASURE, VISPROJ, VISEDGE, VISREADQR, BLOB, SHDEFMODEL, SHMODEL, SHCORNER, SHCIRCLE

Beispiel

```
VISWORKPLN 0            'Gibt das Verarbeitungsobjekt für Speicher 0 an.
```

VISGETP (Funktion)

Funktion

Ruft eine angegebene Koordinatenhelligkeit aus dem Speicher ab (Prozeßfenster).

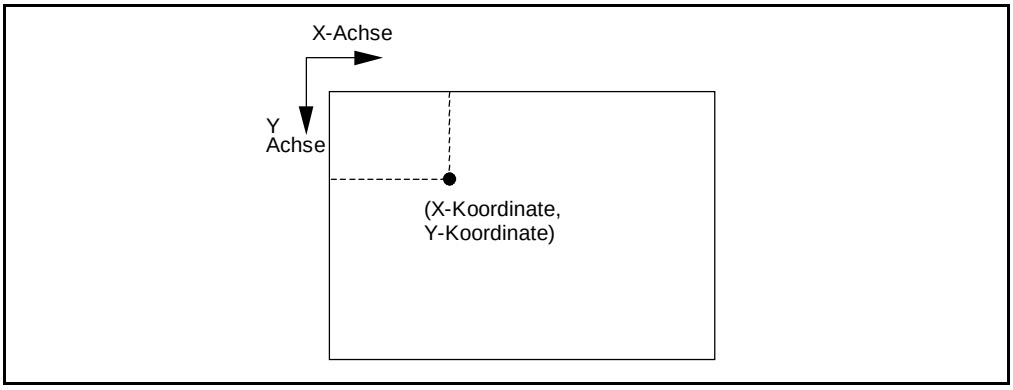
Format

VISGETP (<Koordinate X>, <Koordinate Y>)

Erklärung

<Koordinate X> Gibt die X-Koordinate (0 bis 511) an.
<Koordinate Y> Gibt die Y-Koordinate (0 bis 479) an.

Anmerkung (1):	**Das Verarbeitungsobjekt ist der mit VISWORKPLN angegebene Bildschirm.**
Anmerkung (2):	**Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.**



Verwandte Begriffe

VISWORKPLN

Beispiel

```
VISPLNOUT 0                      'Zeigt Speicher 0 (Prozeßfenster) an.
FOR I1 = 10 TO 200 STEP 5        '
  I2 = VISGET  P(100+I1,100)     'Ordnet den Helligkeitswert der Koordinaten
                                 '(100+I1, 100) der Variablen I2 zu.
  VISLOC 10,10                   'Gibt die Anzeigeposition an.
  VISDEFCHAR 1,1,3               'Gibt die Größe der Anzeigezeichen und den Anzeigemodus an.
  VISPRINT "Helligkeitswert =";I2
                                 '
NEXT I1
```

VISHIST (Anweisung)

Funktion

Ruft das Histogramm (die Helligkeitsverteilung) des Bildschirmes ab.

Format

VISHIST <Fensternummer>, <Koordinate X>, <Koordinate Y>

Erklärung

<Fensternummer> Gibt die Fensternummer (0 bis 511) an.
<Koordinate X> Gibt die X-Koordinate (0 bis 511) an.
<Koordinate Y> Gibt die Y-Koordinate (0 bis 479) an.

Anmerkung (1): Gibt den Verarbeitungsbereich mit einem Fenster an.
Anmerkung (2): Wenn die angegebene Fensterposition außerhalb des Bildschirmes liegt, führt die Ausführung zu einem Fehler.
Anmerkung (3): Die einzige Fensterform, die Sie angeben können, ist ein Rechteck mit 0 Grad. Wenn eine andere Fensterform angegeben wird, tritt ein Fehler auf.
Anmerkung (4): Das Verarbeitungsobjekt ist der mit VISWORKPLN angegebene Bildschirm.
Anmerkung (5): Das Verarbeitungsergebnis kann mit der Anweisung VISREFHIST gelesen werden.
Anmerkung (6): Das Verarbeitungsergebnis wird im Speicher abgelegt. Es bleibt im Speicher, bis diese Anweisung erneut ausgeführt wird.
Anmerkung (7): Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.

Verwandte Begriffe

WINDMAKE, VISWORKPLN, VISREFHIST

Beispiel

```
WINDMAKE R,1,100,100,0,2     'Stellt Fenster 1 auf Rechteck ein.
CAMIN 1                      'Ruft ein Kamerabild aus dem Speicher ab.
VISWORKPLN 0                 'Stellt den Verarbeitungsobjektbildschirm für
                             'Speicher 0 ein.
VISHIST 1,100,100            'Führt das Histogramm für Fenster 1 mit den Koordinaten
                             '(100, 100) als Ausgangsposition aus.
FOR I1 = 0 TO 255            '
    I2 = VISREFHIST(I1)      'Weist die Verteilungsdaten von Helligkeitswert I1
                             'der Variablen I2 zu.
    VISLOC 10, 10            'Gibt die Anzeigeposition an.
    VISDEFCHAR 1,1,3         'Gibt die Zeichengröße und den Anzeigemodus an.
    VISPRINT "Verteilungsdaten=";I2
                             '
NEXT I1
```

VISREFHIST (Funktion)

Funktion

Liest Ergebnisse des Histogrammes.

Format

VISREFHIST (<Helligkeitswert>)

Erklärung

<Helligkeitswert> Gibt den Helligkeitswert des zu lesenden Histogrammes an (0 bis 255).

Verwandte Begriffe

VISHIST

Beispiel

```
WINDMAKE R,1,100,100,0,2     'Stellt Fenster 1 auf Rechteck ein.
CAMIN 1                      'Ruft ein Kamerabild aus dem Speicher ab.
VISWORKPLN 0                 'Stellt den Verarbeitungsobjektbildschirm für
                             'Speicher 0 ein.
VISHIST 1,100,100            'Führt das Histogramm für Fenster 1 mit den Koordinaten
                             '(100, 100) als Ausgangsposition aus.
FOR I1 = 0 TO 255            '
    I2 = VISREFHIST(I1)      'Weist die Verteilungsdaten von Helligkeitswert I1
                             'der Variablen I2 zu.
    VISLOC 10, 10            'Gibt die Anzeigeposition an.
    VISDEFCHAR 1,1,3         'Gibt die Zeichengröße und den Anzeigemodus an.
    VISPRINT “Verteilungsdaten=”;I2
                             '
NEXT I1
```

VISLEVEL (Funktion)

Funktion

Ruft eine Binärisierungsstufe entsprechend dem Ergebnis des Histogrammes ab.

Format

VISLEVEL (<Modus>[, <Referenzbereich>[, <Verarbeitungsobjekt>]])

Erklärung

<Modus> Gibt die Methode zum Abrufen der Binärisierungsstufe an (0 bis 2).
0: Modusmethode
1: Unterscheidungsanalysemethode
2: P-Kachelungsmethode
Ausführliche Hinweise, siehe Teil 1, Abschitt 5.1.1.5 “Binärisierung” in Kapitel 5 dieses Handbuches.

<Referenzbereich> Gibt den Wert des Bereiches an, wenn das System die Binärisierungssystufe über die P-Kachelungsmethode (1 bis 245760) abruft. Wenn diese Option nicht angegeben wird, gilt der Standardwert 1.

<Verarbeitungsobjekt> Gibt das Objekt (weiß und schwarz) an, wenn das System die Binärisierungsstufe über die P-Kachelungsmethode (0 oder 1) abruft. Wenn diese Option nicht angegeben wird, gilt der Standardwert 1.
0: Schwarz (akkumuliert aus Helligkeitswert 0)
1: Weiß (akkumuliert aus Helligkeitswert 255)

Anmerkung (1):	**<Referenzbereich> und <Verarbeitungsobjekt> werden in der Modusmethode und der Unterscheidungsanalysemethode nicht verwendet.**
Anmerkung (2):	**Vor der Ausführung dieser Funktion sollten Sie VISHIST ausführen.**
Anmerkung (3):	**Wenn die angegebenen Daten nicht korrekt sind oder der Prozeß einen Fehler ergibt, lautet der Rückgabewert –1.**
Anmerkung (4):	**Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.**

Verwandte Begriffe

VISHIST

Beispiel

```
VISCLS 0                        '
WINDMAKE R,1,100,100,0,2        'Stellt Fenster 1 auf Rechteck ein.
CAMIN 1                         'Ruft ein Kamerabild aus dem Speicher ab.
VISWORKPLN 0                    'Stellt den Verarbeitungsobjektbildschirm für
                                'Speicher 0 ein.
VISHIST 1,100,100               'Führt das Histogramm in Fenster 1 mit den
                                'Koordinaten (100, 100) als Ausgangsposition aus.
WINDDISP 1                      '
I2 = VISLEVEL(0)                'Ruft die Binärisierungsstufe mit der
                                'Modusmethode auf.
VISLOC 10,10                    '
IF I2 = -1 THEN VISPRINT “Fehler” ELSE VISPRINT “Binärisierungsstufe=”;I2
                                '
I2 = VISLEVEL(1)                'Ruft die Binärisierungsstufe mit der
                                'Unterscheidungsanalysemethode auf.
VISLOC 10,11                    '
IF I2 = -1 THEN VISPRINT “Fehler” ELSE VISPRINT “Binärisierungsstufe=”;I2
                                '
I2 = VISLEVEL(2,100,0)          'Ruft die Binärisierungsstufe über die
                                'P-Kachelungsmethode ab.
VISLOC 10,12                    '
IF I2 = -1 THEN VISPRINT “Fehler” ELSE VISPRINT “Binärisierungsstufe=”;I2
                                '
```

VISBINA (Anweisung)

Funktion

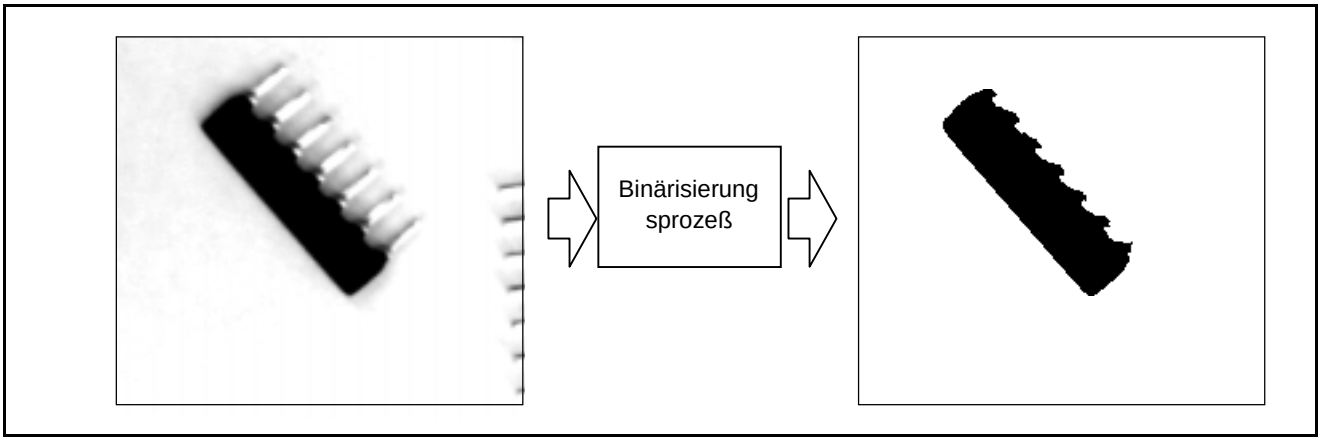
Binärisiert die Anzeige.

Format

VISBINA <Fensternummer>, <Koordinate X>, <Koordinate Y>, <Binäre Untergrenze>[, <Binäre Obergrenze>]

Erklärung

<Fensternummer> Gibt die Fensternummer (0 bis 511) an.
<Koordinate X> Gibt die X-Koordinate (0 bis 511) an.
<Koordinate Y> Gibt die Y-Koordinate (0 bis 479) an.
<Binäre Untergrenze> Gibt die Untergrenze für die Binärisierung an (0 bis 254 Untergrenze < Obergrenze).
<Binäre Obergrenze> Gibt die Obergrenze für die Binärisierung an (1 bis 255 Untergrenze < Obergrenze). Wenn diese Option nicht angegeben wird, gilt der Standardwert 255.



Beispiel zur Binärisierung

Anmerkung (1):	**Gibt den Verarbeitungsbereich mit einem Fenster an.**
Anmerkung (2):	**Wenn die angegebene Fensterposition außerhalb des Bildschirmes liegt, führt die Ausführung zu einem Fehler.**
Anmerkung (3):	**Die einzige Fensterform, die Sie angeben können, ist ein Rechteck mit 0 Grad. Wenn eine andere Fensterform angegeben wird, tritt ein Fehler auf.**
Anmerkung (4):	**Das Verarbeitungsobjekt ist der mit VISWORKPLN angegebene Bildschirm.**
Anmerkung (5):	**Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.**

Verwandte Begriffe

WIMDMAKE, VISWORKPLN

Beispiel

```
VISSCREEN 1,0,1                 'Zeichnet sofort auf das Prozeßfenster 0.
WINDMAKE R,1,100,100,0,2        'Stellt Fenster 1 auf Rechteck ein.
CAMIN 1                         'Ruft ein Kamerabild aus dem Speicher ab.
VISWORKPLN 0                    'Stellt den Verarbeitungsobjektbildschirm auf
                                'Speicher 0 ein.
VISPLNOUT 0                     'Zeigt Speicher 0 auf dem Bildschirm an.
VISBINA 1,100,100,128,255       'Binärisiert im Fenster.
WINDDISP 1                      'Zeichnet das Fenster.
```

VISBINAR (Anweisung)

Funktion

Zeigt einen binärisierten Bildschirm an.

Format

VISBINAR <Modus>[, <Binäre Untergrenze>[, Binäre Obergrenze]]

Erklärung

<Modus> Gibt den Modus für die Anzeige des binärisierten Bildschirmes (0 oder 1) an.
- 0: Verläßt die Binärisierungsanzeige und kehrt zurück zur ursprünglichen Anzeige.
- 1: Führt die Binärisierungsanzeige aus.

<Binäre Untergrenze> Gibt die Untergrenze für die Binärisierung (0 bis 254 Untergrenze < Obergrenze) an.

<Binäre Obergrenze> Gibt die Obergrenze für die Binärisierung (1 bis 255 Untergrenze < Obergrenze) an.
Wenn diese Option nicht angegeben wird, gilt der Standardwert 255.

Anmerkung (1):	**Der gesamte auf dem aktuellen Monitor angezeigte Bildschirm wird binärisiert.**
Anmerkung (2):	**Der Inhalt des Speichers ändert sich nicht, auch wenn der Speicher (Prozeßfenster) ausgewählt und angezeigt wird.**
Anmerkung (3):	**Die Binärisierungsanzeige wird fortgesetzt, bis der Modus auf 0 gesetzt wird.**
Anmerkung (4):	**Zur Ausführung dieser Anweisung ist eine μBildverarbeitungskarte (Option) erforderlich.**

Verwandte Begriffe

VISBINA

Beispiel

```
VISBINAR 1,128,255      'Startet die Binärisierungsanzeige.
VISCAMOUT 1             'Zeigt Kamera 1 auf dem Monitor an.
DELAY 5000              'Stoppt 5 Sekunden lang.
VISPLNOUT 0             'Zeigt Speicher 0 auf dem Bildschirm an.
DELAY 5000              'Stoppt 5 Sekunden lang.
VISBINAR 0              'Kehrt zurück zur vorigen Anzeige.
```

VISFILTER (Anweisung)

Funktion

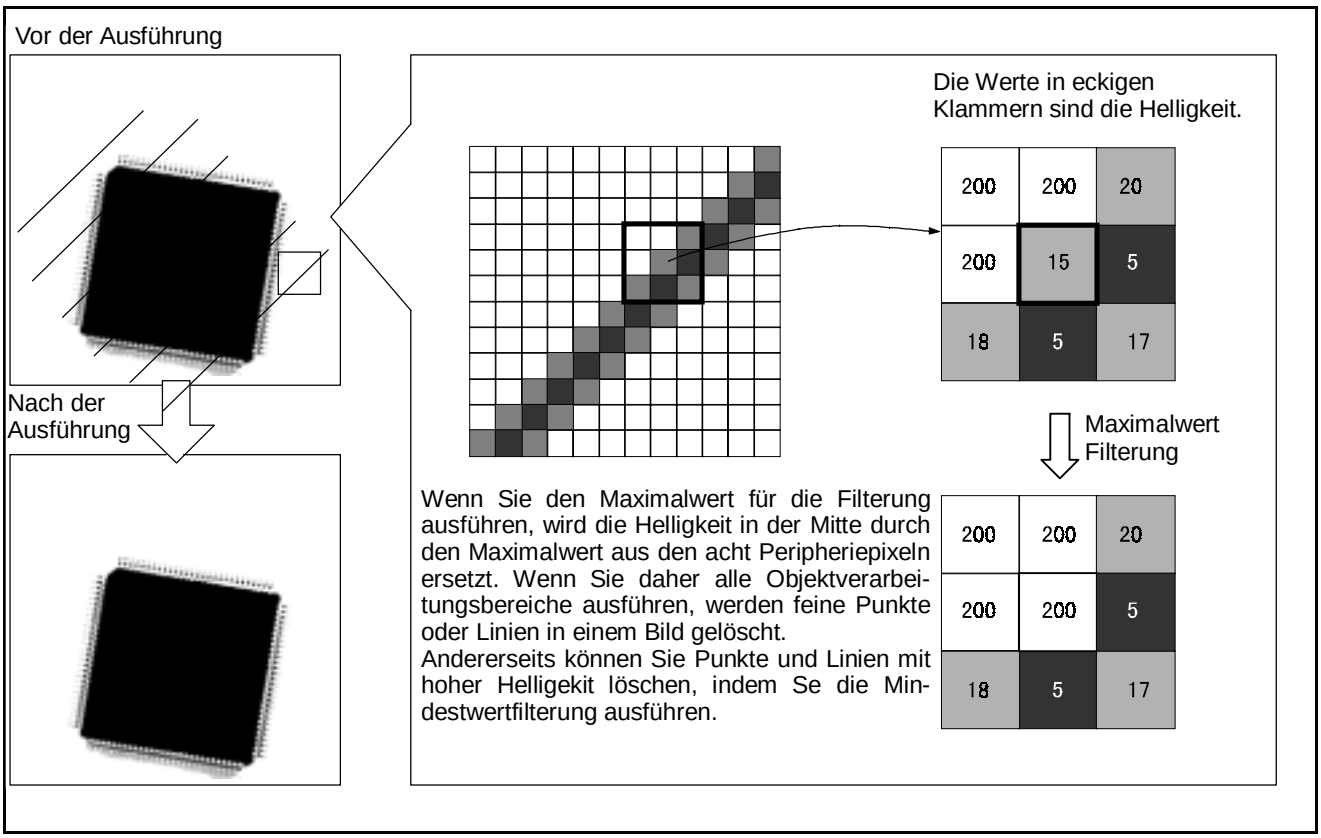
Führt eine Filterung auf der Anzeige aus.

Format

VISFILTER <Fensternummer>, <Koordinate X>, <Koordinate Y>, <Prozeß-fenster>, <Speicherbildschirm>[, <Modus>]

Erklärung

<Fensternummer> Gibt die Fensternummer (0 bis 511) an.
<Koordinate X> Gibt die X-Koordinate (0 bis 511) an.
<Koordinate Y> Gibt die Y-Koordinate (0 bis 479) an.
<Prozeßfenster> Gibt die Nummer des Speichers (0 bis 3) an.
<Speicherbildschirm> Gibt die Nummer des Speichers an, in dem die Filterergebnisse abgelegt werden sollen (0~3).
<Modus> Gibt den Typ der Flterung (3 x 3 Abstandsfilterung) an.
0: Mindestwert Filterung
1: Maximalwert Filterung
Wenn diese Option nicht angegeben wird, gilt der Standardwert 0.



Anmerkung (1):	**Wenn das Prozeßfenster und der Speicherbildschirm die gleiche Nummer haben, tritt ein Fehler auf.**
Anmerkung (2):	**Gibt den Verarbeitungsbereich mit einem Fenster an.**
Anmerkung (3):	**Wenn die angegebene Fensterposition außerhalb des Bildschirmes liegt, führt die Ausführung zu einem Fehler.**
Anmerkung (4):	**Die einzige Fensterform, die Sie angeben können, ist ein Rechteck mit 0 Grad. Wenn eine andere Fensterform angegeben wird, tritt ein Fehler auf.**
Anmerkung (5):	**Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.**

Verwandte Begriffe

WINDMAKE

Beispiel

```
VISSCREEN 1,0,1                'Zeichnet sofort auf das Prozeßfenster 0.
WINDMAKE R,1,100,100,0,2       'Stellt Fenster 1 auf Rechteck ein.
CAMIN 1                        'Ruft ein Kamerabild aus dem Speicher ab.
VISPLNOUT 1                    'Zeigt Speicher 1 auf dem Bildschirm an.
VISFILTER 1,100,100,0,1,0      '
VISFILTER 1,100,100,1,0,0      '
VISFILTER 1,100,100,0,1,0      '
VISFILTER 1,100,100,1,0,0      '
VISFILTER 1,100,100,0,1,0      '
WINDDISP 1                     'Zeichnet das Fenster.
```

VISMASK (Anweisung)

Funktion

Führt Berechnungen zwischen Abbildern aus.

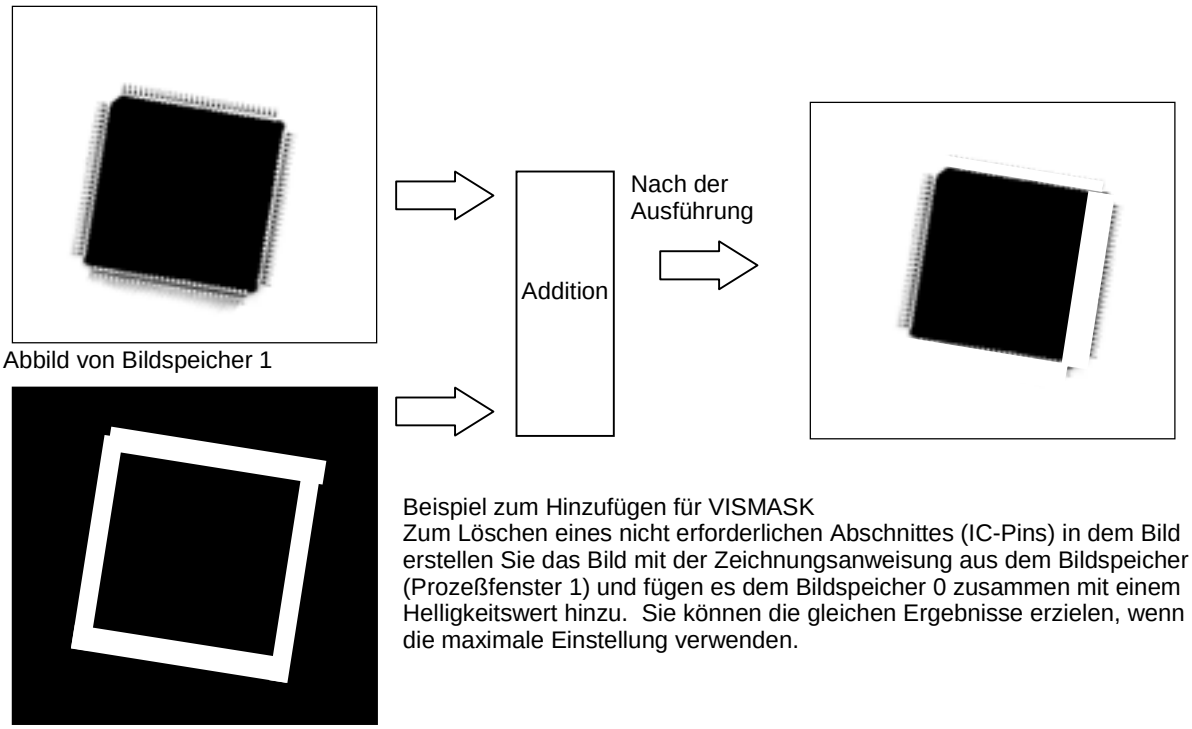
Format

VISMASK <Fensternummer>, <Koordinate X>, <Koordinate Y>,
<Anzeige 1>, <Anzeige 2>,
<Modus> [, <Binäre Untergrenze>[, <Binäre Obergrenze>]]

Erklärung

<Fensternummer> Gibt die Fensternummer (0 bis 511) an.
<Koordinate X> Gibt die X-Koordinate (0 bis 511) an.
<Koordinate Y> Gibt die Y-Koordinate (0 bis 479) an.
<Bildschirm 1> Gibt die Nummer des Speichers für die Berechnung an (0 bis 3).
<Bildschirm 2> Gibt die Nummer des Speichers für die Berechnung an. Dies ist das Speicherziel für das Verarbeitungsergebnis (0 bis 3).
<Modus> Gibt die Art der Berechnung zwischen Bildern an (0 bis 10).

0: Binäres AND
1: Binäres OR
2: Binäres XOR
3: AND (Führt für jedes Bit eine AND-Verknüpfung aus, nachdem der Helligkeitswert binärisiert wurde.)
4: OR (Führt für jedes Bit eine ODER-Verknüpfung aus, nachdem der Helligkeitswert binärisiert wurde.)
5: XOR (Führt für jedes Bit eine XOR-Verknüpfung aus, nachdem der Helligkeitswert binärisiert wurde.)
6: Addition (Ergibt 255, wenn der Helligkeitswert 255 oder größer ist.)
7: Subtraktion (Ergibt 0, wenn der Helligkeitswert 0 oder kleiner ist.)
8: Maximaler Wert (Wählt einen größeren Helligkeitswert aus)
9: Minimaler Wert (Wählt einen kleineren Helligkeitswert aus)
10: Absoluter Wert (der absolute Wert für den Unterschied zwischen den Helligkeitswerten.)



Beispiel zum Hinzufügen für VISMASK
Zum Löschen eines nicht erforderlichen Abschnittes (IC-Pins) in dem Bild erstellen Sie das Bild mit der Zeichnungsanweisung aus dem Bildspeicher 1 (Prozeßfenster 1) und fügen es dem Bildspeicher 0 zusammen mit einem Helligkeitswert hinzu. Sie können die gleichen Ergebnisse erzielen, wenn Sie die maximale Einstellung verwenden.

Anmerkung (1):	**Wenn das Prozeßfenster und der Speicherbildschirm die gleiche Nummer haben, tritt ein Fehler auf.**
Anmerkung (2):	**Gibt den Verarbeitungsbereich mit einem Fenster an.**
Anmerkung (3):	**Wenn die angegebene Fensterposition außerhalb des Bildschirmes liegt, führt die Ausführung zu einem Fehler.**
Anmerkung (4):	**Die einzige Fensterform, die Sie angeben können, ist ein Rechteck mit 0 Grad. Wenn eine andere Fensterform angegeben wird, tritt ein Fehler auf.**
Anmerkung (5):	**Bei der Berechnung der Binärisierung wird auch Bildschirm 1 binärisiert.**
Anmerkung (6):	**Zur Ausführung dieser Anweisung ist eine μBildverarbeitungskarte (Option) erforderlich.**

Verwandte Begriffe

VISBINA

Beispiel

```
VISSCREEN 0,0,1         'Zeichnet sofort auf Speicher 0.
WINDMAKE R,1,512,480,0,2
                        'Stellt Fenster 1 auf Rechteck ein.
CAMIN 1                 'Ruft ein Kamerabild aus dem Speicher ab.
VISCOPY 0,1             'Kopiert Speicher 0 in 1.
VISBRIGHT 255           'Stellt die Helligkeit der Zeichnung auf 255 ein.
VISRECT 100,100,100,100,1
                        'Zeichnet ein ausgefülltes Rechteck auf dem Bildschirm.
VISPLNOUT 1             'Zeigt Speicher 1 auf dem Bildschirm an.
VISMASK 1,0,0,0,1,10
                        'Berechnet den absoluten Wert des Unterschieds zwischen den
                        'beiden Bildschirmen.
WINDDISP 1              'Zeichnet das Fenster.
```

VISCOPY (Anweisung)

Funktion

Kopiert die Anzeige.

Format

VISCOPY <Kopierquellbildschirm>, <Kopierzielbildschirm>

Erklärung

<Kopierquellbildschirm>	Gibt die Nummer des Speichers der Kopierquelle an (0 bis 3)
<Kopierzielbildschirm>	Gibt die Nummer des Speichers des Kopierziels an (0 bis 3)

Anmerkung (1):	**Wenn die Nummern für Kopierquelle und Kopierziel identisch sind, tritt ein Fehler auf.**
Anmerkung (2):	**Wenn die Nummer des Kopierziels auf dem Bildschirm angezeigt wurde, folgt das System der Angabe des VISSCREEN-Zeichnungsmodus.**
Anmerkung (3):	**Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.**

Verwandte Begriffe

VISSCREEN

Beispiel

```
VISSCREEN 1,0,1        'Zeichnet sofort auf Zeichnungsbildschirm 0.
CAMIN 1                'Ruft ein Kamerabild aus dem Speicher ab.
VISCOPY 0,1            'Kopiert Speicher 0 in 1.
VISPLNOUT 1            'Zeigt Speicher 1 auf dem Bildschirm an.
```

VISMEASURE (Anweisung)

Funktion

Mißt Merkmale in dem Fenster (Bereich, Schwerpunkt, Winkel der Hauptachse).

Format

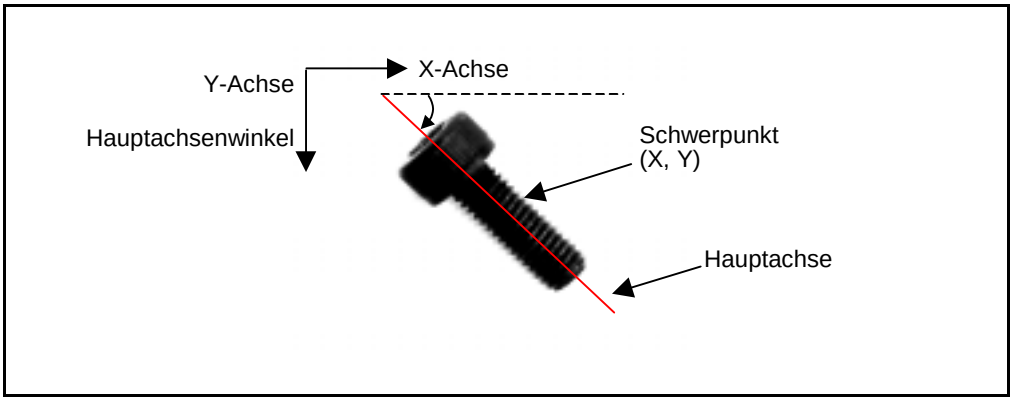
VISMEASURE <Fensternummer>, <Koordinate X>, <Koordinate Y>, <Verarbeitungsobjekt>, <Modus>, <Binäre Untergrenze>[, <Binäre Obergrenze>]

Erklärung

<Fensternummer> Gibt die Fensternummer (0 bis 511) an.
<Koordinate X> Gibt die X-Koordinate (0 bis 511) an.
<Koordinate Y> Gibt die Y-Koordinate (0 bis 479) an.
<Verarbeitungsobjekt> Gibt das zu messende Objekt an (0 oder 1).
- 0: Schwarz (Helligkeitswert < binäre Untergrenze, binäre Obergrenze < Helligkeitswert)
- 1: Weiß (binäre Untergrenze ≤ Helligkeitswert ≤ binäre Obergrenze)

<Modus> Gibt das zu messende Merkmal an.
- 0: Akkumulierte Helligkeit, Bereich
- 1: Akkumulierte Helligkeit, Bereich, Schwerpunkt (primäres Moment)
- 2: Akkumulierte Helligkeit, Bereich, Schwerpunkt (primäres Moment), Hauptachsenwinkel (sekundäres Moment)

Anmerkung (1): Je mehr Merkmale vorliegen, desto länger dauert der Vorgang.
Anmerkung (2): Bei der Ausführung wird das Prozeßfenster 3 (VISWORKPLN 3) als Arbeitsbereich verwendet, und die Daten auf dem Prozeßfenster 3 sind nicht garantiert. Außerdem können Sie das Prozeßfenster 3 nicht für die Verarbeitung verwenden.
Anmerkung (3): Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.



<Binäre Untergrenze> Gibt die Untergrenze für die Binärisierung (0 bis 254 Untergrenze < Obergrenze) an.
<Binäre Obergrenze> Gibt die Obergrenze für die Binärisierung (1 to 255 Untergrenze < Obergrenze) an.
Wenn diese Option nicht angegeben wird, gilt der Standardwert 255.

Anmerkung (1): Gibt den Verarbeitungsbereich mit einem Fenster an. (Das Sektorfenster allein kann den Hauptachsenwinkel nicht messen.)

		Modus		
	○: verfügbar ×: nicht verfügbar	0	1	2
Linie (2-Punkt-Angabe)	Windmake P	○	○	○
Linie (Länge- und Winkel)	Windmake L	○	○	○
Kreis	Windmake C	○	○	○
Ellipse	Windmake E	○	○	○
Sektor	Windmake S	○	○	×
Rechteck	Windmake R	○	○	○

Anmerkung (2): Wenn die angegebene Fensterposition außerhalb des Bildschirmes liegt, führt die Ausführung zu einem Fehler.

Anmerkung (3): Die folgenden Daten können mit der Funktion zum Abrufen des Verarbeitungsergebnisses abgerufen werden.

	VISSTATUS (n)
n	Element
0	Ausführungsergebnis 0= normal -1= abnormal
1	Unbekannt
2	Ausführungszeit

	VISGETNUM (a, b)
b	a=0
0	Fläche
1	Schwerpunkt X-Koordinatenwert
2	Schwerpunkt Y-Koordinatenwert
3	Hauptachsenwinkel
4	Akkumulierte Helligkeit
5	Primäres Moment X
6	Primäres Moment Y
7	Sekundäres Moment X
8	Sekundäres Moment Y
9	Sekundäres Moment XY

Anmerkung (4): Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.

Verwandte Begriffe

WINDMAKE, VISWORKPLN, VISGETNUM, VISSTATUS

Beispiel

```
VISSCREEN 0,0,1                 'Zeichnet sofort auf Speicher 0.
WINDMAKE R,1,512,480,0,2        'Stellt Fenster 1 auf Rechteck ein.
CAMIN 1                         'Ruft ein Kamerabild aus dem Speicher ab.
VISWORKPLN 0                    'Gibt das Objekt aus Speicher 0 an.
VISPLNOUT 0                     '
VISMEASURE 1,0,0,1,2,128        'Ruft die akkumulierte Helligkeit, Bereich,
                                'Schwerpunkt und Hauptachsenwinkel ab.
IF VISSTATUS(0) = 0.0 THEN      '
    FOR I1 = 0 TO 9             '
        VISLOC 0,I1             '
        VISPRINT "Ergebnis";I1;"=";VISGETNUM(0,I1)
                                '
    NEXT I1                     '
END IF                          '
```

VISPROJ (Anweisung)

Funktion

Mißt die projizierten Daten in dem Fenster.

Format

VISPROJ <Fensternummer>, <Koordinate X>, <Koordinate Y>, <Verarbeitungsobjekt>, <Binäre Untergrenze>[, <Binäre Obergrenze>]

Erklärung

<Fensternummer> Gibt die Fensternummer (0 bis 511) an.
<Koordinate X> Gibt die X-Koordinate (0 bis 511) an.
<Koordinate Y> Gibt die Y-Koordinate (0 bis 479) an.
<Verarbeitungsobjekt> Gibt das zu messende Objekt an (0 oder 1).
0: Schwarz (Helligkeitswert < binäre Untergrenze, binäre Obergrenze < Helligkeitswert)
1: Weiß (binäre Untergrenze ≤ Helligkeitswert ≤ binäre Obergrenze)
<Binäre Untergrenze> Gibt die Untergrenze für die Binärisierung (0 bis 254 Untergrenze < Obergrenze) an.
<Binäre Obergrenze> Gibt die Obergrenze für die Binärisierung (1 bis 255 Untergrenze < Obergrenze) an.
Wenn diese Option nicht angegeben wird, gilt der Standardwert 255.

Anmerkung (1): Gibt den Verarbeitungsbereich mit einem Fenster an.

○: Verfügbar ×: Nicht verfügbar

Linie (2-Punkt-Angabe)	Windmake P	○
Linie (Länge- und Winkel)	Windmake L	○
Kreis	Windmake C	×
Ellipse	Windmake E	×
Sektor	Windmake S	○
Rechteck	Windmake R	○

Anmerkung (2): Wenn die angegebene Fensterposition außerhalb des Bildschirmes liegt, führt die Ausführung zu einem Fehler.

Anmerkung (3): Die folgenden Daten können mit der Funktion zum Abrufen des Verarbeitungsergebnisses abgerufen werden.

	VISSTATUS (n)
n	Element
0	Ausführungsergebnis 0= normal -1= Abnormal
1	Unbekannt
2	Ausführungszeit

	VISGETNUM (a, b)
b	a = 0~511
0	Fläche
1	Unbekannt
2	Unbekannt
3	Unbekannt
4	Helligkeitsintegration
5	Unbekannt
6	Unbekannt
7	Unbekannt
8	Unbekannt
9	Unbekannt

Anmerkung (4): Bei der Ausführung wird das Prozeßfenster 3 (VISWORKPLN 3) als Arbeitsbereich verwendet, und die Daten auf dem Prozeßfenster 3 sind nicht garantiert. Außerdem können Sie das Prozeßfenster 3 nicht für die Verarbeitung verwenden.

Anmerkung (5): Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.

Verwandte Begriffe

WINDMAKE, VISWORKPLN, VISGETNUM, VISSTATUS

Beispiel

```
VISSCREEN 0,0,1                 'Zeichnet sofort auf Speicher 0.
WINDMAKE R,1,100,20,0,1         'Stellt Fenster 1 auf Rechteck ein.
CAMIN 1                         'Ruft ein Kamerabild aus dem Speicher ab.
VISWORKPLN 0                    'Gibt ein Objekt aus Speicher 0 an.
VISPROJ 1,100,100,1,128         'Mißt die Projektdaten.
IF VISSTATUS(0) = 0.0 THEN      '
    FOR I1 = 0 TO 19            '
      VISPRINT VISGETNUM(I1,0)  '
    NEXT I1                     '
END IF                          '
```

VISEDGE (Anweisung)

Funktion

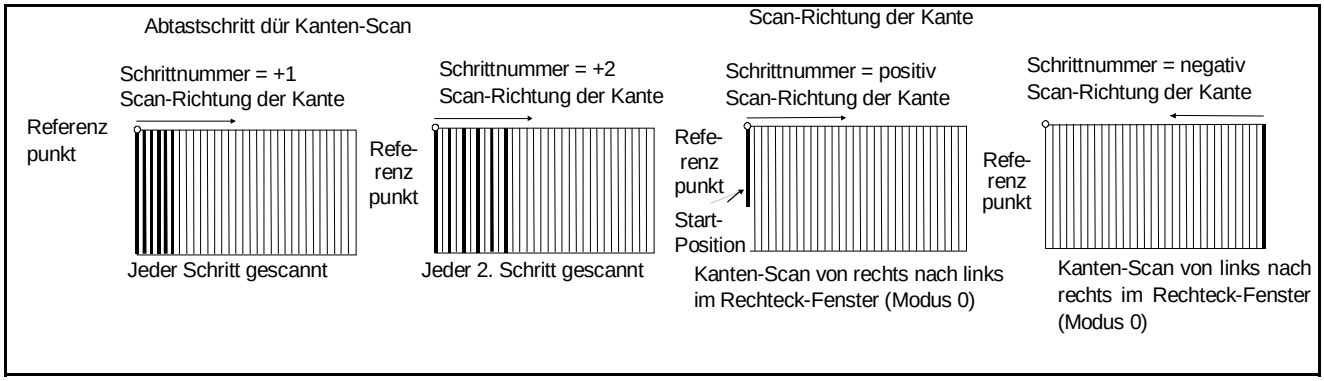
Mißt die Kante in einem Fenster.

Format

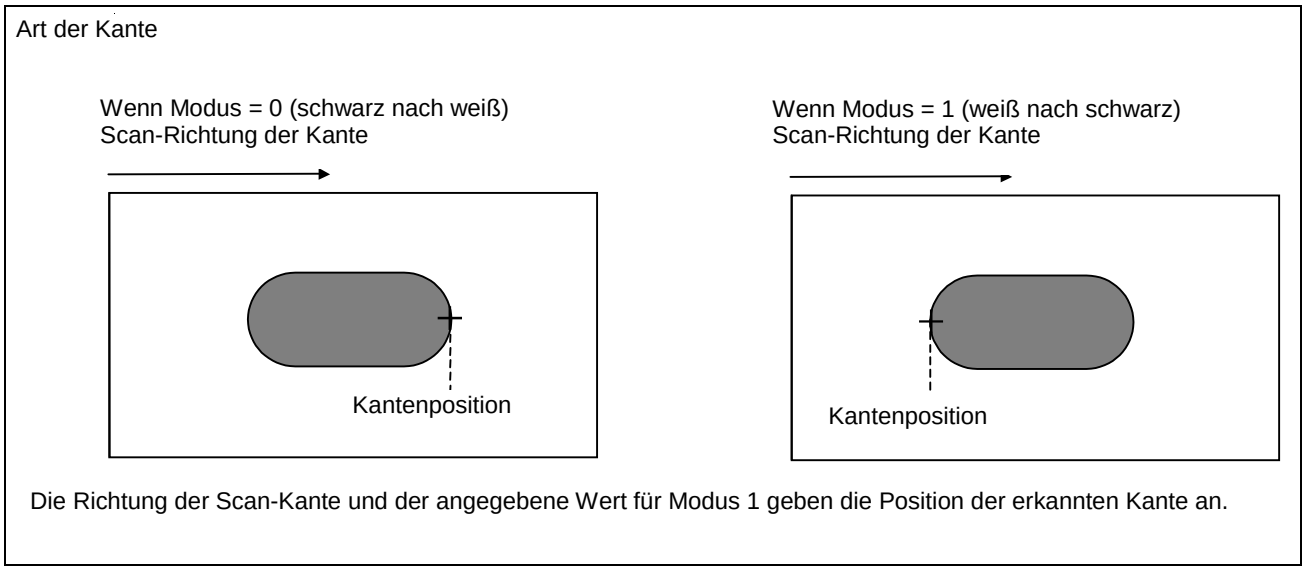
VISEDGE<Fensternummer>, <Koordinate X>, <Koordinate Y>, <Schritt>, <Verarbeitungsobjekt>, <Stufe>[, <Modus>[, <Binäre Untergrenze>[, <Binäre Obergrenze>]]]

Erklärung

<Fensternummer> Gibt die Fensternummer (0 bis 511) an.
<Koordinate X> Gibt die X-Koordinate (0 bis 511) an.
<Koordinate Y> Gibt die Y-Koordinate (0 bis 479) an.
<Schritt> Gibt die Scan-Richtung und die Abtastrate an (-511 to 511).



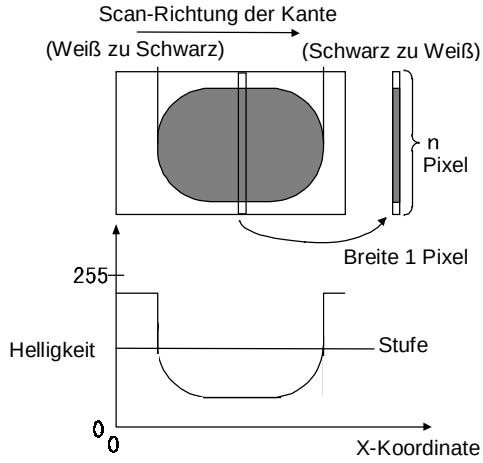
<Verarbeitungsobjekt> Gibt die zu messenden Objekte an (0 bis 2)
0: Umkehrpunkt von schwarz nach weiß (Kante)
1: Umkehrpunkt von weiß nach schwarz
2: Umkehrpunkt (beide Kanten 0 und 1)



Die Richtung der Scan-Kante und der angegebene Wert für Modus 1 geben die Position der erkannten Kante an.

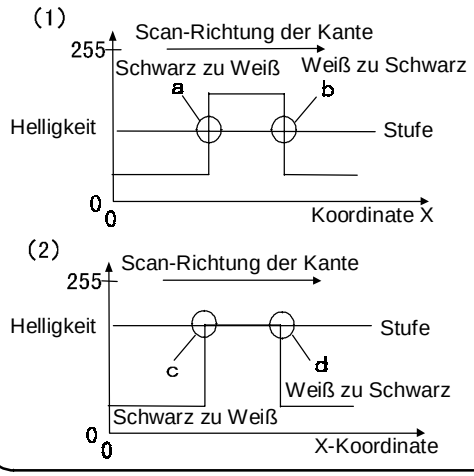
<Stufe> Gibt die Stufe für die Erkennung der Kante an (0 bis 512).
<Modus> Gibt den Modus für die Erkennung der Kante an. Wenn diese Option nicht angegeben wird, gilt der Standardwert 0.
0: Absoluter Wert der durchschnittlichen Helligkeit
1: Unterschiedswert der durchschnittlichen Helligkeit
2: Absoluter Wert des Bereiches
3: Unterschiedswert des Bereiches

Durchschnittlicher Helligkeitswert



In diesem Beispiel wird die Kante von links nach rechts in einem Rechteckfenster erkannt. Die Grafik zeigt, wie sich die Helligkeit mit dem Helligkeitsdurchschnitt ändert.
Das verwendete Fenster ist in Einheiten von 1 Pixel in der Richtung der Scan-Kante unterteilt. Durchschnittlicher Helligkeitswert:
Ein durchschnittlicher Wert von 1 Pixel, erhalten durch Dividieren des gemessenen Helligkeitswerts in jedem unterteilten Pixel-Fenster durch die Anzahl der Pixel (n Pixel) in jedem Fenster. Geben Sie den Stufenwert mit dem durchschnittlichen Helligkeitswert an (0~255).

Relation zwischen Stufe und Kante



Die Grafiken zeigen die Änderung der Helligkeit in einem Fenster.
Die erkannte Kantenposition entspricht den Punkten a und b, wobei die Helligkeit eine angegebene Stufe in Grafik (1) durchläuft.
In Grafik (2) ist c der Punkt, an dem die Helligkeit die angegebene Stufe erreicht.
Wenn die Helligkeit die angegebene Stufe nicht erreicht, wird die Kante von schwarz nach weiß erkannt.
Wenn die Helligkeit eine höhere Stufe als die angegebene erreicht, wird die Kante von weiß nach schwarz erkannt.
Wenn sich die Helligkeit von der gleichen Stufe wie der angegebenen Stufe in einen niedrigeren Wert ändert, wie bei Punkt d in der Grafik (2), wird dies nicht as Kante erkannt.

<Binäre Untergrenze> Gibt die Untergrenze für die Binärisierung (0 bis 254 Untergrenze < Obergrenze) an. Der Standardwert ist 0.
<Binäre Obergrenze> Gibt die Obergrenze für die Binärisierung (1 bis 255 Untergrenze < Obergrenze) an. Der Standardwert lautet 255.

Anmerkung (1): Gibt den Verarbeitungsbereich mit einem Fenster an.

○: Verfügbar ×: Nicht verfügbar

Linie (2-Punkt-Angabe)	Windmake P	○
Linie (Länge, Winkel)	Windmake L	○
Kreis	Windmake C	×
Ellipse	Windmake E	×
Sektor	Windmake S	○
Rechteck	Windmake R	○

Anmerkung (2): Wenn das angegebene Fenster größer ist als der Bildschirm, tritt ein Fehler auf.

Anmerkung (3): Die folgenden Daten können mit der Funktion zum Abrufen des Verarbeitungsergebnisses abgerufen werden.

	VISSTATUS (n)
n	Element
0	Ausführungsergebnis 0= Normal -1= Abnormal
1	Anzahl der erkannten Kanten
2	Ausführungszeit

	VISGETNUM (a, b)
b	a = 0~511
0	Unbekannt
1	Koordinate X Wert
2	Koordinate Y Wert
3	Winkel (Anmerkung (4))
4	Unbekannt
5	Unbekannt
6	Unbekannt
7	Unbekannt
8	Unbekannt
9	Unbekannt

Anmerkung (4): Gauging-Ergebnisse (Winkel) können nur abgerufen werden, wenn der Verarbeitungsbereich im Sektorfenster angegeben ist (Modus=0).

Anmerkung (5): Das Verarbeitungsobjekt ist der mit VISWORKPLN definierte Bildschirm.

Anmerkung (6): Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.

Anmerkung (7): Durch Vergrößern der Anzahl der Schritte wird die Erkennungszeit reduziert; die Leistung bei der Auflösung zur Erkennung wird jedoch verringert.

Verwandte Begriffe

WINDMAKE, VISWORKPLN, VISGETNUM, VISSTATUS

Beispiel

```
VISSCREEN 1,0,1                'Zeichnet sofort auf das Prozeßfenster 0.
VISPLNOUT 0                    '
VISCLS 0                       '
WINDMAKE R,1,300,20,0,0        'Stellt Fenster 1 auf Rechteck ein.
CAMIN 2                        'Ruft ein Kamerabild aus dem Speicher ab.
VISWORKPLN 0                   'Gibt das Objekt für Speicher 0 an.
VISPLNOUT 0                    '
VISEDGE 1,100,100,1,0,128      'Mißt eine Kante.
WINDDISP 1                     '
I1 = VISSTATUS(0)              '
IF I1 = 0 THEN                 '
    FOR I1 = 0 TO VISSTATUS(1)-1
                               '
        VISCROSS VISPOSX(I1),VISPOSY(I1)
                               '
    NEXT I1                    '
    I1 = VISSTATUS(1)          '
    IF I1 = 0 THEN             '
        VISLOC 10,10           '
        VISPRINT “Es wurde keine Kante gefunden.”
                               '
    END IF                     '
ELSEIF I1 <> 0 THEN            '
    VISLOC 10,10               '
    VISPRINT “Es kann keine Messung ausgeführt werden.”
                               '
END IF                         '
```

21.6 Codeerkennung

VISREADQR (Anweisung)

Funktion

Liest den QR-Code.

Format

VISREADQR <Fensternummer>, <Koordinate X>, <Koordinate Y>,
<Modus> [, <Binäre Untergrenze>[, <Binäre Obergrenze>]]

Erklärung

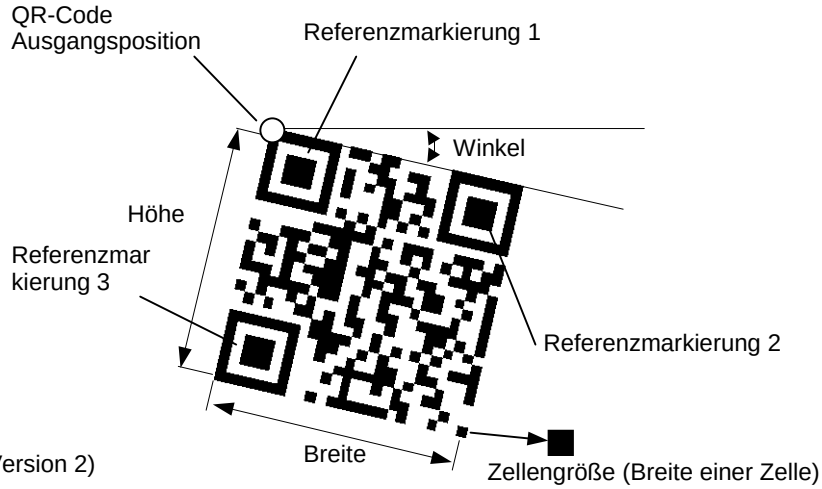
<Fensternummer> Gibt die Fensternummer (0 bis 511) an.
<Koordinate X> Gibt die X-Koordinate (0 bis 511) an.
<Koordinate Y> Gibt die Y-Koordinate (0 bis 479) an.
<Modus> Gibt die Methode für die Binärisierung an.

- 0: Automatische Binärisierung (Binarisiert den Verarbeitungsbereich mit der Entscheidungsanalysemethode.)
- 1: Angegebener Binärisierungswert (Binärisierung auf der Basis der Binärisierungsstufe in den Anweisungen.)

<Binäre Untergrenze> Gibt die Untergrenze für die Binärisierung (0 bis 254 Untergrenze < Obergrenze) an.
Wird diese Option nicht angegeben, gilt der Standardwert 0.

<Binäre Obergrenze> Gibt die Obergrenze für die Binärisierung (1 bis 255 Untergrenze < Obergrenze) an.
Wird diese Option nicht angegeben, gilt der Standardwert 255.

Beispiel eines QR-Codes (Version 2)



Anmerkung (1): Gibt den Verarbeitungsbereich mit einem Fenster an.
Anmerkung (2): Die einzige Fensterform, die Sie angeben können, ist ein Rechteck mit 0 Grad. Wenn eine andere Fensterform angegeben wird, tritt ein Fehler auf.
Anmerkung (3): Das Verarbeitungsobjekt ist der mit VISWORKPLN angegebene Bildschirm.
Anmerkung (4): Die Speicherkarte speichert den Inhalt der Codes, und Sie können die Codes mit VISGETSTR abrufen.
Anmerkung (5): Wenn der Code einen Fehler aufweist und nicht gelesen werden kann, lautet das Ergebnis von VISSTATUS (0) -1.
Anmerkung (6): Die folgenden Daten können mit der Funktion zum Abrufen des Verarbeitungsergebnisses abgerufen werden.

	VISSTATUS(n)
n	Element
0	Ausführungsergebnis 0= Normal -1=Abnormal
1	Anzahl der Zeichen (Einheit Byte)
2	Ausführungszeit

Anmerkung (7): Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.

	VISGETNUM (a, b)			
b	a = 0	a = 1	a = 2	a = 3
0	Anzahl der Zeichen (Einheit Byte)	Unbekannt	Unbekannt	Unbekannt
1	Koordinate X Wert	Referenzmarkierung 1 X Koordinatenwert	Referenzmarkierung 2 X Koordinatenwert	Referenzmarkierung 3 X Koordinatenwert
2	Koordinate Y Wert	Referenzmarkierung 1 Y Koordinatenwert	Referenzmarkierung 2 Y Koordinatenwert	Referenzmarkierung 3 Y Koordinatenwert
3	Winkel	Unbekannt	Unbekannt	Unbekannt
4	Breite	Unbekannt	Unbekannt	Unbekannt
5	Höhe	Unbekannt	Unbekannt	Unbekannt
6	Version	Unbekannt	Unbekannt	Unbekannt
7	Zellengröße	Unbekannt	Unbekannt	Unbekannt
8	Modellidentifikation [V1.4 oder neuer]	Unbekannt	Unbekannt	Unbekannt
9	Unbekannt	Unbekannt	Unbekannt	Unbekannt

Verwandte Begriffe

WINDMAKE, VISWORKPLN, VISGETNUM, VISSTATUS, VISGETSTR

Beispiel

```
VISSCREEN 1,0,1                '
VISCLS                         '
WINDMAKE R,1,512,480,0,2       'Stellt Fenster 1 auf Rechteck ein.
CAMIN 1                        'Ruft ein Kamerabild aus dem Speicher ab.
VISPLNOUT 0                    '
VISREADQR 1,0,0,0              'Liest den QR-Code.
I1 = VISSTATUS(0)              '
VISPRINT I1,VISSTATUS(1)       '
IF I1 = 0 THEN                 '
    VISLOC 10,10               '
    VISPRINT VISGETSTR(1,VISSTATUS(1))
                               '
END IF                         '
VISCAMOUT 1                    '
```

21.7 Kennzeichnung

BLOB (Anweisung)

Funktion

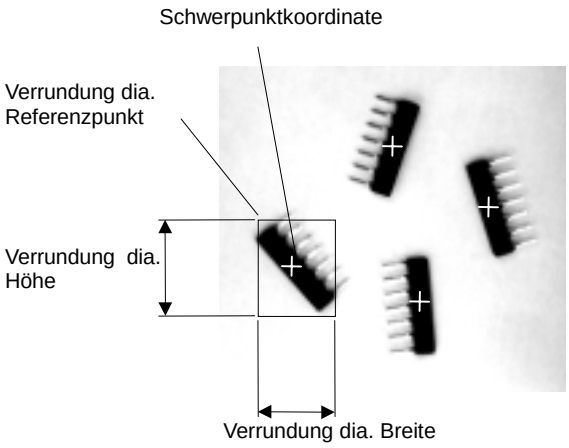
Kennzeichnung ausführen.

Format

BLOB <Fensternummer>, <Koordinate X>, <Koordinate Y>, <Verarbeitungs-objekt>, <Binäre Untergrenze>[, <Binäre Obergrenze>[, <Verbindung>[, <Bereichsuntergrenze>[, <Sortierung>]]]]

Erklärung

<Fensternummer> Gibt die Fensternummer (0 bis 511) an.
<Koordinate X> Gibt die X-Koordinate (0 bis 511) an.
<Koordinate Y> Gibt die Y-Koordinate (0 bis 479) an.
<Verarbeitungsobjekt> Gibt das Objekt an, das mit der Kennzeichnung (0 oder 1) abgerufen werden soll.
0: Schwarz (Helligkeitswert < binäre Untergrenze, binäre Obergrenze < Helligkeitswert)
1: Weiß (binäre Untergrenze ≤ Helligkeitswert ≤ binäre Obergrenze)
<Binäre Untergrenze> Gibt die Untergrenze für die Binärisierung an (0 bis 254 Untergrenze < Obergrenze).
<Binäre Obergrenze> Gibt die Obergrenze für die Binärisierung an (1 bis 255 Untergrenze < Obergrenze).
Wird diese Option nicht angegeben, gilt der Standardwert 255.
<Verbindung> Gibt die Bedingung der Verbindung an (0 oder 1).
0: Nachbar 4 Verbindung (benachbarte Pixel rechts und links sowie oben und unten werden geprüft.)
1: Nachbar 8 Verbindung (benachbarte Pixel rechts und links, oben und unten sowie diagonal werden geprüft.)
<Bereichsuntergrenze> Gibt einen unteren Grenzwert eines bei der Kennzeichnung zu ignorierenden Bereichswert an. (0 bis 245760)
<Sortierung> Gibt die Sortierung für die mit der Kennzeichnung abgerufenen Zahlen an (0 bis 2).
0: Abgerufene Reihenfolge
1: Bereichswert absteigend
2: Bereichswert aufsteigend



Durch Ausführung einer Kennzeichnung können Sie die Zahlen und Positionen mehrerer Objekte abrufen. Im Beispiel auf der linken Seite lautet die Kennzeichnungsnummer 4.
Wenn Sie BLOBMEASURE verwenden, können Sie die Indination (Richtung : Hauptachsenwinkel) und die Peripherielänge einzeln angeben.

Beispiel für die Kennzeichnung

Anmerkung (1): Gibt den Verarbeitungsbereich mit einem Fenster an.
Anmerkung (2): Wenn die angegebene Fensterposition außerhalb des Bildschirms liegt, führt die Ausführung zu einem Fehler.
Anmerkung (3): Die einzige Fensterform, die Sie angeben können, ist ein Rechteck mit 0 Grad. Wenn eine andere Fensterform angegeben wird, tritt ein Fehler auf.
Anmerkung (4): Das Verarbeitungsobjekt ist ein mit VISWORKPLN angegebener Bildschirm.
Anmerkung (5): Die folgenden Daten können mit der Funktion zum Abrufen des Verarbeitungsergebnisses abgerufen werden.

	VISSTATUS (n)
n	Element
0	Ausführungsergebnis 0= Normal -1=Abnormal
1	Anzahl der Kennungen
2	Ausführungszeit

	VISGETNUM (a, b)
b	a = 0~ max.511
0	Bereich
1	Schwerpunkt X-Koordinatenwert
2	Schwerpunkt Y-Koordinatenwert
3	Unbekannt
4	Verrundung dia. Referenzpunkt Koordinate Y
5	Verrundung dia. Referenzpunkt Koordinate X
6	Verrundung dia. Breite
7	Verrundung dia. Höhe
8	Unbekannt

Anmerkung (6): Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.

Verwandte Begriffe

WINDMAKE, VISWORKPLN, VISGETNUM, VISSTATUS

Beispiel

```
VISSCREEN 1,0,1                'Zeichnet sofort auf das Prozeßfenster 0.
VISCLS 0                       '
WINDMAKE R,1,512,480,0,2       'Stellt Fenster 1 auf Rechteck ein.
CAMIN 1                        'Ruft ein Kamerabild für den Speicher 0 ab.
VISPLNOUT 0                    '
VISWORKPLN 0                   'Gibt das Objekt für Speicher 0 an.
BLOB 1,0,0,0,128               'Führt die Kennzeichnung aus.
I1 = VISSTATUS(0)              '
IF I1 = 0 THEN                 '
    I2 = VISSTATUS(1)          '
    VISDEFCHAR 1,1,2           '
    VISLOC 10,10               '
    VISPRINT I1,I2             '
    IF I2 <> 0 THEN            '
        FOR I1 = 0 TO I2 -1    '
            VISLOC 10,11       '
            VISPRINT VISGETNUM(I1,1), VISGETNUM(I1,2)
                               '
            VISCROSS VISGETNUM(I1,1), VISGETNUM(I1,2)
                               '
        NEXT I1
    END IF                     '
END IF                         '
VISCAMOUT 1                    '
```

BLOBMEASURE (Anweisung)

Funktion

Führt eine Funktionsmessung der Objektkennungsnummer aus.

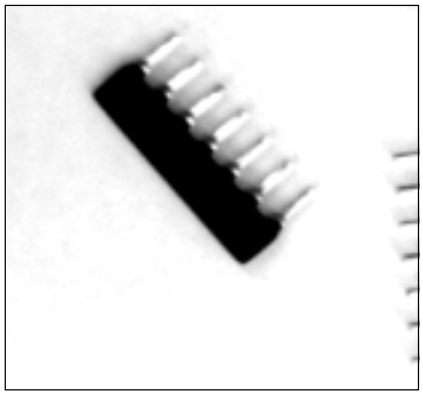
Format

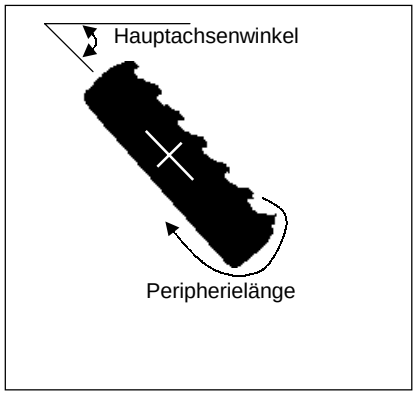
BLOBMEASURE <Kennungsnummer>, <Funktion>

Erklärung

<Kennungsnummer> Gibt die mit der Kennzeichnung abgerufene Kennungsnummer an. (0 bis 511)

<Funktion> Gibt die abzurufende Funktion an. (0 oder 1)
0: Hauptachsenwinkel
1: Peripherielänge





Die Binärisierung erfolgt automatisch intern, ohne das ursprüngliche Bild zu ändern.
Sie brauchen vorab keine Binärisierung durchzuführen.

Anmerkung (1): Vor der Ausführung dieser Funktion müssen Sie eine Kennzeichnung mit BLOB ausführen.

Anmerkung (2): Wenn Sie die Peripherielänge abrufen, müssen Sie das Ausgangsbild nach der Ausführung der Kennzeichnung verlassen.

Anmerkung (3): Das folgende Beispiel zeigt die Daten, die Sie mit der Funktion zum Abrufen des Verarbeitungsergebnisses abrufen können.

	VISSTATUS (n)
n	Element
0	Ausführungsergebnis 0=Normal -1=Abnormal
1	Unbekannt
2	Ausführungszeit

	VISGETNUM (a, b)
b	a = 0~ max.511
0	Bereich
1	Schwerpunkt X-Koordinatenwert
2	Schwerpunkt Y-Koordinatenwert
3	Hauptachsenwinkel
4	Verrundung dia. Referenzpunkt Koordinate X
5	Verrundung dia. Referenzpunkt Koordinate Y
6	Verrundung dia. Breite
7	Verrundung dia. Höhe

Anmerkung (4): Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.

Verwandte Begriffe

BLOB

Beispiel

```
VISSCREEN 1,0,1                 'Zeichnet sofort den Zeichnungsbildschirm 0.
WINDMAKE R,1,512,480,0,2        'Stellt Fenster 1 auf Rechteck ein.
CAMIN 1                         'Ruft ein Kamerabild aus dem Speicher ab.
BLOB 1,0,0,0,128                'Führt die Kennzeichnung aus.
IF VISSTATUS(0)=0.0 THEN        '
    IF VISSTATUS(1)<>0.0 THEN   '
        FOR I1 = 0 TO VISSTATUS(1)-1
                                '
            BLOBMEASURE I1,0    'Ruft den Hauptachsenwinkel von Kennung 1 ab.
            VISCROSS VISGETNUM(I1,1), VISGETNUM(I1,2),10,20,VISGETNUM(I1,3)
       NEXT I1                  '
    END IF                      '
END IF                          '
```

BLOBLABEL (Funktion)

Funktion

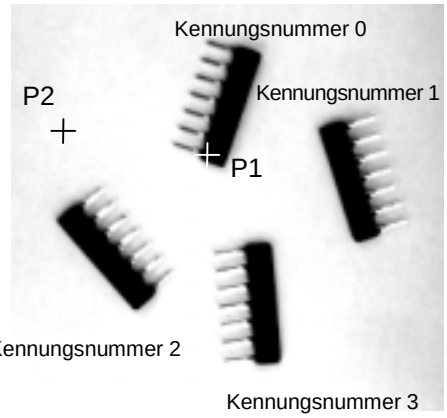
Ruft die Kennungsnummer für angegebene Koordinaten ab.

Format

BLOBLABEL(<Koordinate X>, <Koordinate Y>)

Erklärung

<Koordinate X> Gibt die X-Koordinate (0 bis 511) an.
<Koordinate Y> Gibt die Y-Koordinate (0 bis 479) an.



BLOBLABEL prüft, ob die angegebenen Koordinaten in den gekennzeichneten Objekten enthalten sind.
Wenn Sie die Koordinaten von P1 mit BLOBLABEL für ein gemessenes Bild angeben, wenn die Anzahl der zu kennzeichnenden Objekte 4 beträgt und die Kennungsnummer wie in der linken Abbildung dargestellt wird, wird der Rückgabewert für die Kennungsnummer 0.
Wenn Sie P2 angeben, wird der Rückgabewert -1, da es kein entsprechendes Objekt gibt.

Beispiel zum Abrufen der Kennungsnummer

Anmerkung (1): Vor der Ausführung dieser Funktion müssen Sie eine Kennzeichnung mit BLOB ausführen.
Anmerkung (2): Rufen Sie die für die angegebenen Koordinaten vorhandene Kennungsnummer ab.
Anmerkung (3): Wenn keine Kennungsnummer vorhanden ist, wird -1 zurückgegeben.
Anmerkung (4): Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.

Verwandte Begriffe

BLOB

Beispiel

```
VISSCREEN 1,0,1                 'Zeichnet sofort das Prozeßfenster 0.
WINDMAKE R,1,512,480,0,2        'Stellt Fenster 1 auf Rechteck ein.
CAMIN 1                         'Ruft ein Kamerabild aus dem Speicher ab.
BLOB 1,0,0,0,128                'Führt die Kennzeichnung aus.
IF VISSTATUS(0)=0.0 THEN        '
    IF VISSTATUS(1)<>0.0 THEN   '
        IF BLOBLABEL(100,100)<>-1 THEN
                                '
            VISLOC 100,100,1    '
            VISPRINT “Kennungsnummer =”;BLOBLABEL(100,100)
                                '
        END IF                  '
    END IF                      '
END IF                          '
```

BLOBCOPY (Anweisung)

Funktion

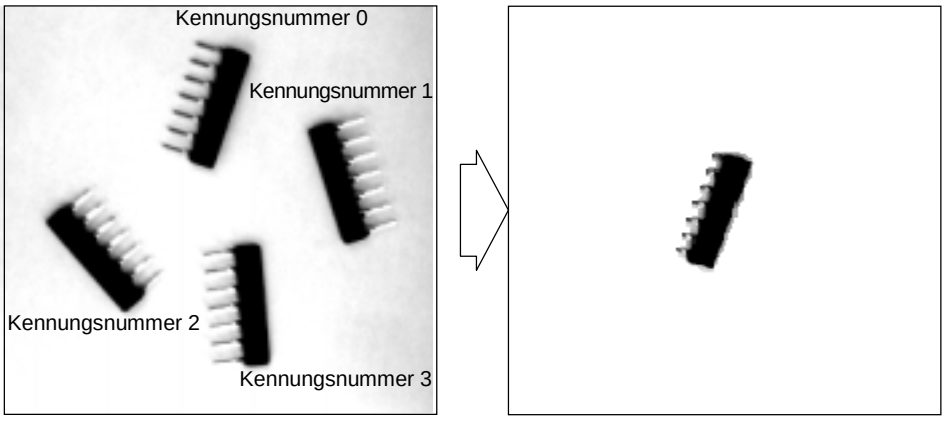
Kopiert eine Objektkennungsnummer.

Format

BLOBCOPY <Kennungsnummer>, <Kopierzielbildschirm>, <Koordinate X>, <Koordinate Y>

Erklärung

<Kennungsnummer> Gibt die Kennungsnummer an, die mit der Kennzeichnung (0 bis 511) abgerufen wurde.
<Kopierzielbildschirm> Gibt die Speichernummer des Kopierziels (0 bis 3) an.
<Koordinate X> Gibt die X-Koordinate (0 bis 511) an.
<Koordinate Y> Gibt die Y-Koordinate (0 bis 479) an.



Das Objektbild wird nach dem Kopieren mit einer Kennungsnummer angegeben.
Im Beispiel oben wird die Kennungsnummer 0 in das Prozeßfenster 1 kopiert.

Beispiel für BLOBCOPY

Anmerkung (1): Vor der Ausführung dieser Funktion müssen Sie eine Kennzeichnung mit BLOB ausführen.
Anmerkung (2): Ändern Sie diesen Bildschirm nicht, da beim Kopieren auf den Kopierquellbildschirm (Kennzeichnungs-objekt-bildschirm) verwiesen wird.
Anmerkung (3): Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.

Verwandte Begriffe

BLOB

Beispiel

```
VISSCREEN 0,1,                  'Zeichnet sofort auf Speicher 1.
VISCLS 128                      'Löscht den Bildschirm.
VISSCREEN 1,0,1                 'Zeichnet sofort auf das Prozeßfenster 0.
VISCLS 0                        '
WINDMAKE R,1,512,480,0,2        'Stellt Fenster 1 auf Rechteck ein.
CAMIN 1                         'Ruft ein Kamerabild für den Speicher ab.
BLOB 1,0,0,0,128                'Führt die Kennzeichnung aus.
I1 = VISSTATUS(0)               '
IF I1 = 0 THEN                  '
    I1 = VISSTATUS(1)           '
    IF I1<>0 THEN               '
        BLOBCOPY 0,1,100,100    'Kopiert Kennung 0 in den Speicher 1.
    END IF                      '
END IF                          '
VISPLNOUT 0                     '
DELAY 2000                      '
VISPLNOUT 1                     '
DELAY 2000                      '
VISCAMOU VISCAMOUT 1            '
```

21.8 Suchfunktion

SHDEFMODEL (Anweisung)

Funktion

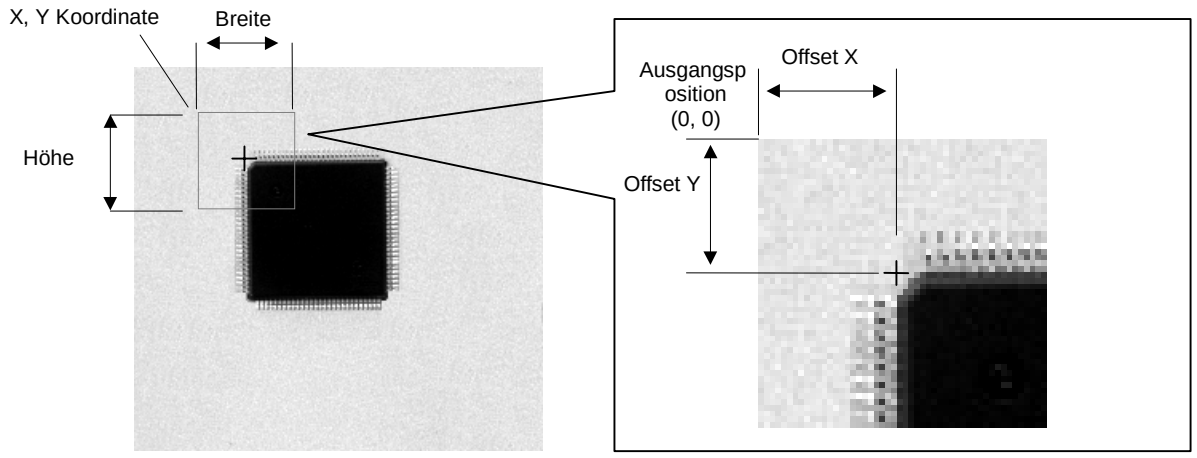
Registriert das Suchmodell.

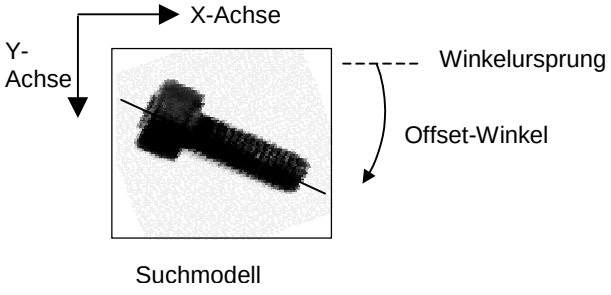
Format

SHDEFMODEL <Modellnummer>, <Koordinate X>, <Koordinate Y>, <Breite>, <Höhe>, <Offset X>, <Offset Y>[, <Offset-Winkel>]

Erklärung

<Modellnummer> Gibt die zu registrierende Modellnummer an (0 bis 99).
<Koordinate X> Gibt die X-Koordinate der Ausgangsposition (16 bis 485) an.
<Koordinate Y> Gibt die Y-Koordinate der Ausgangsposition (16 bis 463) an.
<Breite> Gibt die Breite des registrierten Modells (10 bis 256) an.
<Höhe> Gibt die Höhe des registrierten Modells (10 bis 256) an.
<Offset X> Gibt den X-Offset, gemessen vom Ursprung (-511 to +511) an.
<Offset Y> Gibt den Y-Offset, gemessen vom Ursprung (-511 to +511) an.
<Offset-Winkel> Gibt den Offset-Winkel (-360 bis 360), gemessen vom Ursprung, an.
[V1.4 oder neuer]





Sie geben einen Offset-Wert, gemessen vom Winkelursprung zum [Offset-Winkel], an. Wenn ein Suchmodell einen Offset-Winkel hat, gemessen vom Winkelursprung wie in der Abbildung angegeben, können Sie [Offset-Winkel] angeben, um den Offset-Winkel zu [Winkel] in einem mit [SHMODEL] erhaltenen gemessenen Ergebnis anzugeben.

Anmerkung (1): Wenn das zu registrierende Modell nicht mehr als 16 Pixel innerhalb der Kante des Bildschirmes liegt, kann es nicht registriert werden.

Anmerkung (2): Die Referenzkoordinaten werden zum Angeben eines Punktes verwendet, mit dem erkannt wird, wenn ein Modell gesucht wird.

Anmerkung (3): Ein registriertes Modell erfordert einen bestimmten Betrag für die Helligkeitsverteilung. Wenn die Helligkeitsverteilung zu niedrig ist oder sehr kleine Unterschiede in dem Modell vorliegen, kann es nicht registriert werden.

Anmerkung (4): Wenn Sie eine Zahl angeben, die bereits registriert wurde, überschreibt sie das bereits registrierte Modell.

Anmerkung (5): Sie finden die Verfügbarkeit der Registrierung über die Verwendung von VISSTATUS nach der Ausführung der Anweisung.

Anmerkung (6): Bei der Ausführung wird das Prozeßfenster 3 (VISWORKPLN 3) als Arbeitsbereich verwendet, und die Daten auf dem Prozeßfenster 3 sind nicht garantiert. Außerdem können Sie das Prozeßfenster 3 nicht für die Verarbeitung verwenden.

	VISSTATUS (n)
n	Element
0	Ausführungsergebnis 0 = Normal -1 = Angegebener Bereichsfehler -2 = Abstand voll -3 = Gleichförmiges Modell -4 = Kompliziert 1,2 = Suchzeit lang 3,4 = Winkel nicht definierbar [V1.4 oder neuer]
1	Nicht verwendet 0
2	Ausführungszeit

Anmerkung (7): Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.

Anmerkung (8): Schalten Sie während der Ausführung dieser Anweisung den Controller nicht aus. Wenn Sie den Controller ausschalten, erkennt er in der nächsten Einschaltsequenz nicht, daß er nicht normal beendet wurde und initialisiert die Bildverarbeitungsinformationen neu.

Verwandte Begriffe

SHREFMODEL, SHDISPMODEL, SHCLRMODEL, SHMODEL

Beispiel

```
VISSCREEN 1,0,1              'Zeichnet sofort auf dem Zeichnungsbildschirm 0.
VISCLS 0                     '
VISRECT 100,100,100,100      '
CAMIN 1                      'Ruft ein Kamerabild aus dem Speicher ab.
SHDEFMODEL 1,100,100,100,100,50,50
                             'Registriert das Modell.
I1 = VISSTATUS(0)            '
VISLOC 10,10                 '
IF I1 = 0 THEN VISPRINT “Registrierung abgeschlossen” ELSE VISPRINT “Registrierung
nicht möglich.”
VISCAMOUT 1                  '
```

SHREFMODEL (Anweisung)

Funktion

Bezieht sich auf die Daten des registrierten Modelles.

Format

SHREFMODEL (<Modellnummer>, <Element>)

Erklärung

<Modellnummer> Gibt die Referenzmodellnummer an (0 bis 99).
<Element> Gibt den Datentyp eines Referenzmodells (0 bis 8) an.

0: Status (Präsenz der Modellregistrierung
Ja = 0 Nein = -1)
1: Breite des registrierten Modells
2: Höhe des registrierten Modells
3: Offset X des registrierten Modells
4: Offset Y des registrierten Modells
5: Dateigröße des registrierten Modells
6: Registrierbare Dateigröße (Verfügbare Kapazität).
7. Offset-Winkel des registrierten Modells [V1.4 oder neuer]
8: Kompatibler Modus (1: für den Modus ohne Winkelmessung, 2: für den Modus mit Winkelmessung, 3 für den Modus mit und ohne Winkelmessung) [V1.4 oder neuer]

Verwandte Begriffe

SHDEFMODEL

Beispiel

```
VISSCREEN 1,0,1                          '
VISCLS 0                                 '
IF SHREFMODEL(1,0) = 0 THEN              'Bestätigt die Präsenz der Registrierung.
    VISLOC 10,1                          '
    VISPRINT “Breite =”;SHREFMODEL(1,1)  'Zeigt die Breite an.
    VISLOC 10,2                          '
    VISPRINT “Höhe =”;SHREFMODEL(1,2)    'Zeigt die Höhe an.
    VISLOC 10,3                          '
    VISPRINT “X=”;SHREFMODEL(1,3)        'Zeigt die X-Referenzkoordinate an.
    VISLOC 10,4                          '
    VISPRINT “Y=”;SHREFMODEL(1,4)        'Zeigt die Y-Referenzkoordinate an.
    VISLOC 10,5                          '
    VISPRINT “Größe =”;SHREFMODEL(1,5)   'Zeigt die Dateigröße an.
    VISLOC 10,6                          '
    VISPRINT “Kapazität =”;SHREFMODEL(1,6)
                                         'Zeigt die verfügbare Kapazität an.
END IF                                   '
```

SHCOPYMODEL (Anweisung)

Funktion

Kopiert ein registriertes Modell.

Format

SHCOPYMODEL <Nummer des Kopierquellmodells>,<Nummer des Kopierzielmodells>

Erklärung

<Nummer des Kopierquellmodells> Gibt die Nummer des zu kopierenden Modells an (0 bis 99).
<Nummer des Kopierzielmodells> Gibt die neue Modellnummer an (0 bis 99).

Anmerkung (1): Wenn die Nummern für Kopierquelle und Kopierziel identisch sind, tritt ein Fehler auf.

Anmerkung (2): Wenn die Kopierquelldatei nicht vorhanden ist oder die Kapazität nicht ausreicht, wird das Kopieren nicht ausgeführt.

Anmerkung (3): Die folgenden Daten können mit der Funktion zum Abrufen des Verarbeitungsergebnisses abgerufen werden.

	VISSTATUS (n)
n	Element
0	Ausgeführtes Ergebnis 0=Normal -1=Abnormal
1	Unbekannt
2	Unbekannt

Anmerkung (4): Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.

Anmerkung (5): Schalten Sie während der Ausführung dieser Anweisung den Controller nicht aus. Wenn Sie den Controller ausschalten, erkennt er in der nächsten Einschaltsequenz nicht, daß er nicht normal beendet wurde und initialisiert die Bildverarbeitungsinformationen neu.

Verwandte Begriffe

SHDEFMODEL, SHREFMODEL

Beispiel

```
VISSCREEN 1,0,1          '
VISCLS 0                 '
SHCOPYMODEL 2,0          'Kopiert Modell 2 Informationen in Modell 0.
VISLOC 10,10             '
I1 = VISSTATUS(0)
IF I1 = 0 THEN VISPRINT “Kopieren fertig” ELSE VISPRINT “Kopieren fehlgeschlagen”
                         '
```

SHCLRMODEL (Anweisung)

Funktion

Löscht ein registriertes Modell.

Format

SHCLRMODEL <Modellnummer>

Erklärung

<Modellnummer> Gibt die zu löschende Modellnummer an (0 bis 99).

Anmerkung (1):	**Wenn die angegebene Modellnummer nicht vorhanden ist, erfolgt keine Ausführung.**
Anmerkung (2):	**Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.**
Anmerkung (3):	**Schalten Sie während der Ausführung dieser Anweisung den Controller nicht aus. Wenn Sie den Controller ausschalten, erkennt er in der nächsten Einschaltsequenz nicht, daß er nicht normal beendet wurde und initialisiert die Bildverarbeitungsinformationen neu.**

Verwandte Begriffe

SHDEFMODEL, SHREFMODEL

Beispiel

```
VISSCREEN 1,0,1          '
VISCLS 0                 '
SHCLRMODEL 1             'Löscht das registrierte Modell 1.
I1 = SHREFMODEL(1,0)     '
VISLOC 10,10             '
IF I1 = -1 THEN VISPRINT “Löschen abgeschlossen.” ELSE VISPRINT “Löschen
fehlgeschlagen.”
                         '
```

SHDISPMODEL (Anweisung)

Funktion

Zeigt ein registriertes Modell auf dem Bildschirm an.

Format

SHDISPMODEL <Modellnummer>, <Anzeigebildschirm>, <Koordinate X>, <Koordinate Y>

Erklärung

<Modellnummer> Gibt die anzuzeigende Modellnummer an (0 bis 99).
<Anzuzeigender Zielbildschirm> Gibt die Nummer des anzuzeigenden Speichers an (0 bis 3).
<Koordinate X> Gibt die X-Koordinate der Ausgangsposition des registrierten Modells an (0 bis 511).
<Koordinate Y> Gibt die Y-Koordinate der Ausgangsposition des registrierten Modells an (0 bis 479).

Anmerkung (1): Wenn die angegebene Modellnummer nicht vorhanden ist, erfolgt keine Ausführung.
Anmerkung (2): Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.

Verwandte Begriffe

SHDEFMODEL

Beispiel

```
VISSCREEN 0,0,1              'Zeichnet sofort auf Speicher 0.
VISPLNOUT 0                  'Zeigt Speicher 0 auf dem Bildschirm an.
SHDISPMODEL 1,0,100,100      'Zeigt Modell 1 auf dem Bildschirm an.
```

SHMODEL (Anweisung)

Funktion

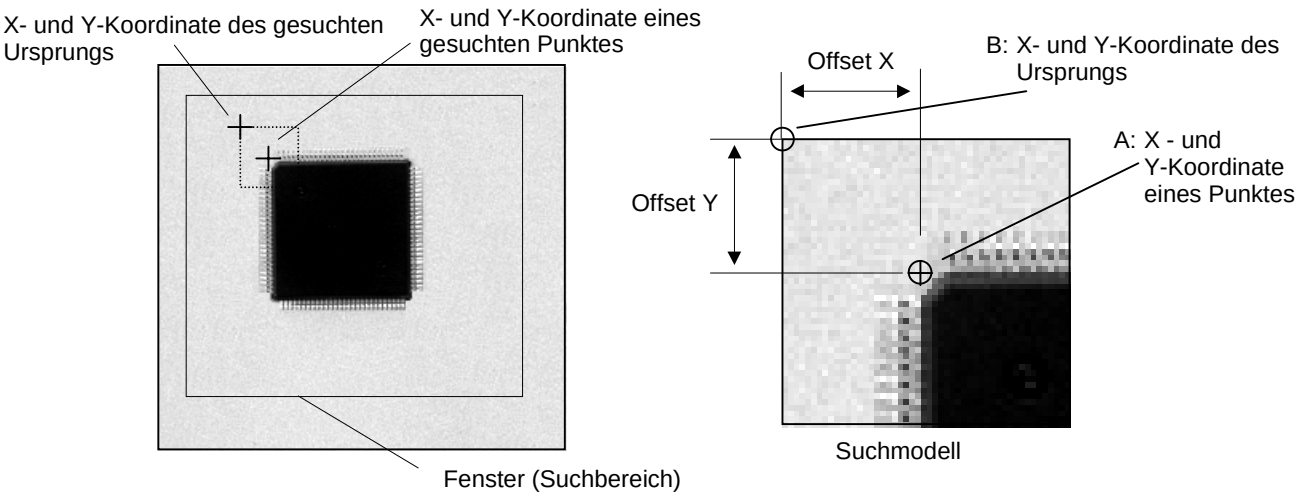
Sucht ein Modell.

Format

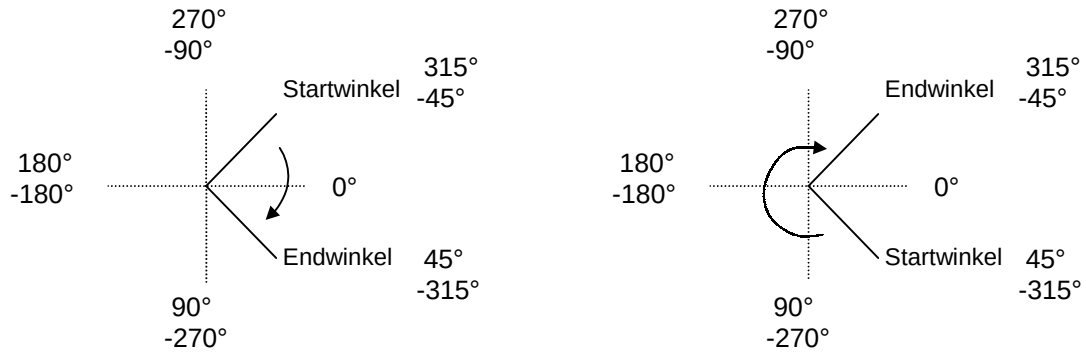
SHMODEL <Fensternummer>, <Koordinate X>, <Koordinate Y>, <Modellnummer>, <Korrespondenzwinkel>[, <Modus>[, <Nummer>[, <Startwinkel>[, <Endwinkel>]]]]

Erklärung

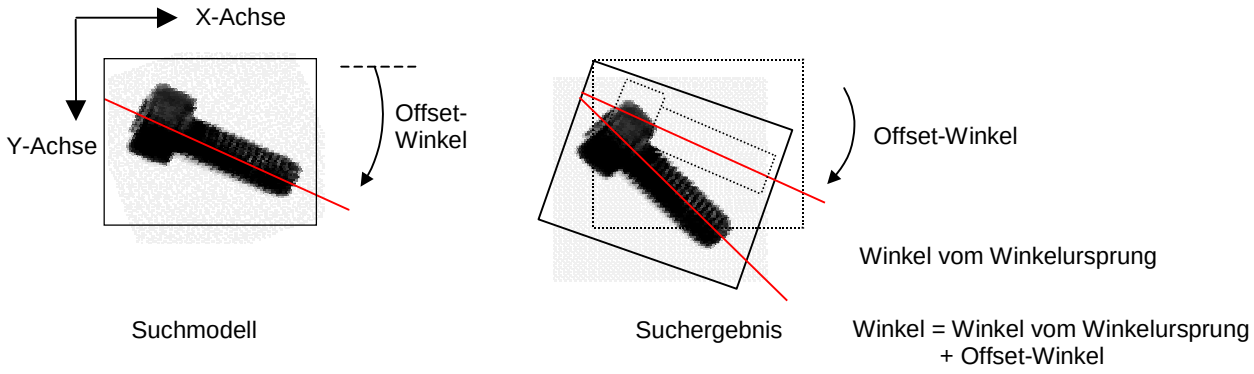
<Fensternummer> Gibt die Fensternummer (0 bis 511) an.
<Koordinate X> Gibt die X-Koordinate (0 bis 511) an.
<Koordinate Y> Gibt die Y-Koordinate (0 bis 479) an.
<Modellnummer> Gibt die Nummer des zu suchenden Modells an (0 bis 99).
<Korrespondenzwinkel> Gibt den Korrespondenzwinkel des zu suchenden Modells (0 bis 100) an.
<Modus> Gibt den Modus für die Ausführung der Suche an.
0: Suche in Pixeleinheiten.
1: Suche in Subpixeleinheiten.
Wird diese Option nicht angegeben, gilt der Standardwert 0.
<Anzahl> Gibt die Anzahl der gleichzeitig zu suchenden Modelle an (1 bis 49). Wenn diese Option nicht angegeben wird, gilt die Angabe 1.
<Startwinkel> Gibt einen Winkelbereich (Startwinkel) für ein zu suchendes Objekt (-360 bis 360) an. Der Standardwert lautet 0. [V1.4 oder neuer]
<Endwinkel> Gibt einen Winkelbereich (Endwinkel) für ein zu suchendes Objekt (-360 bis 360) an. Der Standardwert lautet 0. [V1.4 oder neuer]



Nach Abschluß einer Suche wird die Position zum Punkt A eines Suchmodells im [X, Y-Koordinatenwert] gespeichert; die Position zum Ursprung B eines Suchmodells wird in [X, Y-Koordinatenwert für Ursprung] gespeichert.



Mißt einen Winkel eines Objekts, dessen Winkel vom Winkelursprung in einem Bereich von [Startwinkel] bis [Endwinkel] liegt. Wie in der Abbildung gezeigt, liefern die Bedingungen (1) und (2) den gleichen Meßbereich.
(1) [Startwinkel] = -45°, [Endwinkel] = 45° (2) [Startwinkel] = -45°, [Endwinkel] = -315°



Ein Winkel, der aus einem als Suchmodell registrierten Bild abgeleitet wurde, wird in [Winkel aus Winkelursprung] als Messergebnis gespeichert. Die Summe aus dem in einem Suchmodell registrierten [Offset-Winkel] und [Winkel aus Winkelursprung] wird in [Winkel] als Messergebnis gespeichert.

Anmerkung (1):	**Registrieren Sie das zu suchende Modell mit SHDEFMODEL.**
Anmerkung (2):	**Wenn Sie zwei oder mehr Nummern angeben und das System die angegebene Nummer nicht finden kann, tritt ein Fehler auf.**
Anmerkung (3):	**Eine Suche braucht Zeit; wenn die Suche nicht innerhalb einer bestimmten Zeit abgeschlossen ist, tritt ein Zeitüberschreitungsfehler auf. Sie können diese Zeit auf Ihrem PC angeben. Die Anfangseinstellung lautet 2 Sekunden.**
Anmerkung (4):	**Wenn der Korrespondenzwinkel auf Hoch eingestellt ist, kann keine Suche ausgeführt werden; verringern Sie daher den Korrespondenzwinkel.**
Anmerkung (5):	**Geben Sie den Verarbeitungsbereich mit einem Fenster an.**
Anmerkung (6):	**Wenn die angegebene Fensterposition außerhalb des Bildschirmes liegt, führt die Ausführung zu einem Fehler.**

Anmerkung (7): Die einzige Fensterform, die Sie angeben können, ist ein Rechteck mit 0 Grad. Wenn Sie eine andere Form angeben, tritt ein Fehler auf.

Anmerkung (8): Das Verarbeitungsobjekt ist der mit VISWORKPLN angegebene Bildschirm.

Anmerkung (9): Bei der Ausführung wird das Prozeßfenster 3 (VISWORKPLN 3) als Arbeitsbereich verwendet, und die Daten auf dem Prozeßfenster 3 sind nicht garantiert. Außerdem können Sie das Prozeßfenster 3 nicht für die Verarbeitung verwenden.

Anmerkung (10): Sie können nach dem Ausführen einer Anweisung Daten mit der Funktion zum Abrufen des Verarbeitungsergebnisses abrufen.

	VISSTATUS (n)
n	Element
0	Ausführungsergebnis 0 = Normal -1 = Modell nicht registriert -2 = Fehler -3 = Zeitüberschreitung
1	Nummer
2	Ausführungszeit

	VISGETNUM (a, b)
b	a = 0~49
0	Nummer
1	X-Koordinatenwert
2	Y-Koordinatenwert
3	Korrespondenzwinkel
4	Winkel [V1.4 oder neuer]
5	X-Koordinatenwert für Ursprung [V1.4 oder neuer]
6	Y-Koordinatenwert für Ursprung [V1.4 oder neuer]
7	Winkel vom Winkelursprung [V1.4 oder neuer]
8	Unbekannt
9	Unbekannt

Anmerkung (11): Eine µBildverarbeitungskarte (optional) ist für diesen Befehl erforderlich.

Anmerkung (12): Wenn die Größe des Suchmodus und die Fenstergröße übereinstimmen, wird der Korrespondenzwinkel nur an dieser Position berechnet.

Anmerkung (13): Eine Messung beider Winkel wird ausgeführt, wenn Sie sowohl [Startwinkel] als auch [Endwinkel] auf 0 setzen.

Anmerkung (14): Eine Messung eines Winkels wird ausgeführt, wenn Sie [Startwinkel] = 0 und [Endwinkel] = 360 setzen oder [Startwinkel] und [Endwinkel] auf einen identischen Wert ungleich Null setzen.

Verwandte Begriffe

WINDMAKE, VISWORKPLN, VISGETNUM, VISSTATUS, SHDEFMODEL

Beispiel

```
VISSCREEN 1,0,1                'Zeichnet sofort auf das Prozeßfenster 0.
VISCLS 0                       '
VISPLNOUT 0                    '
WINDMAKE R,1,512,480,0,2       'Stellt Fenster 1 auf Rechteck ein.
CAMIN 1                        'Ruft ein Kamerabild aus dem Speicher ab.
VISWORKPLN 0                   'Gibt ein Objekt für Speicher 0 an.
SHMODEL 1,0,0,1,80             'Sucht Modell 1 und das Teil, dessen Korrespondenzteil
                               '80% des Bildschirmes beträgt.
I1 = VISSTATUS(0)              '
VISLOC 10,10                   '
VISDEFCHAR 1,1,3               '
VISPRINT I1                    '
IF I1 = 0 THEN                 '
    VISPRINT VISGETNUM(0,1),VISGETNUM(0,2)
                               '
    VISCROSS VISPOSX(0),VISPOSY(0)
                               '
END IF                         '
```

SHDEFCORNER (Anweisung)

Funktion

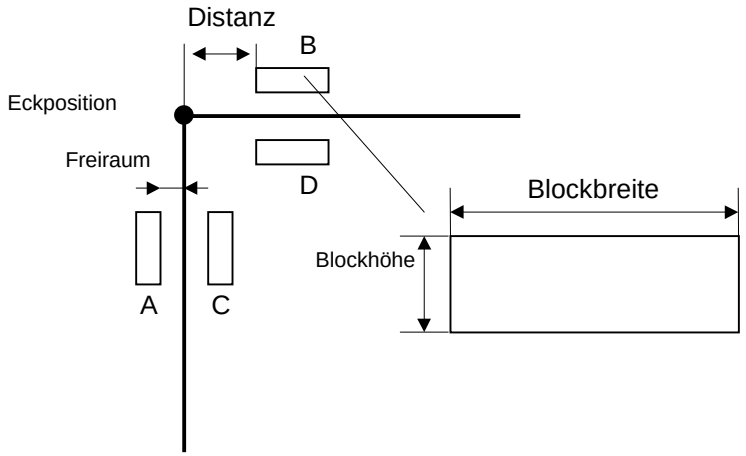
Legt die Bedingungen zum Suchen einer Ecke fest.

Format

SHDEFCORNER <Distanz>, <Freiraum>, <Breite>, <Höhe>

Erklärung

<Distanz> Gibt die Distanz von einer Ecke des Blockes bei einer Eckensuche an. (1 bis 10)
<Freiraum> Gibt den Abstand der Blöcke bei der Eckensuche an (1 bis 10).
<Breite> Gibt die Breite der Blöcke bei der Eckensuche an (1 bis 10).
<Höhe> Gibt die Höhe der Blöcke bei der Eckensuche an (1 bis 10).



Eine Ecke wird mit der durchschnittlichen Helle von 4 Blöcken erkannt, wie in der Abbildung links gezeigt.
Mit anderen Worten: Wenn SHCORNER 1 ist, wird eine Ecke erkannt, wo die durchschnittliche Helligkeit von A und B und die durchschnittliche Helligkeit von C und D größer ist als der Unterschied in den SHCORNER-Stufen.
Sie können den Block-Freiraum und die Breite mit SHDEFCORNER anpassen.

Anmerkung (1): Diese Anweisung entspricht einer temporären Einstellung; der Anfangswert wird nicht dauerhaft geändert.

Anmerkung (2): Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.

Anmerkung (3): Wenn der Freiraum zu breit gesetzt wird, wird er durch Störungen bei der Erkennung nicht beeinträchtigt; die Erkennung wird jedoch ungenau.

Anmerkung (4): Die vor der Auslieferung ab Werk eingestellten Anfangswerte lauten wie folgt: Distanz = 4, Freiraum = 1, Breite = 3 und Höhe =3.

Verwandte Begriffe

SHCORNER

Beispiel

```
SHDEFCORNER 4,1,3,3
```

SHCORNER (Anweisung)

Funktion

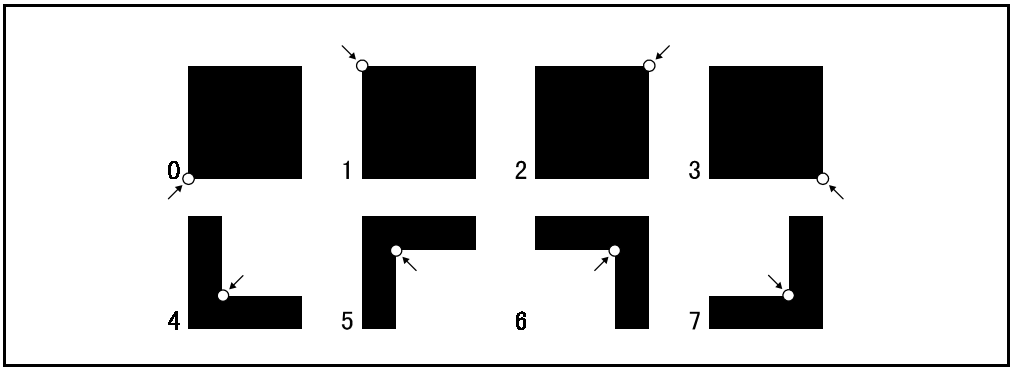
Sucht eine Ecke.

Format

SHCORNER <Fensternummer>, <Koordinate X>, <Koordinate Y>, <Niveauunterschied>, <Modus>

Erklärung

<Fensternummer> Gibt die Fensternummer (0 bis 511) an.
<Koordinate X> Gibt die X-Koordinate (0 bis 511) an.
<Koordinate Y> Gibt die Y-Koordinate (0 bis 479) an.
<Niveauunterschied> Gibt die Erkennungsstufe bei der Eckensuche an (0 bis 255).
<Modus> Gibt ein Objekt bei der Eckensuche an (0 bis 7).



Anmerkung (1): **Geben Sie den Verarbeitungsbereich mit einem Fenster an.**

Anmerkung (2): **Wenn die angegebene Fensterposition außerhalb des Bildschirmes liegt, führt die Ausführung zu einem Fehler.**

Anmerkung (3): **Nur rechteckige Fenster mit einem Winkel von 0 Grad können angegeben werden. Wenn Sie eine andere Form angeben, tritt ein Fehler auf.**

Anmerkung (4): **Das Verarbeitungsobjekt ist der mit VISWORKPLN angegebene Bildschirm.**

Anmerkung (5): **Bei der Ausführung wird das Prozeßfenster 3 (VISWORKPLN 3) als Arbeitsbereich verwendet, und die Daten auf dem Prozeßfenster 3 sind nicht garantiert. Außerdem können Sie das Prozeßfenster 3 nicht für die Verarbeitung verwenden.**

Anmerkung (6): **Die folgenden Daten können mit der Funktion zum Abrufen des Verarbeitungsergebnisses abgerufen werden.**

	VISSTATUS (n)
n	Element
0	Ausführungsergebnis 0 = Normal -1 = Abnormal
1	Nummer
2	Ausführungszeit

	VISGETNUM (a, b)
b	a = 0~511
0	Nummer
1	X-Koordinatenwert
2	Y-Koordinatenwert
3	Unbekannt
4	Stufe
5	Unbekannt
6	Unbekannt
7	Unbekannt
8	Unbekannt
9	Unbekannt

Anmerkung (7): **Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.**

Verwandte Begriffe

WINDMAKE, VISWORKPLN, VISGETNUM, VISSTATUS, SHDEFCORNER

Beispiel

```
VISSCREEN 1,0,1             'Zeichnet sofort auf das Prozeßfenster 0.
VISCLS 0                    '
WINDMAKE R,1,512,480,0,2    'Stellt Fenster 1 auf Quader ein.
CAMIN 1                     'Ruft ein Kamerabild aus dem Speicher ab.
VISWORKPLN 0                'Gibt ein Objekt für Speicher 0 an.
SHCORNER 1,0,0,180,0        'Sucht nach einer schwarzen Ecke im linken unteren
                            'Abschnitt.
I1 = VISSTATUS(0)           '
IF I1 = 0 THEN              '
    I2 = VISSTATUS(1)       '
    IF I2 <> 0 THEN         '
        FOR I1 = 0 TO I2-1  '
            VISCROSS VISGETNUM(I1,1),VISGETNUM(I1,2)
                            '
        NEXT I1             '
    END IF                  '
END IF                      '
```

SHDEFCIRCLE (Anweisung)

Funktion

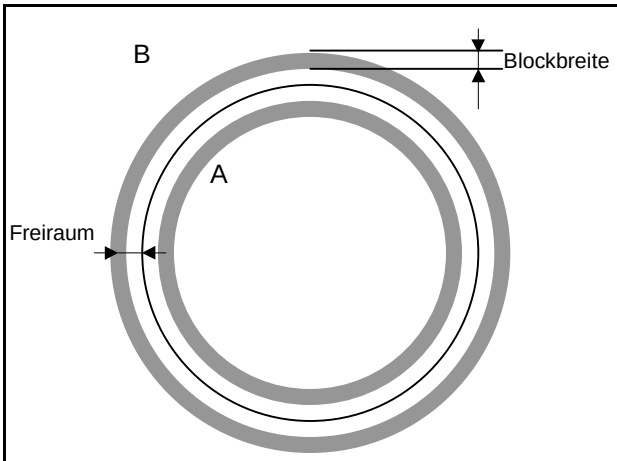
Legt die Bedingung zum Suchen eines Kreises fest.

Format

SHDEFCIRCLE <Freiraum>, <Breite>

Erklärung

<Freiraum> Gibt den Abstand zwischen den Blöcken bei der Kreissuche an (1 bis 10).
<Breite> Gibt die Breite der Blöcke bei der Kreissuche an (1 bis 10).



Ein Kreis wird mit der durchschnittlichen Helligkeit von 2 Blöcken erkannt, wie in der Abbildung links gezeigt.
Mit anderen Worten, wenn SHCIRCLE 0 ist, wird ein Kreis erkannt, wo der Unterschied zwischen der durchschnittlichen Helligkeit von A und der von B größer ist als der Unterschied in SHDEFCIRCLE.
Sie können den Block-Freiraum und die Breite mit SHDEFCIRCLE anpassen.

Anmerkung (1):	**Diese Anweisung entspricht einer temporären Einstellung; der Anfangswert wird nicht dauerhaft geändert.**
Anmerkung (2):	**Für diese Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.**
Anmerkung (3):	**Wenn ein großer Wert für den Freiraum angegeben wird, haben Störungen bei der Erkennung nur geringfügige Beeinträchtigungen zur Folge; die Genauigkeit der Erkennung ist jedoch geringer.**
Anmerkung (4):	**Die vor der Auslieferung ab Werk eingestellten Anfangswerte lauten wie folgt: Freiraum = 1 und Breite = 3**

Verwandte Begriffe

SHCIRCLE

Beispiel

```
SHDEFCIRCLE 1,3
```

SHCIRCLE (Anweisung)

Funktion

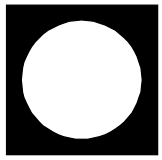
Sucht einen Kreis.

Format

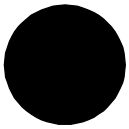
SHCIRCLE <Fensternummer>, <Koordinate X>, <Koordinate Y>, <Radius>, <Niveauunterschied>, <Modus>

Erklärung

<Fensternummer> Gibt die Fensternummer (0 bis 511) an.
<Koordinate X> Gibt die X-Koordinate (0 bis 511) an.
<Koordinate Y> Gibt die Y-Koordinate (0 bis 479) an.
<Radius> Gibt den Radius des zu suchenden Kreises (3 bis 240) an.
<Niveauunterschied> Gibt die Erkennungsstufe bei der Kreissuche an (0 bis 255).
<Modus> Gibt das Objekt für die Kreissuche an (0 oder 1).



Weißer Kreis (Modus = 0)



Schwarzer Kreis (Modus = 1)

Anmerkung (1): Geben Sie den Verarbeitungsbereich mit dem Fenster an.

Anmerkung (2): Wenn die angegebene Fensterposition außerhalb des Bildschirmes liegt, führt die Ausführung zu einem Fehler.

Anmerkung (3): Die einzige Fensterform, die Sie angeben können, ist ein Rechteck mit 0 Grad. Wenn Sie eine andere Form angeben, tritt ein Fehler auf.

Anmerkung (4): Das Verarbeitungsobjekt ist der mit VISWORKPLN angegebene Bildschirm.

Anmerkung (5): Bei der Ausführung wird das Prozeßfenster 3 (VISWORKPLN 3) als Arbeitsbereich verwendet, und die Daten auf dem Prozeßfenster 3 sind nicht garantiert. Außerdem können Sie das Prozeßfenster 3 nicht für die Verarbeitung verwenden.

Anmerkung (6): Die folgenden Daten können mit der Funktion zum Abrufen des Verarbeitungsergebnisses abgerufen werden.

	VISSTATUS (n)
n	Element
0	Ausführungsergebnis 0 = Normal -1 = Abnormal
1	Nummer
2	Ausführungszeit

	VISGETNUM (a, b)
b	a = 0~511
0	Nummer
1	X-Koordinate Wert
2	Y-Koordinate Wert
3	Unbekannt
4	Stufe
5	Unbekannt
6	Unbekannt
7	Unbekannt
8	Unbekannt
9	Unbekannt

Anmerkung (7): Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.

Anmerkung (8): Wenn der Suchbereich eines angegebenen Fensters breit und der Radius klein ist, erhöht sich die Verarbeitungszeit, und es kann ein Zeitüberschreitungsfehler auftreten.

Verwandte Begriffe

WINDMAKE, VISWORKPLN, VISGETNUM, VISSTATUS, SHDEFCIRCLE

Beispiel

```
VISSCREEN 1,0,1                'Zeichnet sofort auf das Prozeßfenster 0.
VISCLS 0                       '
WINDMAKE R,1,512,480,0,2       'Stellt Fenster 1 auf Rechteck ein.
CAMIN 1                        'Ruft ein Kamerabild aus dem Speicher ab.
VISWORKPLN 0                   'Gibt ein Objekt für Speicher 0 an.
SHCIRCLE 1,0,0,30,128,1        'Sucht nach einem schwarzen Kreis.
I1 = VISSTATUS(0)              '
VISLOC 10,10                   '
VISPRINT I1                    '
IF I1 = 0 THEN                 '
    I2 = VISSTATUS(1)          '
    IF I2 <> 0 THEN            '
        FOR I1 = 0 TO I2-1     '
            VISCROSS VISGETNUM(I1,1), VISGETNUM(I1,2)
                               '
        NEXT I1                '
    END IF                     '
END IF                         '
```

21.9 Ergebnisse abrufen

VISGETNUM (Funktion)

Funktion

Ruft ein Ergebnis einer Abbildverarbeitung aus dem Speicher ab.

Format

VISGETNUM(<Parameter 1>, <Parameter 2>)

Erklärung

<Parameter 1> Gibt die Nummer des abzurufenden Prozeßergebnisses an (0 bis 511).
<Parameter 2> Gibt den Typ des abzurufenden Prozeßergebnisses an (0 bis 9).

Anmerkung (1):	**Der Inhalt des gespeicherten Ergebnbisses ändert sich entsprechend der vorab ausgeführten Bildverarbeitungsanweisung, siehe Bildverarbeitungsanweisungen.**
Anmerkung (2):	**Das Ergebnis wird gespeichert, bis die nächste Bildverarbeitungsanweisung ausgeführt wird; Sie können daher das Ergebnis zu einem beliebigen Zeitpunkt abrufen.**
Anmerkung (3):	**Der Wert wird unbekannt, wenn Sie einen unbekannten Speicher mit dem vorab ausgeführten Bildverarbeitungsprozeß angeben.**
Anmerkung (4):	**Für diese Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.**

Verwandte Begriffe

VISMEASURE, VISPROJ, VISEDGE, VISREADQR, BLOB, BLOBMEASURE, SHMODEL, SHCORNER, SHCIRCLE, VISPOSX, VISPOSY

Beispiel

```
VISSCREEN 1,0,1                'Zeichnet sofort auf das Prozeßfenster 0.
VISCLS 0                       '
WINDMAKE R,1,512,480,0,2       'Stellt Fenster 1 auf Rechteck ein.
CAMIN 1                        'Ruft ein Kamerabild aus dem Speicher ab.
VISPLNOUT 0                    '
VISWORKPLN 0                   'Gibt ein Objekt für Speicher 0 an.
BLOB 1,0,0,0,128               'Führt die Kennzeichnung aus.
I1 = VISSTATUS(0)              '
IF I1 = 0 THEN                 '
    I2 = VISSTATUS(1)          '
    IF I2 <> 0 THEN            '
        FOR I1 = 0 TO 15       '
            VISDEFCHAR 1,1,0   '
            VISLOC 10,10+I1    '
            VISPRINT "X=";VISGETNUM(I1,1), VISGETNUM(I1,2)
                               '
        NEXT I1                '
    END IF                     '
END IF                         '
```

VISGETSTR (Funktion)

Funktion

Ruft das Ergebnis der Codeerkennung ab.

Format

VISGETSTR(<Nummer des führenden Zeichens>, <Anzahl der Zeichen>)

Erklärung

<Nummer des führenden Zeichens> Gibt die Nummer des abzurufenden führenden Zeichens an (1 bis 611).

<Anzahl der Zeichen> Gibt die Anzahl der abzurufenden Zeichen (1 bis 240) an.

Anmerkung (1): Wenn die Anzahl der abzurufenden Zeichen unbekannt ist, schlagen Sie im Prozeßergebnis zu der jeweiligen Anweisung nach.

Anmerkung (2): Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.

Verwandte Begriffe

VISREADQR, VISGETNUM

Beispiel

```
VISSCREEN 1,0,1                '
VISCLS 0                       '
WINDMAKE R,1,512,480,0,2       'Stellt Fenster 1 auf Rechteck ein.
CAMIN 1                        'Ruft ein Kamerabild aus dem Speicher ab.
VISPLNOUT 0                    '
VISREADQR 1,0,0,0              'Liest den QR-Code.
I1 = VISSTATUS(0)              '
VISPRINT I1, VISSTATUS(1)      '
IF I1 = 0 THEN                 '
    VISLOC 10,10               '
    VISPRINT VISGETSTR(1, VISSTATUS(1))
                               '
END IF                         '
VISCAMOUT 1                    '
```

VISPOSX (Funktion)

Funktion

Ruft ein Ergebnis einer Abbildverarbeitung (Koordinate X) aus dem Speicher ab.

Format

VISPOSX (<Parameter>)

Erklärung

<Parameter> Gibt die Nummer des abzurufenden Prozeßergebnisses (Koordinate X) an (0 bis 511).

Anmerkung (1):	**Dies entspricht der Einstellung des Parameters VISGETNUM auf den Wert 1. VISGETNUM (n, 2) = VISPOSY (n)**
Anmerkung (2):	**Die Prozeßergebnisse unterscheiden sich von den Ergebnissen, die mit früheren Bildverarbeitungsanweisungen erzielt wurden. Schlagen Sie unter den Bildverarbeitungsanweisungen nach.**
Anmerkung (3):	**Sie können wiederholt Ergebnisse abrufen, da diese bis zur Ausführung der nächsten Bildverarbeitungsanweisung beibehalten werden.**
Anmerkung (4):	**Wenn die vorherige Bildverarbeitungsanweisung eine unerwartete Speicherposition angibt, wird der Wert eine unbekannte Zahl.**
Anmerkung (5):	**Für diese Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.**

Verwandte Begriffe

VISMEASURE, VISPROJ, VISEDGE, VISREADQR, BLOB, BLOBMEASURE, SHMODEL, SHCORNER, SHCIRCLE, VISGETNUM, VISPOSY

Beispiel

```
VISSCREEN 1,0,1                'Zeichnet sofort auf das Prozeßfenster 0.
VISCLS 0                       '
WINDMAKE R,1,512,480,0,2       'Stellt Fenster 1 auf Rechteck ein.
CAMIN 1                        'Ruft ein Kamerabild aus dem Speicher ab.
VISPLNOUT 0                    '
VISWORKPLN 0                   'Gibt ein Objekt für Speicher 0 an.
BLOB 1,0,0,0,128               'Führt die Kennzeichnung aus.
I1 = VISSTATUS(0)              '
IF I1 = 0 THEN                 '
    I2 = VISSTATUS(1)          '
    IF I2 <> 0 THEN            '
        FOR I1 = 0 TO 15       '
            VISDEFCHAR 1,1,0   '
            VISLOC 10,10+I1    '
            VISPRINT "X=";VISPOSX(I1), "Y=";VISPOSY(I1)
                               '
            VISCROSS VISPOSX(I1), VISPOSY(I1)
                               '
         NEXT I1               '
    END IF                     '
END IF                         '
VISCAMOUT 1                    '
```

VISPOSY (Funktion)

Funktion

Ruft ein Ergebnis einer Abbildverarbeitung (Koordinate Y) aus dem Speicher ab.

Format

VISPOSY (<Parameter>)

Erklärung

<Parameter> Gibt die Nummer für das abzurufende Prozeßergebnis (Koordinate Y) an (0 bis 511). Ein abgerufener Wert wird als Konstante des Typs Real mit einfacher Genauigkeit (F-Typ) angegeben.

Anmerkung (1):	**Dies entspricht der Einstellung des Parameters 1 von VISGETNUM auf den Wert 2.** **VISGETNUM (n, 2) = VISPOSY (n)**
Anmerkung (2):	**Die Prozeßergebnisse unterscheiden sich von den Ergebnissen, die mit früheren Bildverarbeitungsanweisungen erzielt wurden. Schlagen Sie unter den Bildverarbeitungsanweisungen nach.**
Anmerkung (3):	**Sie können wiederholt Ergebnisse abrufen, da diese bis zur Ausführung der nächsten Bildverarbeitungsanweisung beibehalten werden.**
Anmerkung (4):	**Wenn die vorherige Bildverarbeitungsanweisung eine unerwartete Speicherposition angibt, wird der Wert eine unbekannte Zahl.**
Anmerkung (5):	**Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.**

Verwandte Begriffe

VISMEASURE, VISPROJ, VISEDGE, VISREADQR, BLOB, BLOBMEASURE, SHMODEL, SHCORNER, SHCIRCLE, VISGETNUM, VISPOSX

Beispiel

```
VISSCREEN 1,0,1                'Zeichnet sofort auf das Prozeßfenster 0.
VISCLS 0                       '
WINDMAKE R,1,512,480,0,2       'Stellt Fenster 1 auf Rechteck ein.
CAMIN 1                        'Ruft ein Kamerabild aus dem Speicher ab.
VISPLNOUT 0                    '
VISWORKPLN 0                   'Gibt ein Objekt für Speicher 0 an.
BLOB 1,0,0,0,128               'Führt die Kennzeichnung aus.
I1 = VISSTATUS(0)              '
IF I1 = 0 THEN                 '
    I2 = VISSTATUS(1)          '
    IF I2 <> 0 THEN            '
        FOR I1 = 0 TO 15       '
            VISDEFCHAR 1,1,0   '
            VISLOC 10,10+I1    '
            VISPRINT "X=";VISPOSX(I1), "Y=";VISPOSY(I1)
                               '
            VISCROSS VISPOSX(I1), VISPOSY(I1)
                               '
         NEXT I1               '
    END IF                     '
END IF                         '
VISCAMOUT 1                    '
```

VISSTATUS (Funktion)

Funktion

Überwacht das Verarbeitungsergebnis jeder Anweisung.

Format

VISSTATUS (<Parameter>)

Erklärung

<Parameter> Gibt die abzurufenden Daten an (0 bis 2).
0: Status der Ausführungsergebnisse
1: Hilfsdaten
2: Verarbeitungszeit

Ein abgerufener Wert wird als Konstante des Typs Real mit einfacher Genauigkeit (F-Typ) angegeben.

Anmerkung (1):	**Die abgerufenen Daten unterscheiden sich je nach den zuvor ausgeführten Anweisungen. Siehe die Erläuterung zu den einzelnen Anweisungen.**
Anmerkung (2):	**Alle Verarbeitungszeiten für Anweisungen werden zu Objekten. Sie können die Verarbeitungszeiten der zuvor ausgeführten Anweisungen abrufen.**
Anmerkung (3):	**Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.**

Verwandte Begriffe

VISMEASURE, VISPROJ, VISEDGE, VISREADQR, BLOB, BLOBMEASURE, SHMODEL, SHCORNER, SHCIRCLE

Beispiel

```
VISSCREEN 1,0,1                'Zeichnet sofort auf das Prozeßfenster 0.
VISCLS 0                       '
WINDMAKE R,1,512,480,0,2       'Stellt Fenster 1 auf Rechteck ein.
CAMIN 1                        'Ruft ein Kamerabild aus dem Speicher ab.
VISPLNOUT 0                    '
VISWORKPLN 0                   'Gibt ein Objekt für Speicher 0 an.
BLOB 1,0,0,0,128               'Führt die Kennzeichnung aus.
I1 = VISSTATUS(0)              '
I2 = VISSTATUS(1)              '
F1 = VISSTATUS(2)              '
VISLOC 10,9                    '
VISPRINT “Ausführungsergebnis=”;I1;“Kennzeichnungsnummer =”;I2, “Ausführungszeit
(Sekunden)=”;F1                '
IF I1 = 0 THEN                 '
    IF I2 <> 0 THEN            '
        FOR I1 = 0 TO I2 -1    '
            VISCROSS VISGETNUM(I1,1), VISGETNUM(I1,2)
                               '
        NEXT I1                '
    END IF                     '
END IF                         '
VISCAMOUT 1                    '
```

VISREFCAL (Funktion)

Funktion

Ruft Kalibrierungsdaten (Koordinatenumwandlung Bildverarbeitung-Roboter) ab.

Format

VISREFCAL (<Einstellnummer>, <Datennummer>)

Erklärung

<Einstellnummer> Gibt die Nummer der zu verwendenden Kalibrierungsdatengruppe an (0 bis 31).
<Datennummer> Gibt die Nummer der Kalibrierungsdaten an (0 bis 11).

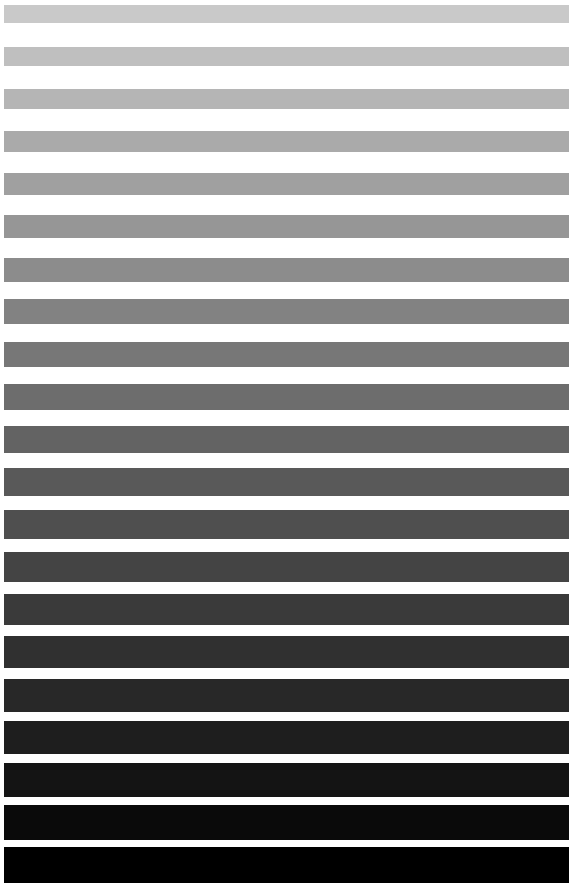
Anmerkung (1):	**Das System kann bis zu 32 Sets von Kalibrierungsdatengruppen speichern**
Anmerkung (2):	**Verwenden Sie einen PC zum Einstellen der Kalibrierungsdaten.**
Anmerkung (3):	**Zur Ausführung dieser Anweisung ist eine µBildverarbeitungskarte (Option) erforderlich.**

Beispiel

```
VISSCREEN 1,0,1                         'Zeichnet sofort auf Speicher 0.
VISCLS 0                                '
FOR I1 = 0 TO 11                        '
    VISLOC 10,10+I1                     '
    VISPRINT “Daten”;I1;“=”;VISREFCAL(0,I1)
                                        '
NEXT I1                                 '
```

Kapitel 22

PAC-Bibliothek



Dieses Kapitel enthält eine Erläuterung der PAC-Standardprogrammbibliothek.

Verwendung der Programmbibliothek

Zur Verwendung der Programmbibliothek müssen Sie eine Programmbank (Tool) im PAC Manager in WINCAPSII ausführen und die erforderlichen Bibliotheken installieren.
Information zur Programmbank, siehe im Benutzerhandbuch (WINCAPSII).

Krititerium der Programmbibliothek

Die Standardprogrammbibliothek ist in folgende sechs Kategorien unterteilt:

	Klassenname	Beschreibung
1	Konventionelle Sprache	Bietet Funktionen ähnlich den konventionellen Sprachbefehlen.
2	Palettierung	Bietet Palettierungsfunktionen.
3	Werkzeugbedienung	Bietet Funktionen zur Werkzeugbedienung.
4	Ein-/Ausgabe	Bietet Funktionen für DIO- und RS232C-Eingabe/ Ausgabefunktionen.
5	Armbewegung	Bietet Funktionen zur Armbewegung.
6	Bildverarbeitung	Bietet Funktionen zur Bildverarbeitung.
7	Kompatibel mit Ver. 1.2	Bietet die mit Version 1.2 kompatible Bibliothek, die in der Controller Software Version 1.2* oder früher verwendet werden kann. Diese Bibliothek enthält drei Programme: ndVcom, pltMove und pltMove0. Wenn in Version 1.2* oder früher eines dieser Programme verwendet wird, das nicht aus dieser Bibliothek, sondern aus den oben beschriebenen Kategorien 1 bis 6 stammt, führt dies zu einem Kompilierungsfehler. Verwenden Sie Bibliotheken in den oben beschriebenen Kategorien 1 bis 6 mit Ausnahme dieser drei Programme. ndVcom.pltMove.pltMove0

22.1 Konventionelle Sprache

aspACLD (Bibliothek)

Funktion

Ändert die internen Lastbedingungswerte. Zu den Nutzlastbedingungswerten gehören die Masse der Nutzlast in Gramm (g) und der Nutzlastschwerpunkt (Angabe in mm). Geben Sie beide Werte an.

Format

aspACLD (<Masse der Nutzlast>, <Nutzlastschwerpunkt der Koordinate X >, <Nutzlastschwerpunkt der Koordinate Y>, <Nutzlastschwerpunkt der Koordinate Z>)

Erklärung

Die Nutzlastmasse ist die Masse der Belastung (Endwerkzeug und Werkstück), die an der 6. Achse des Roboters angebracht wurde. Diese Einheit ist als (g) gekennzeichnet.
Geben Sie für den Ladeschwerpunkt den Nutzlastschwerpunkt mit den WERKZEUG0-Koordinaten an. Die Einheit lautet Millimeter (mm).
Die Referenzposition der Werkzeug-0-Koordinaten ist das Flanschzentrum der 6. Achse. Für die Komponente Y ist die Richtung vom Flanschzentrum zu der Bohrung von 6H7 (Richtung des Orientierungsvektors). Für die Komponente Z ist die Richtung vertikal zur Flanschfläche durch den Flanschmittelpunkt (Richtung des Anfahrvektors). Für die Komponente X ist die Richtung der X-Achse (Richtung des Normalvektors) im rechten Koordinatensystem, wenn der Orientierungsvektor auf die Y-Achse gesetzt ist und der Anfahrvektor auf die Z-Achse Weitere Informationen, siehe Abschnitt 4.7 “Einstellen der Master-Steuerparameter in den Benutzereinstellungen” in Kapitel 4 dieses Handbuches.

Auch wenn Sie einen der vier Werte für die Masse der Nutzlast ändern, beschreiben der Nutzlastschwerpunkt X_6, der Nutzlastschwerpunkt Y_6 und der Nutzlastschwerpunkt Z_6 alle 4 Werte erneut.

Die Umschaltung der Lastbedingungswerte dauert ca. 0,1 Sekunden. Ein häufiges Umschalten der Lastbedingung kann eine operative Verzögerung zur Folge haben. Ändern Sie den Modus während der Durchlaufbewegung nicht in der Nähe eines Hindernisses, da sich dadurch der Pfad verschieben könnte. **Dadurch kann sich die Umschaltung verzögern, wenn Sie die Werte für die Lastbedingung ändern.**

Makrodefinition

Erfordert die Datei <pacman.h>

Verwandte Begriffe

Siehe Abschnitt 4.7 “Einstellen der Master-Steuerparameter in den Benutzereinstellungen”.

Beispiel

```
CALL aspACLD(8500,-50,100,80)  'Stellt die internen Nutzlastbedingungen ein.
                               'Masse der Nutzlast:8500(g), Nutzlastschwerpunkt
                               'Komponente X: -50(mm), Komponente Y: 100(mm),
                               'Komponente Z: 80(mm)
```

Anmerkungen

(1) Geben Sie für die Masse der Nutzlast einen numerischen Wert in dem angegebenen Bereich für jeden Robotertyp ein. Wenn Sie einen Wert außerhalb dieses Bereiches eingeben, wird die Fehlermeldung “60d2. Der Nutzlastwert überschreitet den zulässigen Wert” angezeigt.

(2) Geben Sie den Nutzlastschwerpunkt so ein, dass er den angegebenen Bereich für jeden Robotertyp erfüllt. Liegt der Wert außerhalb dieses Bereiches, wird die Fehlermeldung “60d2. Der Nutzlastwert überschreitet den zulässigen Wert” angezeigt.

(3) Beachten Sie beim Einstellen der internen Nutzlastbedingung die folgende Regel in Relation zur externen Nutzlastbedingung. Falls Sie diese Regel verletzen, wird die Fehlermeldung “60d2 Der Nutzlastwert überschreitet den zulässigen Wert” angezeigt.

0,5 x Externe Nutzlastbedingung ≤ Interne Nutzlastbedingung ≤ Externe Nutzlastbedingung

aspChange (Bibliothek)

Funktion

Wählt den internen Modus für die richtige Steuereinstellung der Bewegungsoptimierung aus.

Format

aspChange (<Modus>)

Erklärung

Diese Anweisung wechselt den Modus für die Steuereinstellung der Bewegungsoptimierung.

<Einstellungswert>
0 → Ungültig
1 → Nur gültig für Punkt-zu-Punkt
2 → Nur gültig für Linear
3 → Gültig für Punkt-zu-Punkt und für Linear.

Die Umschaltung der Lastbedingungswerte dauert ca. 0,1 Sekunden. Ein häufiges Umschalten der Lastbedingung kann eine operative Verzögerung zur Folge haben. Ändern Sie den Modus während der Durchlaufbewegung nicht in der Nähe eines Hindernisses, da sich dadurch der Pfad verschieben könnte. Dadurch kann sich die Umschaltung verzögern, wenn Sie die Werte für die Lastbedingung ändern.

Makrodefinition

Erfordert die Datei <pacman.h>

Verwandte Begriffe

Weitere Informationen, siehe Abschnitt 4.7 “Einstellen der Master-Steuer-parameter in den Benutzereinstellungen” in Kapitel 4 dieses Handbuches.

Beispiel

```
CALL aspChange(1)          'Stellt den internen Modus in den Steuereinstellungen
                           'der Bewegungsoptimierung auf 1 ein.
```

Anmerkungen

Geben Sie für <Modus> einen numerischen Wert zwischen 0 und 3 ein. Wenn dieser Wert außerhalb des zulässigen Bereiches liegt, wird die Fehlermeldung “6003 Außerhalb des gültigen Bereiches” angezeigt.

ndInb (Bibliothek)

Funktion

Wandelt die Eingabe des angegebenen Anschlusses in eine Dezimalzahl um und behandelt sie wie eine Binärzahl.

Format

ndlnb (<Nummer der ganzzahligen Variablen>, <Am wenigsten signifikante Ziffer der Ausgabe-Anschlußnummer>, <Signifikanteste Ziffer der Ausgabe-Anschlußnummer>)

Erklärung

Diese Anweisung bietet eine ähnliche Funktion wie eine INB-Anweisung in der konventionellen Sprache.
Das System liest den Status eines angegebenen Eingabeanschlußsignals und wandelt ihn in eine Dezimalzahl um, indem es ihn als Binärzahl behandelt.
Das System wandelt einen umgewandelten Wert in die ganzzahlige Variable um.

Verwandte Begriffe

ndOnb, ndOnbl

Bemerkungen

Sie können die gleiche Funktion mit den Anweisungen DEFIO und IN in der PAC-Sprache definieren. Verwenden Sie die Anweisungen DEFIO und IN, da diese Methode effektiver ist.
Verwenden Sie für die Eingabeanschlüsse 16 Anschlüsse oder weniger in der Reihe. Wenn Sie mehr als 16 Anschlüsse angeben, wird keine Verarbeitung durchgeführt.

Beispiel

```
CALL ndInb(1,552,567)     'Liest den Status der Eingabeanschlüsse 552 bis 567 als
                          'eine 16-Bit Binärzahl, wandelt sie in eine Dezimalzahl
                          'um und weist sie der ganzzahligen Variablen (1) zu.
```

ndJf (Bibliothek)

Funktion

Eine bedingte Verzweigung nach dem Erhalten von OK/NG von einem externen Gerät (RS232C Eingabe/Ausgabe).

Format

ndJf (<2-Ziffern-Ganzzahl>, <bestimmendes Argument>)

Erklärung

Diese Anweisung bietet eine ähnliche Funktion wie eine JF-Anweisung in der konventionellen Sprache.
Das System überträgt eine 2-Ziffern-Ganzzahl an ein externes Gerät und berechnet ein Antwortergebnis zur Ausführung einer bedingten Verzweigung in dem Programm.
Wenn die Antwort von dem externen Gerät OK ist, fährt das System mit dem nächsten Schritt fort. Ist die Antwort nicht in Ordnung, verzweigt das System zu einem beschrifteten Schritt.

Verwandte Begriffe

ndVcom, ndVType

Beispiel

```
#include <Pacman.h>
PROGRAM PRO1
DEFINT JF_VAL = 0
     FLUSH
     CALL ndVType(1)         'Stellt das Protokoll (Konventionell = 0/ Neu = 1) ein.
     CALL ndVrst             'Initialisiert das externe Gerät.
     CALL ndVset(0)          'Löscht die empfangenen Daten (VDT) auf 0.
     CALL ndVis(3)           'Startet das externe Gerät (Sendet 03).
     CALL ndJf(3,JF_VAL)     'Erhält ein Antwortergebnis von dem externen Gerät
                             '(Sendet 03).
     IF JF_VAL = TRUE THEN   'Empfängt Daten, wenn die Antwort OK (TRUE) ist.
     CALL ndVset(3)          'Empfängt 10 Daten von dem externen Gerät (Sendet 03).
     CALL ndVdt(pacPOS,1)    'Ordnet die von dem externen Gerät empfangenen Daten
                             'der Variablen (P1) zu.
     CALL ndVdt(pacJNT,1)    'Ordnet die von dem externen Gerät empfangenen Daten
                             'der Variablen (J1) zu.
     CALL ndVdt(pacTRN,1)    'Ordnet die von dem externen Gerät empfangenen Daten
                             'der Variablen (T1) zu.
     END IF
END
```

ndOnb (Bibliothek)

Funktion

Wandelt eine Dezimalzahl in eine Binärzahl um und gibt sie an einen angegebenen Anschluß aus.

Format

ndOnb (<Ganzzahliger Wert>, <Am wenigsten signifikante Ziffer der Ausgabe-Anschlußnummer>, <Signifikanteste Ziffer der Ausgabe-Anschlußnummer>)

Erklärung

Diese Anweisung bietet eine ähnliche Funktion wie eine ONB-Anweisung in der konventionellen Sprache.
Das System wandelt eine Ganzzahl in eine Binärzahl um und gibt sie über einen angegebenen Anschluß aus.
Verwenden Sie die ndOnbI-Bibliothek für die ONB-Anweisung zum Formatieren der Ausgabe von Variablenwerten des Typs I.

Verwandte Begriffe

ndInb, ndOnbI

Bemerkungen

Verwenden Sie für die Ausgabeanschlüsse 16 Anschlüsse oder weniger in der Reihe. Wenn Sie mehr als 16 Anschlüsse angeben, wenn Anschluß niedrigerer Reihenfolge > Anschluß höherer Reihenfolge oder wenn Sie einen für die Ausgabe nicht verfügbaren Anschluß angeben, erfolgt keine Verarbeitung.

Beispiel

```
CALL ndOnb(15,769,784)      'Wandelt eine Dezimalzahl 15 in eine 16-Bit-
                            'Binärzahl um und gibt sie an die Ausgabeanschlüsse
                            '769 bis 784 aus.
```

ndOnbI (Bibliothek)

Funktion

Wandelt eine Dezimalzahl in eine Binärzahl um und gibt sie an einen angegebenen Anschluß aus.

Format

ndOnbI (<Nummer der ganzzahligen Variablen>, <Am wenigsten signifikante Ziffer der Ausgabe-Anschlußnummer>, <Signifikanteste Ziffer der Ausgabe-Anschlußnummer>)

Erklärung

Diese Anweisung bietet eine ähnliche Funktion wie eine ONB-Anweisung in der konventionellen Sprache.
Das System wandelt eine Ganzzahl in eine Binärzahl um und gibt sie über einen angegebenen Anschluß aus.
Verwenden Sie die ndOnb-Bibliothek für die ONB-Anweisung zum Formatieren der Ausgabe von Variablenwerten des Typs I.

Verwandte Begriffe

ndInb, ndOnb

Bemerkungen

Verwenden Sie für die Ausgabeanschlüsse 16 Anschlüsse oder weniger in der Reihe. Wenn Sie mehr als 16 Anschlüsse angeben, wenn Anschluß niedrigerer Reihenfolge > Anschluß höherer Reihenfolge oder wenn Sie einen für die Ausgabe nicht verfügbaren Anschluß angeben, erfolgt keine Verarbeitung.

Beispiel

```
CALL ndOnbI(1,769,784)      'Wandelt den Wert einer ganzzahligen Variablen I
                            '(Nummer 1) in eine 16-Bit-Binärzahl um und gibt sie
                            'an die Ausgabeanschlüsse 769 bis 784 aus.
```

ndVdt (Bibliothek)

Funktion

Ruft eine Variable ab, die von einem externen Gerät übertragen wurde (RS232C Eingabe/Ausgabe).

Format

ndVdt (<Speichervariablentyp>, <Speichervariablennummer>)

Erklärung

Diese Anweisung bietet eine ähnliche Funktion wie eine VDT-Anweisung in der konventionellen Sprache.
Das System weist die von einem externen Gerät übertragenen Daten einer Speichervariablennummer in einem angegebenen Variablentypformat (P/J/T) zu.

Makrodefinition

Erfordert die Datei <pacman.h>

Verwandte Begriffe

ndVcom, ndVset

Beispiel

```
#include <pacman.h>
PROGRAM PRO1
DEFINT JF_VAL = 0
  FLUSH
   CALL ndVType(1)          'Stellt das Protokoll (Konventionell = 0/ Neu = 1) ein.
   CALL ndVrst              'Initialisiert das externe Gerät.
   CALL ndVset(0)           'Löscht die empfangenen Daten (VDT) auf 0.
   CALL ndVis(3)            'Startet das externe Gerät (Sendet 03).
   CALL ndJf(3,JF_VAL)      'Erhält ein Antwortergebnis von dem externen
                            'Gerät (Sendet 03).
   IF JF_VAL = TRUE THEN    'Empfängt Daten, wenn die Antwort OK (TRUE) ist.
      CALL ndVset(3)        'Empfängt 10 Daten von dem externen Gerät (Sendet 03).
      CALL ndVdt(pacPOS,1)  'Ordnet die von dem externen Gerät empfangenen Daten
                            'der Variablen (P1) zu.
      CALL ndVdt(pacJNT,1)  'Ordnet die von dem externen Gerät empfangenen Daten
                            'der Variablen (J1) zu.
      CALL ndVdt(pacTRN,1)  'Ordnet die von einem externen Gerät empfangenen
                            'Daten der Variablen (T1) zu.
   END IF
END
```

ndVis (Bibliothek)

Funktion

Überträgt eine angegebene ganze Zahl mit 2 Ziffern an ein externes Gerät (RS232C Eingabe/Ausgabe).

Format

ndVis (<2-Ziffern-Ganzzahl>)

Erklärung

Diese Anweisung bietet eine ähnliche Funktion wie eine VIS-Anweisung in der konventionellen Sprache.
Der Roboter überträgt eine angegebene 2-Ziffern-Ganzzahl an ein externes Gerät, nachdem der Vorbereitungsstatus des externen Geräts geprüft wurde.

Verwandte Begriffe

ndVcom, ndVType

Beispiel

```
#include <pacman.h>
PROGRAM PRO1
DEFINT JF_VAL = 0
   FLUSH
   CALL ndVType(1)          'Stellt das Protokoll (Konventionell = 0/ Neu = 1) ein.
   CALL ndVrst              'Initialisiert das externe Gerät.
   CALL ndVset(0)           'Löscht die empfangenen Daten (VDT) auf 0.
   CALL ndVis(3)            'Startet das externe Gerät (Sendet 03).
   CALL ndJf(3,JF_VAL)      'Erhält ein Antwortergebnis von dem externen Gerät
                            '(Sendet 03).)
   IF JF_VAL = TRUE THEN    'Empfängt Daten, wenn die Antwort OK (TRUE) ist.
      CALL ndVset(3)        'Empfängt 10 Daten von dem externen Gerät (Sendet 03).
      CALL ndVdt(pacPOS,1)  'Ordnet die von dem externen Gerät empfangenen Daten
                            'der Variablen (P1) zu.
      CALL ndVdt(pacJNT,1)  'Ordnet die von dem externen Gerät empfangenen Daten
                            'der Variablen (J1) zu.
      CALL ndVdt(pacTRN,1)  'Ordnet die von dem externen Gerät empfangenen Daten
                            'der Variablen (T1) zu.
      END IF
END
```

ndVput (Bibliothek)

Funktion

Überträgt die Haltungsangabe an ein externes Gerät (RS232C Eingabe/ Ausgabe).

Format

ndVput (<Speichervariablentyp>, <Speichervariablennummer>)

Erklärung

Diese Anweisung bietet eine ähnliche Funktion wie eine VPUT-Anweisung in der konventionellen Sprache.
Wenn <Nummer der Positionsvariablen> negativ ist:
Das System überträgt die aktuellen Koordinaten des Roboters (die Koordinaten der Ausgangsposition im Werkzeugkoordinatensystem, wenn das Werkzeug definiert ist) und eine Haltung oder einen Inhalt der Positionsvariablen an ein externes Gerät.
Wenn <Nummer der Positionsvariablen> positiv ist:
Das System überträgt einen Speichervariablentyp und den Inhalt der Positionsvariablen, die mit einer Speichervariablennummer angegeben wurde, an das externe Gerät.

Makrodefinition

Erfordert die Datei <pacman.h>

Verwandte Begriffe

ndVcom, ndVType

Beispiel

```
#include <pacman.h>
PROGRAM PRO65
     FLUSH
     CALL   ndVType(1)         'Stellt das Protokoll (Konventionell = 0/
                               'Neu = 1) ein.
     CALL   nsVrst             'Initialisiert das externe Gerät.
     CALL   ndVis(4)           'Startet das externe Gerät (Sendet 03).
     P1 = (1,2,3,4,5,6,7)
     J1 = (11,12,13,14,15,16)
     T1 = (21,22,23,24,25,26,27,28,29,30)
     CALL   ndVput(pacPOS,1)   'Sendet die Daten der Variablen “P1”
     CALL   ndVput(pacJNT,1)   'Sendet die Daten der Variablen “J1”
     CALL   ndVput(pacTRN,1)   'Sendet die Daten der Variablen “T1”
END
```

ndVrst (Bibliothek)

Funktion

Initialisiert ein externes Gerät (RS232C Eingabe/Ausgabe).

Format

ndVrst

Erklärung

Diese Anweisung bietet eine ähnliche Funktion wie eine VRST-Anweisung in der konventionellen Sprache.
Außerdem gibt sie das zu initialisierende externe Gerät an.

Verwandte Begriffe

ndVcom, ndVType

Beispiel

```
#include <pacman.h>
PROGRAM PRO1
DEFINT JF_VAL = 0
   FLUSH
   CALL ndVType(1)            'Stellt das Protokoll (Konventionell = 0/
                              'Neu = 1) ein.
   CALL ndVrst                'Initialisiert das externe Gerät.
   CALL ndVset(0)             'Löscht die empfangenen Daten (VDT) auf 0.
   CALL ndVis(3)              'Startet das externe Gerät (Sendet 03).
   CALL ndJf(3,JF_VAL)        'Erhält ein Antwortergebnis von dem externen Gerät
                              '(Sendet 03).
   IF JF_VAL = TRUE THEN      'Empfängt Daten, wenn die Antwort OK (TRUE) ist.
      CALL  ndVset(3)         'Empfängt 10 Daten von dem externen Gerät (Sendet 03).
      CALL  ndVdt(pacPOS,1)   'Ordnet die von dem externen Gerät empfangenen Daten
                              'einer Variablen (P1) zu.
      CALL  ndVdt(pacJNT,1)   'Ordnet die von dem externen Gerät empfangenen Daten
                              'der Variablen (J1) zu.
      CALL  ndVdt(pacTRN,1)   'Ordnet die von dem externen Gerät empfangenen Daten
                              'der Variablen (T1) zu.
      END IF
END
```

ndVset (Bibliothek)

Funktion

Empfängt Daten von einem externen Gerät (RS232C Eingabe/Ausgabe).

Format

ndVset (<2-Ziffern-Ganzzahl>)

Erklärung

Diese Anweisung bietet eine ähnliche Funktion wie eine VSET-Anweisung in der konventionellen Sprache.
Das System empfängt Daten von einem externen Gerät und fügt (ordnet) sie einer internen Variablen zu, nachdem der Roboter eine angegebene 2-Ziffern-Ganzzahl an das externe Gerät gesendet hat. Wenn Sie eine <2-Ziffern-Ganzzahl> auf “0” angeben, initialisiert das System eine interne Variable mit “0”.

Verwandte Begriffe

ndVcom, ndVType

Beispiel

```
#include <pacman.h>
PROGRAM PRO1
DEFINTJF_VAL = 0
   FLUSH
   CALL ndVType(1)            'Stellt das Protokoll (Konventionell = 0/
                              'Neu = 1) ein.
   CALL ndVrst                'Initialisiert das externe Gerät.
   CALL ndVset(0)             'Löscht die empfangenen Daten (VDT) auf 0.
   CALL ndVis(3)              'Startet das externe Gerät (Sendet 03).
   CALL ndJf(3,JF_VAL)        'Erhält ein Antwortergebnis von dem externen Gerät
                              '(Sendet 03).
   IF JF_VAL = TRUE THEN      'Empfängt Daten, wenn die Antwort OK (TRUE) ist.
      CALL  ndVset(3)         'Empfängt 10 Daten von dem externen Gerät (Sendet 03).
      CALL  ndVdt(pacPOS,1)   'Ordnet die von dem externen Gerät empfangenen
                              'Daten der Variablen (P1) zu.
      CALL  ndVdt(pacJNT,1)   'Ordnet die von dem externen Gerät empfangenen
                              'Daten der Variablen (J1) zu.
      CALL  ndVdt(pacTRN,1)   'Ordnet die von dem externen Gerät empfangenen
                              'Daten der Variablen (T1) zu.
      END IF
END
```

ndVType (Bibliothek)

Funktion

Legt das Protokoll für die Kommunikation mit einem externen Gerät fest (RS232C Ausgabe).

Format

ndVType (<2-Ziffern-Ganzzahl>)

Erklärung

Diese Anweisung gibt das Kommunikationsprotokoll für ndVis, ndJf, ndVset, ndVrst, ndVput und ndVcom an. Der Anfangswert jeder Bibliothek wird im neuen Protokoll gesetzt.
(Altes Protokoll = 0/Neues Protokoll = 1)

Verwandte Begriffe

ndVis, ndJf, ndVset, ndVput, ndVcom

Beispiel

```
#include <pacman.h>
PROGRAM PRO1
DEFINTJF_VAL = 0
   FLUSH
   CALL ndVType(1)            'Stellt das Protokoll (Konventionell = 0/
                              'Neu = 1) ein.
   CALL ndVrst                'Initialisiert das externe Gerät.
   CALL ndVset(0)             'Löscht die empfangenen Daten (VDT) auf 0.
   CALL ndVis(3)              'Startet das externe Gerät (Sendet 03).
   CALL ndJf(3,JF_VAL)        'Erhält ein Antwortergebnis von dem externen Gerät
                              '(Sendet 03).
   IF JF_VAL = TRUE THEN      'Empfängt Daten, wenn die Antwort OK (TRUE) ist.
      CALL  ndVset(3)         'Empfängt 10 Daten von dem externen Gerät (Sendet 03).
      CALL  ndVdt(pacPOS,1)   'Ordnet die von dem externen Gerät empfangenen
                              'Daten der Variablen (P1) zu.
      CALL  ndVdt(pacJNT,1)   'Ordnet die von dem externen Gerät empfangenen
                              'Daten der Variablen (J1) zu.
      CALL  ndVdt(pacTRN,1)   'Ordnet die von dem externen Gerät empfangenen
                              'Daten der Variablen (T1) zu.
      END IF
END
```

ndVcom (Bibliothek)

Funktion

Kommunikation mit einem externen Gerät (Kernel) (RS232C Eingabe/ Ausgabe)

Format

ndVcom (<Funktionscode>, <Array-Variable>)

Erklärung

Dies ist ein Kernel-Programm für die Kommunikation mit einem externen Gerät. Es ist zur Verwendung der Kommunikationsbibliotheken (ndVis, ndJf, ndVset, ndVrst, ndVput und ndVType) erforderlich.

Verwenden Sie für das Controller System Version 1.2* oder früher die Komponente ndVcom; Sie finden diese Komponente in der zu Version 1.2 kompatiblen Klasse. Wenn in Version 1.2* oder früher ndVcom verwendet wird, das nicht aus zu Version 1.2 kompatiblen Klasse, sondern aus der Klasse 1 bis 6 stammt, tritt ein Kompilierungsfehler auf.

Weitere Informationen zur Überprüfung der Version der Controller-Software, siehe Abschnitt 5.7 [F6 Set]–[F6 Maint.]–[F2 Version] in Kapitel 5 des Installations- und Wartungshandbuches.

Anmerkung:	**Dieses Programm wird durch den Multitasking-Betrieb nicht automatisch ausgeführt.** **Wenn Sie dieses Programm über den Multitasking-Betrieb ausführen, müssen Sie es für die Synchronisierung mit einem Semaphor etc. anpassen.**

Verwandte Begriffe

ndVis, ndJf, ndVset, ndVrst, ndVput, ndVdt, ndVType

ndApra (Bibliothek)

Funktion

Führt eine absolute Operation mit dem angegebenen Werkzeugkoordinatensystem aus (ausschließlich für Roboter mit 4 Achsen).

Format

ndApra (<Referenzposition>, <Z-Koordinate>, <Stopgenauigkeit>)

Erklärung

Diese Anweisung bietet eine ähnliche Funktion wie eine APRA-Anweisung in der konventionellen Sprache.

Führt nur durch Angabe der Z-Koordinate der Referenzposition eine Punkt-zu-Punkt-Bewegung aus. Die Operation unterscheidet sich je nach angegebener Stopgenauigkeit wie folgt:

0: Endeoperation

1: Pfadoperation

2: Überprüfen der Ankunft an der Zielposition über den Codiererwert

Zur Verwendung einer Funktion ähnlich der APRT-Anweisung in der konventionellen Sprache berechnen Sie den Wert des Positionstypes durch Umwandeln der Abweichungsberechnung (siehe Abschnitt 7.9.7 “Positions-Operation” in Kapitel 7 dieses Handbuches); verweisen Sie hierzu auf diese Bibliothek und verwenden Sie die MOVE-Anweisung.

Verwandte Begriffe

ndDepa, ndAprt, ndDrwt

Beispiel

```
PROGRAM PRO1
  TAKEARM
  CALL  ndApra((P0),400.0,1) 'Führt eine Bewegung nach 400 in Z-Koordinatenrichtung
      'an XP0 durch.
  MOVE P,@0 P0               'Bewegung nach P0.
  CALL  ndDepa(400.0,0)      'Bewegung nach 400 in Z-Koordinatenrichtung an P0.
  GIVEARM
END
```

ndDepa (Bibliothek)

Funktion

Führt eine absolute Operation mit dem angegebenen Werkzeugkoordinatensystem aus (ausschließlich für Roboter mit 4 Achsen).

Format

ndDepa (<Z-Koordinate>, <Stopgenauigkeit>)

Erklärung

Diese Anweisung bietet eine ähnliche Funktion wie eine DEPA-Funktion in der konventionellen Sprache.

Führt nur durch Angabe der Z-Koordinate eine Punkt-zu-Punkt-Bewegung von der aktuellen Position aus. Die Operation unterscheidet sich je nach Stopgenauigkeit wie folgt:

0: Endeoperation

1: Pfadoperation

2: Überprüfen der Ankunft an der Zielposition über den Codiererwert

Zur Verwendung einer Funktion ähnlich der DRET-Anweisung in der konventionellen Sprache berechnen Sie den Wert des Positionstypes durch Umwandeln der Abweichungsberechnung (siehe Abschnitt 7.9.7 “Positions - Operation” in Kapitel 7 dieses Handbuches); verweisen Sie hierzu auf diese Bibliothek und verwenden Sie die MOVE-Anweisung.

Verwandte Begriffe

ndApra

Beispiel

```
PROGRAM PRO1
  TAKEARM
  CALL  ndApra((P0),400.0,1) ' Bewegung nach 400 in Z-Koordinatenrichtung
                             ' an XP0.
  MOVE P,@0 P0               ' Bewegung nach P0.
  CALL  ndDepa(400.0,0)      ' Bewegung nach 400 in Z-Koordinatenrichtung an P0.
  GIVEARM
END
```

22.2 Palettierung

pltDecCnt (Bibliothek)

Funktion

Verringert die Zahl im Palettierungsgesamtzähler.

Format

pltDecCnt (<Palettierungsnummer>)

Erklärung

Der Gesamtzähler für die Palettierung, angegeben mit einer Palettierungsnummer, wird um eins verringert (-1).
Durch die Bearbeitung des Palettierungszählers können Sie ein Sägezahnmuster oder ein Zufallsschrittmuster palettieren.

Verwandte Begriffe

pltGetCnt, pltIncCnt, pltKernel, pltLetCnt

Beispiel

```
CALL pltDecCnt(1)           'Verringert den Gesamtzähler für die Palettierung
                            'Nr. 1 um eins (-1).
```

pltGetCnt (Bibliothek)

Funktion

Ruft den Inhalt des Palettierungsgesamtzählers ab.

Format

pltGetCnt (<Palettierungsnummer>, <Cnt speichern ganzzahlige Variablennummer>)

Erklärung

Ruft den Gesamtzähler für die Palettierung einer angegebenen Palettierungsnummer auf.

Verwandte Begriffe

pltDecCnt, pltIncCnt, pltKernel, pltLetCnt

Beispiel

```
CALL pltGetCnt(1, 0)        'Ordnet dem aktuellen Zähler I[0] zu.
CALL pltLetCnt(1, I[0] - 1) 'Verringert den Zähler um 1.
```

pltGetK (Bibliothek)

Funktion

Ruft den Wert K des Palettierungssatzes ab.

Format

pltGetK (<Palettierungsnummer>, <K speichern ganzzahlige Variablennummer>)

Erklärung

Diese Anweisung ruft den Palettierungs-Set-Wert K einer angegebenen Palettierungsnummer auf.

Verwandte Begriffe

pltGetK1, pltLetK, pltKernel

Beispiel

```
CALL pltGetK(1,10)          'Ordnet den Palettierungswert K der
                            'Palettierungsnummer 1 der ganzzahligen Variablen
                            '10 zu.
```

pltGetK1 (Bibliothek)

Funktion

Ruft den Inhalt des Palettierungszählers K1 ab.

Format

pltGetK1 (<Palettierungsnummer>, <K1 speichern ganzzahlige Variablen-nummer>)

Erklärung

Diese Anweisung ruft den Palettierungszähler K1 einer angegebenen Palettierungsnummer auf.

Verwandte Begriffe

pltGetK, pltLetK1, pltKernel

Beispiel

```
CALL pltGetK1(1,10)          'Ordnet den Palettierungszähler K1 der
                             'Palettierungsnummer 1 der ganzzahligen
                             'Variablen 10 zu.
```

pltGetM (Bibliothek)

Funktion

Ruft den Wert M des Palettierungssatzes ab.

Format

pltGetM (<Palettierungsnummer>, <M speichern ganzzahlige Variablennummer>)

Erklärung

Diese Anweisung ruft den Palettierungs-Set-Wert M einer angegebenen Palettierungsnummer auf.

Verwandte Begriffe

pltGetM1, pltLetM, pltKernel

Beispiel

```
CALL pltGetM(1,10)          'Ordnet den Palettierungswert M der
                            'Palettierungsnummer 1 der ganzzahligen
                            'Variablen 10 zu.
```

pltGetM1 (Bibliothek)

Funktion

Ruft den Inhalt des Palettierungszählers M1 ab.

Format

pltGetM1 (<Palettierungsnummer>, <M1 speichern ganzzahlige Variablen-nummer>)

Erklärung

Diese Anweisung ruft den Palettierungszähler M1 einer angegebenen Palettierungsnummer auf.

Verwandte Begriffe

pltGetM, pltLetM1, pltKernel

Beispiel

```
CALL pltGetM1(1,10)          'Ordnet den Palettierungszähler M1 der
                             'Palettierungsnummer 1 der ganzzahligen
                             'Variablen 10 zu.
```

pltGetN (Bibliothek)

Funktion

Ruft den Wert N des Palettierungssatzes ab.

Format

pltGetN (<Palettierungsnummer>, <N speichern ganzzahlige Variablen-nummer>)

Erklärung

Diese Anweisung ruft den Palettierungs-Set-Wert N einer angegebenen Palettierungsnummer auf.

Verwandte Begriffe

pltGetN1, pltLetN, pltKernel

Beispiel

```
CALL pltGetN(1,10)          'Ordnet den Palettierungswert N der
                            'Palettierungsnummer 1 der ganzzahligen
                            'Variablen 10 zu.
```

pltGetN1 (Bibliothek)

Funktion

Ruft den Inhalt des Palettierungszählers N1 ab.

Format

pltGetN1 (<Palettierungsnummer>, <N1 speichern ganzzahlige Variablen-nummer>)

Erklärung

Ruft den Zähler N1 einer angegebenen Palettierungsnummer auf.

Verwandte Begriffe

pltGetN, pltLetN1, pltKernel

Beispiel

```
CALL pltGetN1(1,10)          'Ordnet den Palettierungszähler N1 der
                             'Palettierungsnummer 1 der ganzzahligen
                             'Variablen 10 zu.
```

pltGetNextPos (Bibliothek)

Funktion

Ruft die nächste Position ab.

Format

pltGetNextPos (<Palettierungsnummer>, <Nächste Position Abrufen der Positionstypvariablen>)

Erklärung

Diese Anweisung ruft die nächste Position ab. Bei der Ausführung dieser Bibliothek wird der Palettierungszähler erhöht.
Wenn Sie daher die nächste Position abrufen, aber nicht ausführen, müssen Sie den pltDecCnt etc. reduzieren.

Verwandte Begriffe

pltMove, pltKernel

Beispiel

```
CALL pltGetNextPos(1,10)     'Ordnet eine Position, die unmmittelbar der
                             'Palettierungsnummer 1 folgt, der ganzzahligen
                             'Variable 10 zu.
```

pltGetPLT1END (Bibliothek)

Funktion

Ruft eine Markierung “Palettierung 1. Zeile” ab.

Format

pltGetPLT1END (<Palettierungsnummer>, <1. Zeile Abschlußmarkierung speichern ganzzahlige Nummer>)

Erklärung

Speichert eine “Palettierung 1. Zeile” Abschlußmarkierung in I[<1. Zeile Abschlußmarkierung speichern ganzzahlige Nummer>].

Verwandte Begriffe

pltGetPLTEND, pltResetPLT1END, pltKernel

Beispiel

```
CALL pltGetPLT1END(1,10)     'Ordnet eine 1. Zeile Abschlußmarkierung
                             'der Palettierungsnummer 1 der ganzzahligen
                             'Variablen 10 zu.
```

pltGetPLTEND (Bibliothek)

Funktion

Ruft eine Markierung “Palettierung alle Zeilen” ab.

Format

pltGetPLTEND (<Palettierungsnummer>, <Alle Zeilen-Abschlußmarkierung speichern ganzzahlige Nummer>)

Erklärung

Diese Anweisung speichert eine “Palettierung Alle-Zeilen”-Abschlußmarkierung in I [<Alle-Zeilen-Abschlußmarkierung speichern ganzzahlige Nummer>].

Verwandte Begriffe

pltGetPLT1END, pltResetPLTEND, pltKernel

Beispiel

```
CALL pltGetPLTEND(1,10)      'Ordnet eine "Alle-Zeilen-Abschlußmarkierung"
                             'der Palettierungsnummer 1 der ganzzahligen
                             'Variablen 10 zu.
```

pltIncCnt (Bibliothek)

Funktion

Erhöht den Inhalt des Palettierungsgesamtzählers.

Format

pltIncCnt (<Palettierungsnummer>)

Erklärung

Diese Anweisung vergrößert den Palettierungsgesamtzähler einer angegebenen Palettierungsnummer um +1, ausgehend vom aktuellen Wert.
Durch die Bearbeitung des Palettierungszählers können Sie ein Sägezahnmuster oder ein Zufallsschrittmuster palettieren.

Verwandte Begriffe

pltDecCnt, pltGetCnt, pltKernel, pltLetCnt

Beispiel

```
CALL pltIncCnt(1)           'Vergrößert den Gesamtzähler der Palettierung von
                            'Palettierungsnummer 1 um eins (+1).
```

pltInit1 (Bibliothek)

Funktion

Palettierungsinitialisierung Schablone 1.

Format

pltInit1

Erklärung

Dies ist ein Beispiel für pltInitialize.

Die Bedeutung der Argumente ist bei der Beschreibung der Argumente für die pltInitialize-Befehle beschrieben.

Verwandte Begriffe

pltInitialize

pltInitialize (Bibliothek)

Funktion

Palettierungsinitialisierung

Format

pltInitialize (<Palettierungsnummer>, <Querrichtungs-Partitionierungsnummer>, <Längsrichtungs-Partitionierungsnummer>, <Stapelnummer>, <Annäherungslänge>, <Fortbewegungslänge>, <Palettenhöhe>, <Palette 4-Ecken-Typ 1P Variablennummer>, <Palette 4-Ecken-Typ 2P Variablennummer>, <Palette 4-Ecken-Typ 3P Variablennummer>, <Palette 4-Ecken-Typ 4P Variablennummer>)

Erklärung

Diese Anweisung ist ein Initialisierungsprogramm für die Definition der Palettierungsbewegung.
Die Verwendung dieses Programmes ist effektiver als die Bibliothek pltResetAll, falls Sie nur den Palettierungszähler initialisieren.

Anmerkung: Ohne die Ausführung von pltInitialize wird keine Palettierungsbewegung ausgeführt.

Makrodefinition

mcApprVal : Variablennummer des Types F zur Zuordnung der Annäherungslänge.
McDepVal : Variablennummer des Types F zur Zuordnung der Fortbewegungslänge.
McPltH : Eine Variablennummer des Types F zur Zuordnung der Palettenhöhe.

Anmerkung: Wenn diese Definitionen einen Konflikt mit der Variablennummer des Benutzers ergeben, definieren Sie den Makro erneut.

Verwandte Begriffe

pltKernel

Beispiel

```
CALL pltInitialize(0,4,3,1,50,50,50,52,53,54,55)
        'Initialisiert die Palettierungsparameter; Palettierungsnummer:0,
        'Querrichtungs-Partitionierungsnummer:4, Längsrichtungs-Partitionierungsnummer:3
        'Stapelnummer:1, Annäherungslänge:50 mm, Fortbewegungslänge:50 mm,
        'Palettenhöhe:50 mm, Palette 4-Ecken-Typ 1P:52, 2P:53,
        '3P:54, 4P:55.
```

pltKernel (Bibliothek)

Funktion

Palettierungsbewegung (Kernel)

Format

pltKernel (<Partitionierungsnummer>, <Aktionsnummer>, <Aktionsargument>, <Fehlernummer>)

Erklärung

Diese Anweisung ist ein Kernel-Programm für die Definition der Palettierungsbewegung. Diese Bibliothek ist zur Verwendung aller anderen Palettierungsbibliotheken erforderlich.

Anmerkung:	**Dieses Programm wird durch den Multitasking-Betrieb nicht automatisch ausgeführt. Wenn Sie dieses Programm über den Multitasking-Betrieb ausführen, müssen Sie es für die Synchronisierung mit einem Semaphor anpassen.**

Makrodefinition

mcPaltMax: Maximale Palettierungsnummer
Sie können die Anzahl der Paletten leicht erhöhen, indem Sie diesen Makro erneut definieren.

Anmerkung:	**Wenn Sie die Anzahl der Paletten erhöhen, verbraucht das System den Bereich der lokalen Variablen.**

Verwandte Begriffe

pltInitialize

Bemerkungen

- Diese Bibliothek wird nur von anderen Bibliotheken aufgerufen. Der Benutzer ruft diese Bibliothek nicht direkt auf.
- Zur direkten Verwendung dieser Bibliothek sollten Sie das Programm vollständig verstanden haben.

pltLetCnt (Bibliothek)

Funktion

Definiert den Inhalt des Palettierungsgesamtzählers.

Format

pltLetCnt (<Palettierungsnummer>, <Alle Zähler-Einstellungswerte>)

Erklärung

Diese Anweisung kann den Gesamtzähler für die Palettierung einer angegebenen Palettierungsnummer einstellen.
Durch die Bearbeitung des Palettierungszählers können Sie ein Sägezahnmuster oder ein Zufallsschrittmuster palettieren.

Verwandte Begriffe

pltDecCnt, pltGetCnt, pltIncCnt, pltKernel

Beispiel

```
CALL pltLetCnt(1,12)        'Stellt den Gesamtzähler der Palettierung von
                            'Nr. 1 bis 12 ein.
```

pltLetK1 (Bibliothek)

Funktion

Definiert den Inhalt des Palettierungszählers K1.

Format

pltLetK1 (<Palettierungsnummer>, <K1-Einstellungswert>)

Erklärung

Diese Anweisung kann den Zähler K1 einer angegebenen Palettierungsnummer einstellen.
Durch die Bearbeitung des Palettierungszählers können Sie ein Sägezahnmuster oder ein Zufallsschrittmuster palettieren.

Verwandte Begriffe

pltGetK1, pltKernel

Beispiel

```
CALL pltLetK1(1,1)     'Stellt den Palettierungszähler K1 der Palettierung 1 auf
'1 ein.
```

pltLetM1 (Bibliothek)

Funktion

Definiert den Inhalt des Palettierungszählers M1.

Format

pltLetM1 (<Palettierungsnummer>, <M1-Einstellungswert>)

Erklärung

Diese Anweisung kann den Zähler M1 einer angegebenen Palettierungsnummer einstellen.
Durch die Bearbeitung des Palettierungszählers können Sie ein Sägezahnmuster oder ein Zufallsschrittmuster palettieren.

Verwandte Begriffe

pltGetM1, pltKernel

Beispiel

```
CALL pltLetM1(1,4)     'Stellt den Palettierungszähler M1 der Palettierung 1 auf
'4 ein.
```

pltLetN1 (Bibliothek)

Funktion

Definiert den Inhalt des Palettierungszählers N1.

Format

pltLetN1 (<Palettierungsnummer>, <N1-Einstellungswert>)

Erklärung

Diese Anweisung kann den Zähler N1 einer angegebenen Palettierungsnummer einstellen.
Durch die Bearbeitung des Palettierungszählers können Sie ein Sägezahnmuster oder ein Zufallsschrittmuster palettieren.

Verwandte Begriffe

pltGetN1, pltKernel

Beispiel

```
CALL pltLetN1(1,3)          'Stellt den Palettierungszähler N1 der
                            'Palettierungsnummer 1 auf 3 ein.
```

pltMain1 (Bibliothek)

Funktion

Palettierung Schablone 1

Format

pltMain1

Erklärung

Diese Anweisung ist die Basis-Arbeitsanweisung zum Aufnehmen eines Teils und zum Verschieben des Teils an die Konstruktionsposition.

Makrodefinition

pltIndex: Palettierungsnummer

Verwandte Begriffe

pltMain2

Bemerkungen

Sie können die Schablone an Ihre Anforderungen anpassen.

pltMain2 (Bibliothek)

Funktion

Palettierung Schablone 2

Format

pltMain2

Erklärung

Diese Anweisung ist die Basis-Arbeitsanweisung zum Aufnehmen eines Teils und zum Verschieben des Teils an die Konstruktionsposition.
Vor dem Aufnehmen eines Teils prüft der Roboter, ob der vorangegangene Aufnahmevorgang korrekt durchgeführt wurde.

Makrodefinition

pltIndex : Palettierungsnummer
ChuckNG : NG-Aufnahmesignal (DIO-Nummer)

Verwandte Begriffe

pltMain1

Bemerkungen

Sie können die Schablone an Ihre Anforderungen anpassen.

pltMove (Bibliothek)

Funktion

Standardpalettierung Schablone 1

Format

pltMove (<Palettierungsnummer>)

Erklärung

Diese Anweisung führt eine Standardpalettierung aus, wie mit pltInitialize definiert.
Sie können entsprechend den Kommentaren einen Umgehungspunkt einfügen.

Verwenden Sie für das Controller System Version 1.2* oder früher die Komponente ndVcom; Sie finden diese Komponente in der zu Version 1.2 kompatiblen Klasse. Wenn in Version 1.2* oder früher ndVcom verwendet wird, das nicht aus zu Version 1.2 kompatiblen Klasse, sondern aus der Klasse 1 bis 6 stammt, tritt ein Kompilierungsfehler auf.

Weitere Informationen zur Überprüfung der Version der Controller-Software, siehe Abschnitt 5.7 in Kapitel 5 des INSTALLATIONS- UND WARTUNGSHANDBUCHES.

Makrodefinition

mcNextPos : Eine Variablennummer des Types P zum Speichern der nächsten Position
mcApprLen : Variablennummer des Types F zum Speichern der Annäherungslänge
mcDepLen : Variablennummer des Types F zum Speichern der Fortbewegungslänge

Anmerkung: Wenn diese Definitionen einen Konflikt mit der Variablennummer des Benutzers ergeben, definieren Sie den Makro erneut.

Verwandte Begriffe

pltInitialize, pltMove0, pltKernel

Bemerkungen

Der Palettierungszähler zählt, wenn das System die nächste Position abruft. Wenn das System daher Daten abruft, aber keine Operation ausführt, müssen Sie den Zähler mit pltDecCnt verringern.

Beispiel

```
CALL pltMove(1)         'Führt eine Palettierungsbewegung der
                        'Palettierungsnummer 1 aus.
```

pltMove0 (Bibliothek)

Funktion

Standardpalettierung Bewegung 1

Format

pltMove0

Erklärung

Diese Anweisung ist ein spezielles Programm für Index=0 von pltMove (Index).

Wenn Sie die Bewegung jeder Palette ändern, fügen Sie dieses Programm hinzu, um die Zeile #DEFINE pltIndex 0 auf die Palettierungsnummer zu ändern und anschließend den Programmnamen zu ändern.

Verwenden Sie für das Controller System Version 1.2* oder früher ndVcom; Sie finden diese Komponente in der zu Version 1.2 kompatiblen Klasse. Wenn in Version 1.2* oder früher ndVcom verwendet wird, das nicht aus der zu Version 1.2 kompatiblen Klasse, sondern aus der Klasse 1 bis 6 stammt, tritt ein Kompilierungsfehler auf.

Zur Überprüfung der Version der Controller-Software schlagen Sie im Installationshandbuch und Kapitel 5, Abschnitt 5.7, [F6 Set]–[F6 Maint.]–[F2 Version] nach.

Makrodefinition

mcNextPos : Eine Variablennummer des Types P zum Speichern der nächsten Position
mcApprLen : Variablennummer des Types F zum Speichern der Annäherungslänge
mcDepLen : Variablennummer des Types F zum Speichern der Fortbewegungslänge

Anmerkung:	**Wenn diese Definitionen einen Konflikt mit der Variablennummer des Benutzers ergeben, definieren Sie den Makro erneut.**

PltIndex : Palettennummer

Verwandte Begriffe

pltInitialize, pltMove, pltKernel

Bemerkungen

Der Palettierungszähler zählt, wenn das System die nächste Position abruft. Wenn das System daher Daten abruft, aber keine Operation ausführt, müssen Sie den Zähler mit pltDecCnt verringern.

Beispiel

```
CALL pltMove0          'Führt eine Palettierungsbewegung der
                       'Palettierungsnummer 0 aus.
```

pltResetAll (Bibliothek)

Funktion

Setzt alle Palettierungszähler zurück.

Format

pltResetAll (<Palettierungsnummer>)

Erklärung

Diese Anweisung setzt alle Palettierungszähler zurück.
Dies hat fast die gleiche Bedeutung wie pltLetCnt (<Palettierungsnummer>, 0). Diese Bibliothek setzt jedoch auch die Markierung “Zeile abgeschlossen” zurück.
Wenn eine Neudefinition der Palettierungsbewegung nicht erforderlich ist und nur ein Zähler zurückgesetzt wird, verwenden Sie diese Bibliothek. Verwenden Sie in diesem Fall nicht pltInitialize.

Verwandte Begriffe

pltInitialize, pltResetPLTEND, pltKernel

Beispiel

```
CALL pltResetAll(1)    'Setzt alle Palettierungszähler von
                       'Palettierungsnummer 1 zurück.
```

pltResetPLT1END (Bibliothek)

Funktion

Setzt eine Markierung Palettierung 1. Zeile zurück.

Format

pltResetPLT1END (<Palettierungsnummer>)

Erklärung

Diese Anweisung setzt eine Markierung “Palettierung 1. Zeile abgeschlossen” zurück.

Verwandte Begriffe

pltGetPLT1END, pltResetPLTEND, pltKernel

Beispiel

```
CALL pltResetPLT1END(1)      'Setzt eine Markierung "Palettierung 1. Zeile
                             'abgeschlossen" von Palettierungsnummer 1 zurück.
```

pltResetPLTEND1 (Bibliothek)

Funktion

Setzt eine Markierung “Palettierung alle Zeilen” zurück.

Format

pltResetPLTEND1 (<Palettierungsnummer>)

Erklärung

Diese Anweisung setzt eine Markierung “Palettierung alle Zeilen abgeschlossen” zurück.

Verwandte Begriffe

pltGetPLTEND, pltResetPLT1END, pltKernel

Beispiel

```
CALL pltResetPLT1END1(1)     'Setzt eine Markierung "Palettierung alle Zeilen
                             'abgeschlossen" von Palettierungsnummer 1 zurück.
```

22.3 Werkzeugoperation

tolChange (Bibliothek)

Funktion

Werkzeugänderung

Format

tolChange (<Werkzeugladenummer>, <Werkzeugoperation>)

Erklärung

Diese Bibliothek führt die Standard-Tool-Bewegung aus, wie in tolInitialize definiert.

Verwandte Begriffe

tolInitialize, tolKernel

Bemerkungen

Die Änderungsanweisung zum Werkzeugkoordinatensystem in PAC (ChangeTool) bewirkt keine tatsächliche Änderung der Werkzeugposition und kein tatsächliches Laden des Werkzeugs, sondern nur die Änderung des Werkzeugkoordinatensystems.

Beispiel

```
CALL tolChange(1,2)          'Ordnet Werkzeug Nr.2 der Werkzeugladenummer 1 zu.
```

tolInit1 (Bibliothek)

Funktion

Werkzeugänderung Initialisierung Schablone 1

Format

tolInit1

Erklärung

Dies ist ein Beispiel für tolInitialize.
Die Bedeutung der Argumente ist bei der Beschreibung der Argumente für die tolInitialize-Befehle beschrieben.

Verwandte Begriffe

tolInitialize, tolKernel

tolInitialize (Bibliothek)

Funktion

Werkzeugänderung Initialisierung

Format

tolInitialize (<Werkzeugladenummer>, <Werkzeugnummer>, <Variablen-nummer Einspannpunkt Typ P>, <Annäherungslänge>, <Einspann-DIO-Nummer>, <Loslassen-DIO-Nummer>)

Erklärung

Diese Anweisung definiert die Standardänderung für die Werkzeugbewegung.

Verwandte Begriffe

tolInit1, tolKernel

Beispiel

```
CALL tolInitialize(0,1,50,50,40,41)  'Werkzeugladenummer:0, Werkzeugnummer:1,
                                     'Einspannpunkttyp:P50,
                                     'Annäherungslänge:50mm
                                     'Einspann-DIO-Nummer:40,
                                     'Loslassen-DIO-Nummer:41
```

tolKernel (Bibliothek)

Funktion

Werkzeugänderung Bewegung (Kernel)

Format

tolKernel (<Werkzeugladenummer>, <Werkzeugnummer>, <Aktionsnummer>, <Aktionsargument>)

Erklärung

Diese Anweisung ist ein Kernel-Programm für die Werkzeugänderung. Dieses Programm ist zur Verwendung aller anderen Werkzeugänderungsbibliotheken erforderlich.

Makrodefinition

mcToolMax	:	maximale Anzahl der Werkzeuge.
McTools	:	maximal verfügbare Anzahl Werkzeuge für mehrere Hände.

Verwandte Begriffe

tolInitialize

tolMain1 (Bibliothek)

Funktion

Werkzeugänderung Schablone 1

Format

tolMain1

Erklärung

Diese Anweisung ist ein Beispiel für eine Werkzeugänderung.
Bei Angabe einer Werkzeugnummer mit einer 8-Bit-Nummer von ioINB wird das Werkzeug auf das angegebene geändert.

Makrodefinition

ioINB : Die erste Nummer einer E/A-Werkzeugnummernangabe

Verwandte Begriffe

tolChange, tolInitialize, tolKernel

22.4 Eingabe/Ausgabe

dioSync (Bibliothek)

Funktion

Führt eine Synchronisierung mit einem an DIO angeschlossenen externen Gerät (z. B. einem Sequenzierer) durch.

Format

dioSync (<Datenempfangsberechtigung Signalnummer>, <Datenempfangsabschluß Signalnummer>)

Erklärung

Diese Anweisung ist die Standardprozedur zur Synchronisierung mit einem an eine DIO angeschlossenen externen Gerät (z. B. einem Sequenzierer).

Beispiel

```
CALL dioSync(220,221)       'Synchronisierung mit dem angeschlossenen externen
                            'Gerät durch Zuordnen eines Signals zur
                            'Datenempfangsberechtigung und
                            'zum Datenempfangsabschluß an IO220 und IO221
```

22.5 Armbewegung

mvResetPulseWidth (Bibliothek)

Funktion

Stellt die Standardimpulswerte des Codierers für die zulässige Positionsabweichung wieder her.

Format

mvResetPulseWidth

Erklärung

Stellt die Standard-Codiererimpulszähler (20) für die Positionierungstoleranz für alle Achsen wieder her.

Ein Codiererimpulszähler für die Positionierungstoleranz verweist auf die Konvergenzgenauigkeit für ein Gelenk, die bei der Ausführung eines Bewegungsbefehls mit der Option @E verwendet wird.

Makrodefinition

Erfordert die Datei <pacman.h>

Beispiel

```
CALL mvResetPulseWidth      'Stellt die Standard-Codiererimpuszähler für alle
                            'Achsen wieder her.
```

mvResetTimeOut (Bibliothek)

Funktion

Stellt den Standard-Zeitüberschreitungswert für den Bewegungsabschluß wieder her.

Format

mvResetTimeOut

Erklärung

Diese Anweisung stellt den Zeitüberschreitungswert für den Bewegungsabschluß auf einen Standardwert von 5600 Millisekunden (5,6 Sek.) ein.
Der Zeitüberschreitungswert für den Bewegungsabschluß ist der Grenzwert für die Zeit, die die Achse braucht, um die zulässige Stopimpulsbreite zu erreichen, wenn sich die Achse während der Bewegung zur Codiererwertprüfung bewegt (@E-Angabe).
Wenn die Achse die zulässige Impulsbreite nicht innerhalb des Grenzwertes erreicht, wird die Alarmmeldung “6651 Prüfzeit für Anweisung überschritten” angezeigt.

Makrodefinition Definition

Erfordert die Datei <pacman.h>

Beispiel

```
CALL mvResetTimeOut          'Stellt den Standard-Zeitüberschreitungswert für den
                             'Bewegungsabschluß wieder her.
```

mvReverseFlip (Bibliothek)

Funktion

Umkehrung der Armstellung bei 4 Achsen

Format

mvReverseFlip (<Basisposition>, <Position ohne Prüfung>, <Entscheidungs-basiswert>)

Erklärung

Das System vergleicht den 4-Achsen-Wert von <Referenzposition> mit dem von <Position ohne Prüfung>. Diese Anweisung kehrt KIPPEN/NICHT KIPPEN um, wenn der 4-Achsen-Wert den <Entscheidungsreferenzwert> übersteigt.

Beispiel

```
CALL mvReverseFlip(P0,P1,90) 'Vergleicht die Basisposition P0 mit der Position
                             'ohne Prüfung P1. Wenn die absolute Differenz 90
                             'übersteigt, kehrt das System die 4-Achsen-
                             'Armstellung um.
```

mvSetPulseWidth (Bibliothek)

Funktion

Definiert die zulässige Breite des Stopimpulses.

Format

mvSetPulseWidth (<Zulässige Stopimpulsbreite J1>, <Zulässige Stopimpulsbreite J2>, < Zulässige Stopimpulsbreite J3>, <Zulässige Stopimpulsbreite J4>, <Zulässige Stopimpulsbreite J5>, <Zulässige Stopimpulsbreite J6>)

Erklärung

Diese Anweisung stellt die zulässige Stopimulsbreite ein.
Die zulässige Stopimpulsbreite ist ein Fehlerimpuls zur Feststellung, daß der Motor gestoppt wurde, wenn der Controller den Codiererwert bestätigt (durch Angabe von @E).

Makrodefinition

Erfordert die Datei <pacman.h>

Beispiel

call mvSetPulseWidth (10,10,10,10,10,10) setzt 10 Impulse zum Stoppen des Genauigkeitsbereiches für alle Achsen.

Anmerkungen

Stellen Sie die Impulsbreite auf 1 oder mehr ein. Bei Angabe von 0 wird der Standardwert (20) verwendet. Bei Angabe eines negativen Wertes wird der Fehler “6003 Ein Wert überschreitet einen effektiven Wertebereich” angezeigt. Bei Angabe eines kleineren Wertes wird der Fehler “6651 Prüfzeit für Anweisung überschritten” angezeigt.

mvSetTimeOut (Bibliothek)

Funktion

Definiert den Zeitüberschreitungswert für das Ende der Bewegung.

Format

mvSetTimeOut (<Zeitüberschreitungswert für den Bewegungsabschluß>)

Erklärung

Diese Anweisung setzt den Zeitüberschreitungswert für den Bewegungsabschluß (Einheit: ms).
Der Zeitüberschreitungswert für den Bewegungsabschluß ist der Grenzwert für die Zeit, die die Achse braucht, um die zulässige Stopimpulsbreite zu erreichen, wenn sich eine Achse während der Bewegung zur Codiererwertprüfung bewegt (@E-Angabe).
Wenn die Achse die zulässige Impulsbreite nicht innerhalb des Grenzwertes erreicht, wird die Alarmmeldung “6651 Prüfzeit für Anweisung überschritten” angezeigt.

Makrodefinition

Erfordert die Datei <pacman.h>

Beispiel

call mvSetTimeOut (3000) Stellt den Zeitüberschreitungswert auf 3 (s) ein.

Anmerkungen

Stellen Sie den Zeitüberschreitungswert auf 1 oder mehr ein. Bei Angabe von 0 wird der Standardwert (5600) verwendet. Bei Angabe eines negativen Wertes wird der Fehler “6003 Ein Wert überschreitet einen effektiven Wertebereich” angezeigt. Bei Angabe eines kleineren Wertes wird der Fehler “6651 Prüfzeit für Anweisung überschritten” angezeigt.

SetGravity [V1.2 oder später]

Funktion

Kompensiert die auf die einzelnen Gelenke angewendete statische Last (Schwerkraftdrehmoment) und erzielt eine Balance mit dem Schwerkraftdrehmoment.

Format

SetGravity

Erklärung

Jedes Gelenk des Roboters unterliegt aufgrund der Erdanziehungskraft einer nach unten gerichteten statischen Belastung (Schwerkraft-Drehmoment). Die Auswirkungen dieses Schwerkraftdrehmomentes unterscheiden sich je nach der Masse der Nutzlast (Endwerkzeug und Werkstück), dem Nutzlastschwerpunkt und der Roboterarmstellung.

Wenn Sie das Drehmoment der Motorausgabe begrenzen, indem Sie den Stromgrenzwert für den Antrieb beschränken, so daß das begrenzte Drehmoment niedriger ist als das Schwerkraftdrehmoment, bewegt sich der Roboter in Richtung Erde. Um dies zu verhindern, kompensiert diese Anweisung das begrenzte Drehmoment für das Schwerkraftdrehmoment und erhält das Gleichgewicht des Drehmomentes.

Makrodefinition

Erfordert die Datei <pacman.h>

Verwandte Begriffe

SetCurLmt, ResetGravity, SetGrvOffset

Anmerkungen

(1) Schreiben Sie diesen Befehl in eine TAKEARMed-Task, die einen Arm-Semaphor abgerufen hat. Wenn dieser Befehl ohne Abrufen eines Arm-Semaphor ausgeführt wird, führt dies zu einem Fehler 21F7.

(2) Stellen Sie die Masse und den Schwerpunkt der Nutzlast genau ein. Andernfalls kann es passieren, daß sich der Roboter in Richtung Erde bewegt, wenn Sie einen zu niedrigen Strombegrenzungswert einstellen (z. B. niedriger als 30). Weitere Informationen zur Eingabeprozedur für die Masse und den Schwerpunkt der Nutzlast, siehe Abschnitt unter 4.7 “Steuerparameter in den Benutzereinstellungen” in Kapitel 4 dieses Handbuches.

(3) Wenn Sie die genaue Masse der Nutzlast oder ihren Schwerpunkt nicht kennen, oder wenn sich der Roboter trotz der genauen Einstellungen in Richtung Erde bewegt, verwenden Sie die Schwerkraft-Offsetfunktion (SetGrvOffset), die den Schwerkraftausgleich kompensiert.

(4) Wenn Sie den Schwerkraft-Offset am Programmiergerät auf “1” einstellen, wird die Schwerkraft-Offsetfunktion aktiviert. Die am Programmiergerät vorgenommene Einstellung wird sofort nach Abschluß der Kalibrierung nach dem Einschalten des Roboters wirksam.

Beispiel

```
CALL SetGravity         'Aktiviert die Funktion zum Schwerkraftausgleich.
Delay 100               'Wartet, bis der Schwerkraftausgleich wirksam wird.
CALL SetCurLmt (2,30)   'Stellt den Strombegrenzungswert der 2. Achse auf 30%
                        'ein.
```

ResetGravity [V1.2 oder später]

Funktion

Deaktiviert die Balanceeinstellung zwischen dem begrenzten Motordrehmoment und dem Schwerkraftdrehmoment, die mit SetGravity definiert wurde.

Format

ResetGravity

Erklärung

Dieser Befehl deaktiviert die Balanceeinstellung zwischen dem durch die Strombegrenzungsfunktion eingeschränkten Motordrehmoment und dem Schwerkraftdrehmoment.

Makrodefinition

Erfordert die Datei <pacman.h>

Verwandte Begriffe

ResetCurLmt, SetEralw

Anmerkungen

(1) Schreiben Sie diesen Befehl in eine TAKEARMed-Task, die einen Arm-Semaphor abgerufen hat. Wenn dieser Befehl ohne Abrufen eines Arm-Semaphor ausgeführt wird, führt dies zu einem Fehler 21F7.

(2) Wenn dieser Befehl ausgeführt wird, während die Strombegrenzungsfunktion aktiviert ist, tritt der Fehler 665b auf. Deaktivieren Sie die Strombegrenzungsfunktion und versuchen Sie es erneut.

(3) Wenn Sie den Schwerkraft-Offset am Programmiergerät auf “0” einstellen, wird die Schwerkraft-Offsetfunktion deaktiviert. Wenn Sie diese Aktion ausführen, während die Strombegrenzungsfunktion aktiviert ist, tritt der Fehler 665b auf, wie in Schritt (2).

Beispiel

```
CALL ResetGravity
```

SetGrvOffset [V1.2 oder später]

Funktion

Kompensiert das Drehmoment jedes Gelenks, das mit SetGravity für das Schwerkraftdrehmoment programmiert wurde.

Format

SetGrvOffset

Erklärung

Jedes Gelenk des Roboters unterliegt aufgrund der Erdanziehungskraft einer nach unten gerichteten statischen Belastung (Schwerkraftdrehmoment). Auch wenn der Befehl für den Schwerkraftausgleich (SetGravity) die Anpassung der Balance zwischen dem begrenzten Drehmoment und dem Schwerkraftdrehmoment ermöglicht, kann die Balance durch den Unterschied zwischen der Masse der eingestellten und der tatsächlichen Nutzlast aus dem Gleichgewicht geraten.

Diese Offset-Function verwendet das Schwerkraftdrehmoment, wenn der Roboter in Halteposition ist, und berechnet den Wert des Schwerkraft-Offset.

Makrodefinition

Erfordert die Datei <pacman.h>

Verwandte Begriffe

SetCurLmt, SetGravity, ResetGrvOffset

Anmerkungen

(1) Schreiben Sie diesen Befehl in eine TAKEARMed-Task, die einen Arm-Semaphor abgerufen hat. Wenn dieser Befehl ohne Abrufen eines Arm-Semaphor ausgeführt wird, führt dies zu einem Fehler 21F7.

(2) Dieser Befehl sollte ausgeführt werden, wenn die Stromversorgung des Motors eingeschaltet ist und der Roboter in Halteposition steht. Wenn er ausgeführt wird, während die Stromversorgung des Motors ausgeschaltet ist, tritt der Fehler 6006 auf. Wird er ausgeführt, während der Roboter in Bewegung ist, tritt der Fehler 600B auf.

(3) Wenn die Robotereinstellung nach der Ausführung dieses Befehles erheblich geändert wird, führen Sie diesen Befehl erneut aus.

(4) Ist der Rücksetz-Stromgrenzwert in den Benutzereinstellungen auf einen anderen Wert als “1”, “3”, “5” oder “7” eingestellt, wird der Kompensationswert beim Einschalten des Motors auf “0” zurückgesetzt.

Beispiel

```
CALL SetGrvOffset
```

ResetGrvOffset [V1.2 oder später]

Funktion

Deaktiviert die Schwerkraft-Offsetfunktion.

Format

ResetGrvOffset

Erklärung

Deaktiviert die Schwerkraft-Offsetfunktion, die mit SetGrvOffset aktiviert wurde.

Makrodefinition

Erfordert die Datei <pacman.h>.

Verwandte Begriffe

SetGrvOffset

Anmerkungen

(1) Schreiben Sie diesen Befehl in eine TAKEARMed-Task, die einen Arm-Semaphor abgerufen hat. Wenn dieser Befehl ohne Abrufen eines Arm-Semaphor ausgeführt wird, führt dies zu einem Fehler 21F7.

(2) Dieser Befehl sollte ausgeführt werden, wenn der Roboter in Halteposition steht. Wird er ausgeführt, während der Roboter in Bewegung ist, tritt der Fehler 600B auf. Dieser Befehl kann auch ausgeführt werden, wenn die Stromversorgung des Motors ausgeschaltet ist.

Beispiel

```
CALL ResetGrvOffset
```

SetCurLmt [V1.2 oder später]

Funktion

Legt den Grenzwert für den Motorstrom fest, der auf die angegebene Achse angewendet werden soll.

Format

SetCurLmt (<Achsennummer>, <Wert>)

Erklärung

Begrenzt den Wert des Motorstroms (Drehmoment) für die mit <Achsennummer> angegebene Achse auf den mit <Wert> angegebenen Wert. Dieser Befehl ist hilfreich, wenn Sie das Drehmoment begrenzen wollen, das beim Zusammenfügen oder Stoßen auf das Werkstück angewandt werden soll.

Der maximale <Wert> lautet 100; dies bezieht sich auf die Angabe der Motorstromrate. Bei Angabe eines Wertes, der das zulässige Limit für die Achsen überschreitet, wird der entsprechende Wert automatisch auf das zulässige Limit begrenzt.

Stellen Sie einen Wert 1 oder größer ein. Bei Angabe von 0 oder einem negativen Wert tritt Fehler 6003 auf.

Makrodefinition

Erfordert die Datei <pacman.h>

Verwandte Begriffe

ResetCurLmt, SetGravity, SetGrvOffset, SetEralw

Anmerkungen

(1) Wenn der Motorstrom mit SetCurLmt begrenzt wird, kann der Roboter nicht mit der maximalen Geschwindigkeit oder Beschleunigung bewegt werden. Verwenden Sie SetCurLmt nur bei Schritten, für die der Stromgrenzwert erforderlich ist. Verringern Sie bei Verwendung von SetCurLmt die Beschleunigung.

(2) Wenn ein Werkstück mit hoher Geschwindigkeit auf ein Objekt trifft, ist die Auswirkung wegen der Trägheit des Werkstückes, des Endwerkzeugs und der Achse erheblich, auch wenn die Antriebskraft kontrolliert ist. Stellen Sie den aktuellen Grenzwert ein, unmittelbar bevor das Werkstück mit dem Objekt in Kontakt kommt, und verringern Sie die Geschwindigkeit.

(3) Stellen Sie den Stromgrenzwert ein, wenn der Roboter in Halteposition ist. Wenn er während einer Durchlaufbewegung eingestellt wird, kann leicht ein Fehler auftreten.

(4) Schreiben Sie diesen Befehl in eine TAKEARMed-Task, die einen Arm-Semaphor abgerufen hat. Wenn dieser Befehl ohne Abrufen eines Arm-Semaphor ausgeführt wird, führt dies zu einem Fehler 21F7.

(5) Stellen Sie beim Einstellen des Stromgrenzwertes sicher, daß die Schwerkraftausgleichfunktion aktiviert ist. Wenn der Stromgrenzwert eingestellt wird, während der Schwerkraftausgleich deaktiviert ist, tritt der Fehler 665a auf. Weitere Informationen zur Schwerkraftausgleichsfunktion, siehe SetGravity.

6 Achsen

(6) Ist der Rücksetz-Stromgrenzwert in den Benutzereinstellungen auf einen anderen Wert als “1+” eingestellt, wird der Stromgrenzwert beim Einschalten des Motors zurückgesetzt. Wenn die Stromgrenzwertfunktion unmittelbar nach dem Einschalten der Stromversorgung des Motors wirksam werden soll, stellen Sie den Rücksetz-Stromgrenzwert auf “1” ein.

6 Achsen

(7) Stellen Sie die Masse und den Schwerpunkt der Nutzlast genau ein. Andernfalls kann es passieren, daß sich der Roboter in Richtung Erde bewegt, wenn Sie einen zu niedrigen Strombegrenzungswert einstellen (z. B. niedriger als 30). Weitere Informationen zur Eingabeprozedur für die Masse und den Schwerpunkt der Nutzlast Abschnitt 5.7, siehe “Einstellen der Master-Steuerparameter in den Benutzereinstellungen” in Kapitel 4 dieses Handbuches.

6 Achsen

(8) Wenn der Rücksetz-Stromgrenzwert auf “1+” eingestellt ist, kann sich der Roboter nach dem Einschalten der Motorstromversorgung nach unten bewegen. Setzen Sie den Stromgrenzwert durch Ausführen von ResetCurLmt zurück, wenn die Motorstromversorgung ausgeschaltet ist, und schalten Sie anschließend die Motorstromversorgung ein.

6 Achsen

(9) Wenn Sie die genaue Masse der Nutzlast oder ihren Schwerpunkt nicht kennen, oder wenn sich der Roboter trotz der genauen Einstellungen in Richtung Erde bewegt, verwenden Sie die Schwerkraft-Offsetfunktion (SetGrvOffset), die den Schwerkraftausgleich kompensiert.

Beispiel

6 Achsen

```
CALL SetGravity         'Aktiviert den Schwerkraft-Offset.
CALL SetGrvOffset       'Kompensiert den Offsetwert für die
                        'Schwerkraft
CALL SetEralw (2, 20)   'Stellt die zulässige Abweichung der 2. Achse
                        'auf 20 Grad ein
CALL SetCurLmt (2, 30)  'Stellt den Strombegrenzungswert der 2.
                        'Achse auf 30% ein.
```

4 Achsen

```
CALL SetEralw (2, 20)   'Stellt die zulässige Abweichung der 2. Achse
                        'auf 20 Grad ein.
CALL SetCurLmt (2, 30)  'Stellt den Strombegrenzungswert der 2.
                        'Achse auf 30% ein.
```

ResetCurLmt [V1.2 oder später]

Funktion

Setzt den Motorstromgrenzwert für die angegebene Achse zurück.

Format

ResetCurLmt (<Achsennummer>)

Erklärung

Setzt den Motorstromgrenzwert der in <Achsennummer> angegebenen Achse zurück. Nach dem Zurücksetzen entsprechen der zulässige Wert für die Abweichung und der Stromgrenzwert wieder ihren Ausgangswerten.

Wenn die <Achsennummer> mit 0 angegeben ist, wird der Stromgrenzwert für alle Achsen zurückgesetzt.

Makrodefinition

Erfordert die Datei <pacman.h>

Verwandte Begriffe

SetCurLmt, ResetEralw

Anmerkungen

(1) Beim Zurücksetzen des Stromgrenzwertes führt der Befehl eine Eliminierung der Abweichung aus. Wenn eine Winkelabweichung aufgrund einer externen Kraft vorliegt, hängt die erforderliche Zeit für die Eliminierung der Abweichung von der eingestellten Geschwindigkeit und Beschleunigung ab. Stellen Sie zum Verkürzen der Zeit eine höhere Geschwindigkeit und Beschleunigung ein.

(2) Der Befehl kann auch ausgeführt werden, wenn die Motorstromversorgung ausgeschaltet ist. Zum Zurücksetzen des Stromgrenzwertes bei ausgeschalteter Motorstromversorgung führen Sie das folgende Programm aus, nachdem die Task zum Abrufen des Arm-Semaphor abgeschlossen ist.

```
PRO999
TAKEARM
CALL ResetCurLmt(0)
END
```

(3) Schreiben Sie diesen Befehl in eine TAKEARMed-Task, die einen Arm-Semaphor abgerufen hat. Wenn dieser Befehl ohne Abrufen eines Arm-Semaphor ausgeführt wird, führt dies zu einem Fehler 21F7.

Beispiel

```
CALL ResetCurLmt(0)          'Setzt den Stromgrenzwert für alle Achsen zurück.
```

SetEralw [V1.2 oder später]

Funktion

Ändert die zulässige Abweichung für die angegebene Achse.

Format

SetEralw (<Achsennummer>, <Wert>)

Erklärung

Stellt die zulässige Abweichung der mit <Achsennummer> angegebenen Achse ein. Verwenden Sie diesen Befehl, wenn aufgrund einer externen Kraft eine Winkelabweichung auftritt, wenn die Strombegrenzungsfunktion aktiviert ist.

Der <Wert> ist der Armgelenkswinkel; er wird in Grad angegeben.

“Zulässiger Abweichungswert” bezieht sich auf den zulässigen Bereich zum “Fehler 611*J* Zu starke Abweichung”, der aus Sicherheitsgründen auftritt, wenn die Servo-Abweichung den angegebenen Wert überschreitet.

Wenn während der Konstruktionsoperation mit aktiviertem Stromgrenzwert aufgrund einer externen Kraft eine Servo-Abweichung auftritt, kann der obige Fehler auftreten. Um dies zu vermeiden, können Sie diesen Befehl temporär verwenden, um die zulässige Abweichung zu vergrößern.

Mit diesem Befehl kann auch die zulässige Abweichung reduziert werden, um ein schnelles Erkennen der Abwärtsbewegung des Roboters aufgrund der Schwerkraft zu erkennen, wenn die Strombegrenzungsfunktion aktiviert ist.

Makrodefinition

Erfordert die Datei <pacman.h>

Verwandte Begriffe

SetCurLmt, ResetEralw

Anmerkungen

(1) Führen Sie diesen Befehl in einer TAKEARMed-Task aus, die einen Arm-Semaphor abgerufen hat. Wenn der Befehl ohne Abrufen eines Arm-Semaphor ausgeführt wird, führt dies zu einem Fehler 21F7.

Beispiel

```
CALL SetEralw(2,20)          'Stellt die zulässige Abweichung der 2. Achse auf
                             '20 Grad ein.
```

ResetEralw [V1.2 oder später]

Funktion

Setzt den Wert für die zulässige Abweichung der angegebenen Achse auf den Anfangswert zurück.

Format

ResetEralw(<Achsennummer>)

Erklärung

Setzt den Wert für die zulässige Abweichung der mit <Achsennummer> angegebenen Achse auf den Anfangswert zurück.
Wenn <Achsennummer> mit “0+” angegeben ist, werden die Werte für die zulässige Abweichung für alle Achsen zurückgesetzt.

Makrodefinition

Erfordert die Datei <pacman.h>

Verwandte Begriffe

ResetCurLmt, SetEralw

Anmerkungen

(1) Führen Sie diesen Befehl in einer TAKEARMed-Task aus, die einen Arm-Semaphor abgerufen hat. Wenn dieser Befehl ohne Abrufen eines Arm-Semaphor ausgeführt wird, führt dies zu einem Fehler 21F7.

(2) Wie dieser Befehl setzt auch die Ausführung von ResetCurLmt die Werte für die zulässige Abweichung auf die Anfangswerte zurück.

Beispiel

```
CALL ResetEralw(0)          'Setzt die Werte für die zulässige Abweichung für
                            'alle Achsen auf die Anfangswerte zurück.
```

OnSrvLock (Bibliothek)

Funktion

Führt eine Servo-Sperre einer angegebenen Achse aus (ausschließlich für Roboter mit 4 Achsen).

Format

OnSrvLock (<angegebene Achse>)

Erklärung

Diese Anweisung bietet eine ähnliche Funktion wie eine ON SVLOCK-Anweisung in der konventionellen Sprache.
Servo-Sperre bedeutet, daß Roboterarme kontrolliert und ihre Positionen beibehalten werden.

Makrodefinition

Hierzu ist die Datei <pacman.h> erforderlich.

Verwandter Begriff

OffSrvLock

Anmerkungen

Setzen Sie die Servo-Sperre, wenn der Roboter stoppt. Wenn diese Einstellung während einer Pfadoperation gesetzt wird, kann ein Fehler auftreten.

Führen Sie diesen Befehl mit einer Task aus, die Robotersteuerbedingungen abruft (TAKEARM). Wenn keine Robotersteuerbedingungen abgerufen werden, wird die Fehlermeldung “21F7 Arm-Semaphor kann nicht abgerufen werden” angezeigt.

Beispiel

```
CALL  OnSrvLock(1)          'Servo-Sperre für Achse 1
CALL  OnSrvLock(0)          'Servo-Sperre für alle Achsen
```

OffSrvLock (Bibliothek)

Funktion

Gibt die Servo-Sperre für die angegebene Achse frei. (Ausschließlich für Roboter mit 4 Achsen)

Format

OffSrvLock (<angegebene Achse>)

Erklärung

Diese Anweisung bietet eine ähnliche Funktion wie eine OFF SVLOCK-Anweisung in der konventionellen Sprache.

Servo-Sperre bezieht sich auf den Status, bei dem der Roboterarm kontrolliert wird, um seine Position zu halten. Beim Freigeben wird der Roboterarm nicht an seiner Position gehalten, sondern von einer darauf angewendeten Kraft bewegt.

Makrodefinition

Die Datei <pacman.h> ist erforderlich.

Verwandter Begriff

OnSrvLock

Anmerkungen

(1) Es kann kein Operationsbefehl für eine Achse ausgeführt werden, für die die Servo-Sperre freigegeben wurde.

(2) Geben Sie die Servo-Sperre frei, während sich der Roboter im Status “Gestoppt” befindet. Wenn diese Einstellung während einer Pfadoperation gesetzt wird, kann ein Fehler auftreten.

(3) Führen Sie diesen Befehl in der Task aus, die die Berechtigung zur Robotersteuerung abruft (TAKEARM). Wenn die Berechtigung zur Robotersteuerung nicht abgerufen wurde, tritt der Fehler “21F7 Arm-Semaphor kann nicht abgerufen werden” auf.

(4) Wenn Bit 2 des für “25: Stromgrenzwert zurücksetzen” gesetzten Wertes in den Bedienungsbedingungen “0” (Anfangswert) ist, wird die Servo-Sperre beim Einschalten des Motors zurückgesetzt (und löst eine Sperre aus). Zur Überprüfung der Freigabe der Servo-Sperre unmittelbar nach dem Einschalten der Motorstromversorgung setzen Sie “+2” als Rücksetzstromgrenzwert.

Anmerkung: Beispiel zum Einstellen der Bertriebsbedingung “25: Stromgrenzwert zurücksetzen”

	PWM	SVLock	Curlmt	
	*	*	*	
Wenn nur SVLock wirksam ist	0	1	0	= 2
Wenn alles wirksam ist	1	1	1	= 7

Beispiel

```
CALL OffSrvLock (1)     ' Gibt die Servo-Sperre für Achse 1 frei.
```

OnPWM (Bibliothek)

Funktion

Führt eine PWM-Umschaltung an einer angegebenen Achse aus (ausschließlich für Roboter mit 4 Achsen).

Format

OnPWM (<angegebene Achse>)

Erklärung

PWM-Umschaltung bedeutet, daß Roboterarme kontrolliert und ihre Positionen beibehalten werden.

Makrodefinition

Hierzu ist die Datei <pacman.h> erforderlich.

Verwandte Begriffe

OffPWM

Anmerkungen

(1) Stellen Sie die PWM-Umschaltung ein, wenn der Roboter stoppt. Wenn diese Einstellung während einer Pfadoperation gesetzt wird, kann ein Fehler auftreten.

(2) Führen Sie diesen Befehl mit einer Task aus, die Robotersteuerbedingungen abruft (TAKEARM). Wenn keine Robotersteuerbedingungen abgerufen werden, wird die Fehlermeldung “21F7 Arm-Semaphor kann nicht abgerufen werden” angezeigt.

Beispiel

```
CALL OnPWM(1) ' PWM-Umschaltung an 1 Achse
CALL OnPWM(1) ' PWM-Umschaltung an allen Achsen
```

OffPWM (Bibliothek)

Funktion

Gibt die PWM-Umschaltung an der angegebenen Achse frei (ausschließlich für Roboter mit 4 Achsen).

Format

OffPWM (<angegebene Achse>)

Erklärung

PWM-Umschaltung bedeutet, daß Roboterarme kontrolliert und ihre Positionen beibehalten werden. Wenn die PWM-Umschaltung freigegeben wird, werden die Positionen der Roboterarme nicht gehalten; sie können durch externe Kräfte geändert werden.

Makrodefinition

Hierzu ist die Datei <pacman.h> erforderlich.

Verwandte Begriffe

OnSrvLock

Anmerkungen

(1) Ein Operationsbefehl kann nicht auf einer Achse mit freigegebener PWM-Umschaltung ausgeführt werden.

(2) Setzen Sie die PWM-Umschaltung, wenn der Roboter stoppt. Wenn diese Einstellung während einer Pfadoperation gesetzt wird, kann ein Fehler auftreten.

(3) Führen Sie diesen Befehl mit einer Task aus, die Robotersteuerbedingungen abruft (TAKEARM). Wenn keine Robotersteuerbedingungen abgerufen werden, wird die Fehlermeldung “21F7 Arm-Semaphor kann nicht abgerufen werden” angezeigt.

(4) Wenn Bit 3 des “25: Stromgrenzwert zurücksetzen” gesetzten Wertes auf 0 (Anfangswert) als Verwendungsbedingung gesetzt ist, wird die PWM-Umschaltung beim Einschalten der Motorstromversorgung zurückgesetzt. Zur Überprüfung der Freigabe der PWM-Umschaltung unmittelbar nach dem Einschalten der Motorstromversorgung setzen Sie +4 für den Wert “Stromgrenzwert zurücksetzen”.

Anmerkung: Beispiel zur Verwendung von Bedingung “25: Stromgrenzwert zurücksetzen”

	PWM *	SVLock *	Curlmt *	
Nur PWM gültig	0	1	0	= 2
Alle gültig	1	1	1	= 7

Beispiel

```
CALL OffPWM(1) ' PWM-Umschaltung Freigabe an Achse 1
```

Setcycloid (Bibliothek)

Funktion

Bewirkt einen Wechsel in den Zykloidmodus zur Unterdrückung von Überschuß und Restvibration am Ende der Punkt-zu-Punkt-Operation.

Format

Setcycloid

Erklärung

Im zykloiden Operationsmodus kann die Geschwindigkeitsänderung bei einer Verzögerung geglättet werden. Der Überschuß und die Restvibration beim Stoppen können reduziert werden.

Die Operationszeit wird im zykloiden Operationsmodus geringfügig erhöht. Überprüfen Sie unbedingt die Zykluszeit.

Dieser Modus ist gültig, wenn nach der Ausführung von Setcycloid ein anderes Programm ausgeführt wird. Zum Zurücksetzen führen Sie entweder Resetcycloid aus, oder stellen Sie nach dem einmaligen Einstellen des manuellen Modus den automatischen Modus ein.

Makrodefinition

Die Datei <pacman.h> ist erforderlich.

Verwandter Begriff

Resetcycloid

Beispiel

```
CALL  Setcycloid' Bewirkt einen Übergang in den Zykloidmodus.
```

Resetcycloid (Bibliothek)

Funktion

Bewirkt einen Übergang vom Zykloidmodus in den normalen Betriebsmodus.

Format

Resetcycloid

Erklärung

Bewirkt einen Übergang vom Zykloidmodus in den normalen Betriebsmodus. Wenn der Zykloidmodus erneut eingestellt werden soll, führen Sie Setcycloid aus.

Makrodefinition

Die Datei <pacman.h> ist erforderlich.

Verwandter Begriff

Setcycloid

Anmerkungen

(1) Führen Sie diesen Befehl in einer Task mit Robotersteuerbedingungen aus (TAKEARM). Wenn keine Robotersteuerbedingungen gehalten werden, wird die Fehlermeldung “21F7 Arm-Semaphor kann nicht abgerufen werden” angezeigt.

Beispiel

```
CALL  Resetcycloid     ' Setzt den Zykloidmodus zurück.
```

SetForce_HM (Bibliothek)

Funktion

Eine Strombegrenzungsbibliothek, die den Schub (Einheit: N) der Z-Koordinate bei einem HM/HS-Roboter angibt.

Format

SetForce_HM (<Schub-(Einheit: N>)

Erklärung

Begrenzt die Stoßkraft durch Begrenzen des Motorstroms entlang der Z-Achse. Die Einheit ist N (Newton). Die maximale Stoßkraft weist erhebliche Unterschiede auf je nach der einwirkenden Kraft, die beim Kontakt mit Peripheriegeräten generiert wird. Die zulässige Stoßkraft ist ein Wert, der auftritt, wenn die bei der Kollision einwirkende Kraft fast Null beträgt (Kollision bei sehr niedriger Geschwindigkeit). Die Stoßkraft hängt auch von der Z-Achsenposition der mechanischen Komponente ab; dieser Wert sollte daher einen typischen Wert (keinen garantierten Wert) annehmen. Wie bei der Strombegrenzungsbibliothek (SetCurlmt) wird ein Wert, der den zulässigen Wert überschreitet, auf den zulässigen Wert begrenzt.

Makrodefinition

Die Datei <pacman.h> ist erforderlich.

Verwandte Begriffe

SetForce_HC, SetCurlmt, ResetCurlmt

Anmerkungen

(1) Da der Motorstrom durch die Strombegrenzung eingeschränkt ist, kann keine Operation mit der höchsten Geschwindigkeit und Beschleunigung durchgeführt werden. Verwenden Sie für die Strombegrenzung nur die erforderlichen Schritte. Verringern Sie zur Verwendung der Strombegrenzung die Beschleunigung.

(2) Wenn die Stoßkraft durch die Strombegrenzung eingeschränkt ist, kann bei einer Kollision mit hoher Geschwindigkeit wegen der Trägheit zwischen Arbeit und Hand eine große Kraft wirken. Stellen Sie die Strombegrenzung vor der Einstellung der Arbeit ein und verringern Sie die Geschwindigkeit.

(3) Stellen Sie die Strombegrenzung ein, wenn der Roboter stoppt. Wenn diese Einstellung während einer Pfadoperation gesetzt wird, kann ein Fehler auftreten.

(4) Führen Sie diese Funktion mit einer Task aus, die das Recht zur Robotersteuerung abruft (TAKEARM). Wenn das Recht zur Robotersteuerung nicht abgerufen wird, wird die Fehlermeldung “21F7 Arm-Semaphor kann nicht abgerufen werden” angezeigt.

(5) Wenn Bit 1 des eingestellten Wertes “25: Strombegrenzung zurück” auf 0 (Anfangswert) als Verwendungsbedingung gesetzt ist, wird die Servo-Sperre beim Einschalten des Motors zurückgesetzt (und löst eine Sperre aus). Zur Überprüfung der Freigabe der Servo-Sperren unmittelbar nach dem Einschalten der Motorstromversorgung setzen Sie +1 für den Wert “Strombegrenzung zurücksetzen”. Ausführliche Hinweise zur Einstellung finden Sie in den Anmerkungen zu “OffSrvlock.”

(6) Führen Sie bei der Freigabe dieses Befehls “ResetCurlmt(3)” aus.

(7) Auch wenn die Einstellung sorgfältig durchgeführt wird, kann die angegebene Stoßkraft aus verschiedenen Gründen wie beispielsweise Reibung eventuell nicht ausgegeben werden.

Beispiel

```
CALL  SetForce_HM(10.0)Stoßkraft angeben '10.0(N).
```

SetForce_HC (Bibliothek)

Funktion

Eine Strombegrenzungsbibliothek, die den Schub (Einheit: N) der Z-Koordinate bei einem HC-Roboter angibt.

Format

SetForce_HC (<SchubEinheit: N>)

Erklärung

Begrenzt die Stoßkraft durch Begrenzen des Motorstroms entlang der Z-Achse. Die Einheit ist N (Newton). Die maximale Stoßkraft weist erhebliche Unterschiede auf je nach der einwirkenden Kraft, die beim Kontakt mit Peripheriegeräten generiert wird. Die zulässige Stoßkraft ist ein Wert, der auftritt, wenn die bei der Kollision einwirkende Kraft fast Null beträgt (Kollision bei sehr niedriger Geschwindigkeit). Die Stoßkraft hängt auch von der Z-Achsenposition der mechanischen Komponente ab; dieser Wert sollte daher einen typischen Wert (keinen garantierten Wert) annehmen. Wie bei der Strombegrenzungsbibliothek (SetCurlmt) wird ein Wert, der den zulässigen Wert überschreitet, auf den zulässigen Wert begrenzt.

Makrodefinition

Hierzu ist die Datei <pacman.h> erforderlich.

Verwandte Begriffe

SetForce_HM, SetCurlmt, ResetCurlmt

Anmerkungen

(1) Da der Motorstrom durch die Strombegrenzung eingeschränkt ist, kann keine Operation mit der höchsten Geschwindigkeit und Beschleunigung durchgeführt werden. Verwenden Sie für die Strombegrenzung nur die erforderlichen Schritte. Verringern Sie zur Verwendung der Strombegrenzung die Beschleunigung.

(2) Wenn die Stoßkraft durch die Strombegrenzung eingeschränkt ist, kann bei einer Kollision mit hoher Geschwindigkeit wegen der Trägheit zwischen Arbeit und Hand eine große Kraft einwirken. Stellen Sie die Strombegrenzung vor der Einstellung der Arbeit ein und verringern Sie die Geschwindigkeit.

(3) Stellen Sie die Strombegrenzung ein, wenn der Roboter stoppt. Wenn diese Einstellung während einer Pfadoperation gesetzt wird, kann ein Fehler auftreten.

(4) Führen Sie diese Funktion mit einer Task aus, die das Recht zur Robotersteuerung abruft (TAKEARM). Wenn das Recht zur Robotersteuerung nicht abgerufen wird, wird die Fehlermeldung “21F7 Arm-Semaphor kann nicht abgerufen werden” angezeigt.

(5) Wenn Bit 1 des eingestellten Wertes “25: Strombegrenzung zurückgesetzt” auf 0 (Anfangswert) als Verwendungsbedingung gesetzt ist, wird die Servo-Sperre beim Einschalten des Motors zurückgesetzt (und löst eine Sperre aus). Zur Überprüfung der Freigabe der Servo-Sperren unmittelbar nach dem Einschalten der Motorstromversorgung setzen Sie +1 für den Wert “Strombegrenzung zurückgesetzt”. Ausführliche Hinweise zur Einstellung finden Sie in den Anmerkungen zu “OffSrvlock.”

(6) Führen Sie bei der Freigabe dieses Befehls “ResetCurlmt(3)” aus.

(7) Auch wenn die Einstellung sorgfältig durchgeführt wird, kann die angegebene Stoßkraft aus verschiedenen Gründen wie beispielsweise Reibung eventuell nicht ausgegeben werden.

Beispiel

```
CALL  SetForce_HC(10.0)Stoßkraft angeben '1(N).
```

SetCompControl (Bibliothek) [V1.4 oder später]

Funktion

Aktiviert die Nachgiebigkeitsfunktion (dedizierter Befehl für 6 Achsen).

Format

SetCompControl

Erklärung

Aktiviert eine Funktion zur Nachgiebigkeitssteuerung. Aktiviert die mit SetFrcLimit, SetCompRate und SetFrcCoord definierten Nachgiebigkeitsbedingungen.

Makrodefinition

<pacman.h> ist erforderlich.

Verwandte Begriffe

SetFrcLimit, SetCompRate, SetFrcCoord, ResetCompControl, SetCompFControl

Anmerkungen

(1) Bei Ausführung dieser Bibliothek mit deaktiviertem Schwerkraft-Offset und aktivierter Strombegrenzung tritt der Fehler “60f5 Nachgiebigkeitssteuerung kann nicht ausgeführt werden” auf. Führen Sie den Befehl erneut aus, nachdem Sie den Schwerkraft-Offset aktiviert und die Strombegrenzung deaktiviert haben. Hinweise zum Deaktivieren der Strombegrenzung und dem Aktivieren des Schwerkraft-Offset finden Sie unter ResetCurLmt und SetGravity.

(2) Die Nachgiebigkeitssteuerung wird nicht aktiviert, während die Motoren ausgeschaltet sind. Die Nachgiebigkeitssteuerung wird deaktiviert, wenn Sie die Motoren unter der Nachgiebigkeitssteuerung ausschalten.

(3) Führen Sie diesen Befehl in einer Task mit Robotersteuerbedingungen aus (TAKEARM). Wenn keine Robotersteuerbedingungen gehalten werden, wird die Fehlermeldung “21F7 Arm-Semaphor kann nicht abgerufen werden” angezeigt.

(4) Führen Sie die Bibliothek aus, während der Roboter stoppt. Die Ausführung dieser Bibliothek in einer Pfadbewegung bewirkt eine Ausführung nach dem Stoppen. Wenn nach der Ausführung dieser Bibliothek ein Fehler “600b Roboter ist in Bewegung” auftritt, warten Sie, bis Ihr Roboter stoppt, nachdem Sie einen Befehl wie beispielsweise Delay eingegeben haben.

(5) Wenn sich der Roboter aufgrund einer externen Kraft bewegt, kann ein Fehler “611* Zu große Abweichung” auftreten. Ändern Sie in diesem Fall die zulässige Abweichung. Verwenden Sie SetEralw zum Ändern der zulässigen Abweichung.

(6) Führen Sie diese Bibliothek nicht aus, wenn eine Kraft wie beispielsweise eine Kontaktkraft auf den Roboter angewendet wird. Verwenden Sie SetCompControl zum Aktivieren der Nachgiebigkeitssteuerungsfunktion, während eine Kraft auf den Roboter einwirkt.

(7) Wenn sich eine Roboterhaltung nach der Ausführung von SetCompControl erheblich ändert, kann ein Fehler im Kompensierungswert für den Schwerkraft-Offset generiert werden, wenn Ihr Roboter sich in Richtung der Schwerkraft bewegt. Wenn sich die Haltung im Verlauf der Nachgiebigkeitssteuerung erheblich ändert, deaktivieren Sie mit ResetCompControl die Nachgiebigkeitssteuerung, und führen Sie SetCompControl erneut aus, um die Nachgiebigkeitssteuerung zu aktivieren.

Beispiel

```
CALL SetFrcCoord (1)      'Stellt das Koordinatensystem für die
                          'Kraftbegrenzung ein
CALL SetFrcLimit (100, 0, 100, 100, 100, 100)
                          'Stellt die Kraftbegrenzungsrate ein
CALL SetCompControl       'Aktiviert die Nachgiebigkeitssteuerungsfunktion
CALL SetEralw (1, 90)'Stellt die zulässige Abweichung ein
```

SetCompFControl (Bibliothek) [V1.4 oder später]

Funktion

Aktiviert die Nachgiebigkeitssteuerungsfunktion (dedizierter Befehl für 6 Achsen).

Format

SetCompFControl

Erklärung

Aktiviert die Nachgiebigkeitssteuerungsfunktion als SetCompControl. Die Schwerkraft-Offsetkompensierung wird nicht als SetCompControl ausgeführt.

Makrodefinition

<pacman.h> ist erforderlich.

Verwandte Begriffe

SetCompControl

Anmerkungen

(1) Bei Ausführung dieser Bibliothek mit deaktiviertem Schwerkraft-Offset und aktivierter Strombegrenzung tritt der Fehler “60f5 Nachgiebigkeitssteuerung kann nicht ausgeführt werden” auf. Führen Sie den Befehl erneut aus, nachdem Sie den Schwerkraft-Offset aktiviert und die Strombegrenzung deaktiviert haben.

(2) Die Nachgiebigkeitssteuerung wird nicht aktiviert, während die Motoren ausgeschaltet sind. Die Nachgiebigkeitssteuerung wird deaktiviert, wenn Sie die Motoren unter der Nachgiebigkeitssteuerung ausschalten.

(3) Führen Sie diesen Befehl in einer Task mit Robotersteuerbedingungen aus (TAKEARM). Wenn keine Robotersteuerbedingungen gehalten werden, wird die Fehlermeldung “21F7 Arm-Semaphor kann nicht abgerufen werden” angezeigt.

(4) Führen Sie die Bibliothek aus, während der Roboter stoppt. Die Ausführung dieser Bibliothek in einer Pfadbewegung bewirkt eine Ausführung nach dem Stoppen. Wenn nach der Ausführung dieser Bibliothek ein Fehler “600b Roboter ist in Bewegung” auftritt, warten Sie, bis Ihr Roboter stoppt, nachdem Sie einen Befehl wie beispielsweise Delay eingegeben haben.

(5) Stellen Sie die Nutzlast an der Spitze exakt ein. Der Roboter kann in Richtung der Schwerkraft fallen, wenn sich die Einstellung der Nutzlast an der Spitze und die tatsächliche Nutzlast an der Spitze unterscheiden. Sie können SetGrvOffset ausführen, um ein durch die Schwerkraft verursachtes Fallen zu verhindern.

Beispiel

```
CALL SetGrvOffset      'Berechnet den Wert für die Schwerkraft-
                       'Offsetkompensierung.
CALL SetFrcCoord (1)   'Stellt die Kraftbegrenzungskoordinate ein.
CALL SetFrcLimit (100, 0, 100, 100, 100, 100)
                       'Stellt die Kraftbegrenzungsrate ein.
CALL SetCompFControl   'Aktiviert die Nachgiebigkeitssteuerungsfunktion.
```

ResetCompControl (Bibliothek) [V1.4 oder später]

Funktion

Deaktiviert die Nachgiebigkeitssteuerungsfunktion (dedizierter Befehl für 6 Achsen).

Format

ResetCompContrl

Erklärung

Deaktiviert eine Funktion zur Nachgiebigkeitssteuerung.

Makrodefinition

<pacman.h> ist erforderlich.

Verwandte Begriffe

SetCompControl

Anmerkungen

(1) Führen Sie diesen Befehl in einer Task mit Robotersteuerbedingungen aus (TAKEARM). Wenn keine Robotersteuerbedingungen gehalten werden, wird die Fehlermeldung “21F7 Arm-Semaphor kann nicht abgerufen werden” angezeigt.

(2) Führen Sie die Bibliothek aus, während der Roboter stoppt. Die Ausführung dieser Bibliothek in einer Pfadbewegung bewirkt eine Ausführung nach dem Stoppen. Wenn nach der Ausführung dieser Bibliothek der Roboter plötzlich stoppt und der Fehler “612* Überstrom” auftritt, während der Roboter in Bewegung ist, stellen Sie 1 für <Einstellwert> ein oder warten Sie nach der Eingabe eines Befehls wie beispielsweise Delay, bis der Roboter stoppt.

(3) Es wird ein Fehler “60f9 Abnormale Operation beim Deaktivieren der Nachgiebigkeitssteuerung” angezeigt, wenn Sie nach einem vorübergehenden Stop während der Ausführung dieser Bibliothek einen Rückschritt ausführen oder das Programm zurücksetzen.

(4) Wenn Sie mit SetEralw die Werte für die zulässige Abweichung ändern, während die Nachgiebigkeitssteuerung aktiviert ist, wird die zulässige Abweichung wieder auf ihre Ausgangswerte zurückgesetzt. Die Werte für die zulässige Abweichung können nicht auf die Ausgangswerte zurückgesetzt werden, wenn während der Ausführung von ResetCompControl ein Fehler auftritt. Initialisieren Sie in diesem Fall mit ResetEralw die zulässigen Abweichungswerte nach dem Inaktivieren der Nachgiebigkeitssteuerung.

(5) Sie erhalten eventuell den Fehler “608* J* Angegebener Geschwindigkeitsgrenzwert überschritten”. Verwenden Sie in diesem Fall aspChange zum Ändern des Modus für die optimale Lastkapazität auf 2 oder 3, bevor Sie ResetCompControl ausführen, und setzen Sie nach der Ausführung den Modus für die optimale Lastkapazität wieder auf den vorigen Wert zurück.

Beispiel

```
CALL ResetCompControl  'Deaktiviert die Nachgiebigkeitssteuerungsfunktion.
CALL ResetEralw        'Initialisiert die Werte für die zulässige Abweichung.
```

SetFrcCoord (Bibliothek) [V1.4 oder später]

Funktion

Wählt ein Koordinatensystem für die Kraftbegrenzung aus (dedizierter Befehl für 6 Achsen).

Format

SetFrcCoord(<Einstellwert>)

Erklärung

Wählt ein Koordinatensystem für die mit SetFrcLimit und SetCompRate angegebenen Kraftbegrenzungswerte aus. Sie können einen eingestellten Wert 0 für das Basiskoordinatensystem, einen eingestellten Wert 1 für das Werkzeugkoordinatensystem und einen eingestellten Wert 2 für das Arbeitskoordinatensystem Ihres Roboters verwenden.

Makrodefinition

<pacman.h> ist erforderlich.

Verwandte Begriffe

SetFrcLimit, SetCompRate, SetFrcCoord, ResetCompControl

Anmerkungen

(1) Führen Sie diesen Befehl in einer Task mit Robotersteuerbedingungen aus (TAKEARM). Wenn keine Robotersteuerbedingungen gehalten werden, wird die Fehlermeldung “21F7 Arm-Semaphor kann nicht abgerufen werden” angezeigt.

(2) Diese Bibliothek ist nicht verfügbar, während die Nachgiebigkeitssteuerung aktiviert ist. Wenn Sie diese Bibliothek unter der Nachgiebigkeitssteuerung ausführen, tritt der Fehler “60fa Nachgiebigkeitssteuerung ist aktiviert” auf.

(3) Wenn Sie 1 für <Einstellwert> zur Auswahl des Werkzeugkoordinatensystems angeben, ist die Werkzeugkoordinate die Werkzeugkoordinate zum Aktivieren der Nachgiebigkeitssteuerung (führt SetCompControl aus). Wenn Sie mit dem changetool-Befehl die Werkzeugkoordinate ändern, während die Nachgiebigkeitssteuerung aktiviert ist, wird die Kraftbegrenzungskoordinate nicht geändert.

(4) Wenn Sie 2 für <Einstellwert> zur Auswahl des Arbeitskoordinatensystems angeben, ist die Arbeitskoordinate die Arbeitskoordinate zum Aktivieren der Nachgiebigkeitssteuerung (fürht SetCompControl aus). Wenn Sie mit dem changetool-Befehl die Arbeitskoordinate ändern, während die Nachgiebigkeitssteuerung aktiviert ist, wird die Kraftbegrenzungskoordinate nicht geändert.

(5) Der Einstellwert wird nach dem Einschalten des Controllers auf 0 (das Basiskoordinatensystem) initialisiert.

Beispiel

```
CALL SetFrcCoord (1)    'Stellt das Koordinatensystem für die Kraftbegrenzung auf
                        'die Werkzeugkoordinate ein.
Changetool 2            'Stellt die Werkzeugkoordinate auf Werkzeug2 ein.
CALL SetFrcLimit (100, 0, 100, 100, 100, 100)
                        'Stellt die Kraftbegrenzungsrate ein.
CALL SetCompControl     'Stellt die Kraftbegrenzungsrate in der Y-Richtung des
                        'Werkzeugkoordinatensystems 2 auf 0% ein und aktiviert
                        'die Nachgiebigkeitssteuerung.
```

SetFrcLimit (Bibliothek) [V1.4 oder später]

Funktion

Stellt die Werte für die Kraftbegrenzung ein (dedizierter Befehl für 6 Achsen).

Format

SetFrcLimit (<Begrenzungsrate entlang X>, <Begrenzungsrate entlang Y>, <Begrenzungsrate entlang Z>, <Begrenzungsrate um X>, <Begrenzungsrate um Y>, <Begrenzungsrate um Z>)
Die Einstellung reicht von 0 bis 100. Es sind bis zu zwei Dezimalstellen gültig.

Erklärung

Stellt die Kraftbegrenzungsraten entlang der X-, Y- und Z-Achse eines mit SetFrcCoord angegebenen Koordinatensystems ein.

Makrodefinition

<pacman.h> ist erforderlich.

Verwandte Begriffe

ResetFrcLimit, SetFrcCoord, SetCompControl

Anmerkungen

(1) Führen Sie diesen Befehl in einer Task mit Robotersteuerbedingungen aus (TAKEARM). Wenn keine Robotersteuerbedingungen gehalten werden, wird die Fehlermeldung “21F7 Arm-Semaphor kann nicht abgerufen werden” angezeigt.

(2) Diese Bibliothek ist nicht verfügbar, während die Nachgiebigkeitssteuerung aktiviert ist. Wenn Sie diese Bibliothek unter der Nachgiebigkeitssteuerung ausführen, tritt der Fehler “60fa Nachgiebigkeitssteuerung ist aktiviert” auf.

(3) Alle eingestellten Werte entlang und um die X-, Y- und Z-Achse werden nach dem Einschalten des Controllers auf 100 initialisiert.

Beispiel

```
CALL SetFrcCoord (1)    'Stellt das Koordinatensystem für die Kraftbegrenzung auf
                        'die Werkzeugkoordinate ein.
CALL SetFrcLimit (100, 0, 100, 100, 100, 100)
                        'Stellt die Kraftbegrenzungsraten ein.
CALL SetCompControl     'Stellt die Kraftbegrenzungsrate in der Y-Richtung des
                        'Werkzeugkoordinatensystems auf 0% ein und aktiviert die
                        'Nachgiebigkeitssteuerung.
```

ResetFrcLimit (Bibliothek) [V1.4 oder später]

Funktion

Initialisiert die Werte für die Kraftbegrenzung (dedizierter Befehl für 6 Achsen).

Format

ResetFrcLimit

Erklärung

Initialisiert die Raten der Kraftbegrenzung. Alle Raten entlang und um die X-, Y- und Z-Achse werden auf 100% eingestellt.

Makrodefinition

<pacman.h> ist erforderlich.

Verwandte Begriffe

SetFrcLimit

Anmerkungen

(1) Führen Sie diesen Befehl in einer Task mit Robotersteuerbedingungen aus (TAKEARM). Wenn keine Robotersteuerbedingungen gehalten werden, wird die Fehlermeldung “21F7 Arm-Semaphor kann nicht abgerufen werden” angezeigt.

(2) Diese Bibliothek ist nicht verfügbar, während die Nachgiebigkeitssteuerung aktiviert ist. Wenn Sie diese Bibliothek unter der Nachgiebigkeitssteuerung ausführen, tritt der Fehler “60fa Nachgiebigkeitssteuerung ist aktiviert” auf.

Beispiel

```
CALL ResetCompControl  'Deaktiviert die Nachgiebigkeitssteuerung.
CALL ResetFrcLimit     'Initialisiert die Kraftbegrenzungsraten.
```

SetCompRate (Bibliothek) [V1.4 oder später]

Funktion

Stellt die Nachgiebigkeitsraten innerhalb der Nachgiebigkeitssteuerung ein (dedizierter Befehl für 6 Achsen).

Format

SetCompRate (<Nachgiebigkeit entlang X>, <Nachgiebigkeit entlang Y>, <Nachgiebigkeit entlang Z>, <Nachgiebigkeit um X>, <Nachgiebigkeit um Y>, <Nachgiebigkeit um Z>)

Erklärung

Stellt die Nachgiebigkeitsraten entlang und um die X-, Y- und Z-Achse eines mit SetFrcCoord angegebenen Koordinatensystems ein.
Die Einstellung reicht von 0 bis 100, wobei 0 einer maximalen Nachgiebigkeit entspricht. Bis zu zwei Dezimalstellen sind gültig.

Makrodefinition

<pacman.h> ist erforderlich.

Verwandte Begriffe

ResetCompRate, SetFrcCoord, SetCompControl

Anmerkungen

(1) Führen Sie diesen Befehl in einer Task mit Robotersteuerbedingungen aus (TAKEARM). Wenn keine Robotersteuerbedingungen gehalten werden, wird die Fehlermeldung “21F7 Arm-Semaphor kann nicht abgerufen werden” angezeigt.

(2) Diese Bibliothek ist nicht verfügbar, während die Nachgiebigkeitssteuerung aktiviert ist. Wenn Sie diese Bibliothek unter der Nachgiebigkeitssteuerung ausführen, tritt der Fehler “60fa Nachgiebigkeitssteuerung ist aktiviert” auf.

(3) Alle eingestellten Werte entlang und um die X-, Y- und Z-Achse werden nach dem Einschalten des Controllers auf 100 initialisiert.

Beispiel

```
CALL SetFrcCoord (1)    'Stellt das Koordinatensystem für die Kraftbegrenzung auf
                        'die Werkzeugkoordinate ein.
CALL SetCompRate (100, 0, 100, 100, 100, 100)
                        'Stellt die Nachgiebigkeitsrate ein.
CALL SetCompControl     'Stellt die Nachgiebigkeitsrate in Y-Richtung des
                        'Werkzeugkoordinatensystems auf 0% ein und aktiviert die
                        'Nachgiebigkeitssteuerung.
```

ResetCompRate (Bibliothek) [V1.4 oder später]

Funktion

Initialisiert die Werte für die Nachgiebigkeit (dedizierter Befehl für 6 Achsen).

Format

ResetCompRate

Erklärung

Initialisiert die Nachgiebigkeitsraten entlang und um die X-, Y- und Z-Achse auf 100%.

Makrodefinition

<pacman.h> ist erforderlich.

Verwandte Begriffe

SetCompRate

Anmerkungen

(1) Führen Sie diesen Befehl in einer Task mit Robotersteuerbedingungen aus (TAKEARM). Wenn keine Robotersteuerbedingungen gehalten werden, wird die Fehlermeldung “21F7 Arm-Semaphor kann nicht abgerufen werden” angezeigt.

(2) Diese Bibliothek ist nicht verfügbar, während die Nachgiebigkeitssteuerung aktiviert ist. Wenn Sie diese Bibliothek unter der Nachgiebigkeitssteuerung ausführen, tritt der Fehler “60fa Nachgiebigkeitssteuerung ist aktiviert” auf.

Beispiel

```
CALL ResetCompControl  'Deaktiviert die Nachgiebigkeitssteuerung.
CALL ResetCompRate     'Initialisiert die Nachgiebigkeitsraten.
```

SetFrcAssist (Bibliothek) [V1.4 oder später]

Funktion

Stellt die Krafthilfe unter der Nachgiebigkeitssteuerung (spezielle Funktionsbibliothek zur Nachgiebigkeitssteuerung) ein (dedizierter Befehl für 6 Achsen).

Format

SetFrcAssist (<Krafthilfe entlang X>, <Krafthilfe entlang Y>, <Krafthilfe entlang Z>, <Momenthilfe um X>, <Momenthilfe um Y>, <Momenthilfe um Z>)

Erklärung

Stellt die Krafthilfe entlang und die Momenthilfe um die X-, Y- und Z-Achse eines mit SetFrcCoord angegebenen Koordinatensystems ein. Der maximale Einstellwert beträgt 10% des maximalen Kraftbegrenzungswertes.
Die Einheit für die Krafteinstellung ist [N]. Die Einheit für die Momenteinstellung ist [Nm]. Es ist maximal eine Dezimalstelle zulässig.

Makrodefinition

<pacman.h> ist erforderlich.

Verwandte Begriffe

ResetFrcAssist, SetFrcCoord, SetCompControl

Anmerkungen

(1) Führen Sie diesen Befehl in einer Task mit Robotersteuerbedingungen aus (TAKEARM). Wenn keine Robotersteuerbedingungen gehalten werden, wird die Fehlermeldung “21F7 Arm-Semaphor kann nicht abgerufen werden” angezeigt.

(2) Der Roboter bewegt sich eventuell in die Richtung, in die die Krafthilfe und die Momenthilfe wirken. Reduzieren Sie in diesem Fall die eingestellten Werte.

(3) Alle eingestellten Werte entlang und um die X-, Y- und Z-Achse werden nach dem Einschalten des Controllers auf 0 initialisiert.

Beispiel

```
CALL SetFrcCoord (1)    'Stellt das Koordinatensystem für die Kraftbegrenzung auf
                        'die Werkzeugkoordinate ein.
CALL SetFrcAssist (-30, 0, 0, 0, 0, 0)
                        'Stellt die Krafthilfe auf 30 [N] in -X-Richtung ein
CALL SetFrcLimit (0, 100, 100, 100, 100, 100)
                        'Stellt die Kraftbegrenzungsraten ein.
CALL SetCompControl     'Aktiviert die Nachgiebigkeitssteuerungsfunktion.
                        'Kraftbegrenzung in X-Richtung ist 0%, und eine Kraft von
                  '30 [N] wird in -X-Richtung angelegt
```

ResetFrcAssist (Bibliothek) [V1.4 oder später]

Funktion

Initialisiert die Krafthilfe (spezielle Funktionsbibliothek zur Nachgiebigkeitssteuerung) (dedizierter Befehl für 6 Achsen).

Format

ResetFrcAssisit

Erklärung

Initialisiert die Krafthilfe entlang und die Momenthilfe um die X-, Y- und Z-Achse eines mit SetFrcCoord angegebenen Koordinatensystems auf 100%.

Makrodefinition

<pacman.h> ist erforderlich.

Verwandte Begriffe

SetFrcAssist

Anmerkungen

(1) Führen Sie diesen Befehl in einer Task mit Robotersteuerbedingungen aus (TAKEARM). Wenn keine Robotersteuerbedingungen gehalten werden, wird die Fehlermeldung “21F7 Arm-Semaphor kann nicht abgerufen werden” angezeigt.

Beispiel

```
CALL ResetCompControl  'Deaktiviert die Nachgiebigkeitssteuerung.
CALL ResetFrcAssist    'Initialisiert die Krafthilfe und die Momenthilfe.
```

SetCompJLimit (Bibliothek) [V1.4 oder später]

Funktion

Stellt den Stromgrenzwert unter der Nachgiebigkeitssteuerung (spezielle Funktionsbibliothek zur Nachgiebigkeitssteuerung) ein (dedizierter Befehl für 6 Achsen).

Format

SetCompJLimit (<J1 Stromgrenzwert>, <J2 Stromgrenzwert>, <J3 Stromgrenzwert>, <J4 Stromgrenzwert>, <J5 Stromgrenzwert>, <J6 Stromgrenzwert>)

Erklärung

Legt den aktuellen Grenzwert unter der Nachgiebigkeitssteuerung fest. Der angegebene Strom eines Motors entspricht 100. Wenn Sie mit SetFrcLimit den Wert 0 für alle Richtungen festlegen, ist der Motorstrom auf einen niedrigeren Wert als die Einstellung begrenzt.
Die Einstellung reicht von 0 bis 100.

Makrodefinition

<pacman.h> ist erforderlich.

Verwandte Begriffe

ResetCompJLimit, SetCompControl

Anmerkungen

(1) Führen Sie diesen Befehl in einer Task mit Robotersteuerbedingungen aus (TAKEARM). Wenn keine Robotersteuerbedingungen gehalten werden, wird die Fehlermeldung “21F7 Arm-Semaphor kann nicht abgerufen werden” angezeigt.

(2) Die Einstellwerte und alle Stromgrenzwerte werden nach dem Einschalten des Controllers auf 0 initialisiert.

(3) Wenn Sie mit SetCompJLimit relativ große Werte einstellen, weist Ihr Roboter eventuell eine Oszillation auf, die zu einem Fehler führt. Verwenden Sie in diesem Fall SetCompRate und SetDumpRate zum Anpassen der Nachgiebigkeit.

Beispiel

```
CALL SetFrcCoord (1)    'Stellt das Koordinatensystem für die Kraftbegrenzung auf
                        'die Werkzeugkoordinate ein.
CALL SetCompJLimit (30, 0, 0, 0, 0, 0)
                        'Stellt den Stromgrenzwert für J1 auf 30% ein.
CALL SetFrcLimit (0, 100, 100, 100, 100, 100)
                        'Stellt die Kraftbegrenzungsraten ein.
CALL SetCompControl     'Aktiviert die Nachgiebigkeitssteuerungsfunktion.
```

ResetCompJLimit (Bibliothek) [V1.4 oder später]

Funktion

Initialisiert den Stromgrenzwert unter der Nachgiebigkeitssteuerung (spezielle Funktionsbibliothek zur Nachgiebigkeitssteuerung) (dedizierter Befehl für 6 Achsen).

Format

ResetCompJLimit

Erklärung

Initialisiert den Stromgrenzwert unter der Nachgiebigkeitssteuerung und setzt alle Achsen auf 0.

Makrodefinition

<pacman.h> ist erforderlich.

Verwandte Begriffe

SetCompJLimit, SetCompControl

Anmerkungen

(1) Führen Sie diesen Befehl in einer Task mit Robotersteuerbedingungen aus (TAKEARM). Wenn keine Robotersteuerbedingungen gehalten werden, wird die Fehlermeldung “21F7 Arm-Semaphor kann nicht abgerufen werden.” angezeigt.

Beispiel

```
CALL ResetCompControl  'Deaktiviert die Nachgiebigkeitssteuerung.
CALL ResetCompJLimit   'Initialisiert die Stromgrenzwerte.
```

SetCompVMode (Bibliothek) [V1.4 oder später]

Funktion

Stellt den Geschwindigkeitssteuermodus unter der Nachgiebigkeitssteuerung (spezielle Funktionsbibliothek zur Nachgiebigkeitssteuerung) ein (dedizierter Befehl für 6 Achsen).

Format

SetCompVMode

Erklärung

Aktiviert den Steuermodus für die Nachgiebigkeitsgeschwindigkeit bei der Ausführung von SetCompControl.

Makrodefinition

<pacman.h> ist erforderlich.

Verwandte Begriffe

ResetCompVMode

Anmerkungen

(1) Führen Sie diesen Befehl in einer Task mit Robotersteuerbedingungen aus (TAKEARM). Wenn keine Robotersteuerbedingungen gehalten werden, wird die Fehlermeldung “21F7 Arm-Semaphor kann nicht abgerufen werden.” angezeigt.

Beispiel

```
CALL SetCompVMode       'Aktiviert den Steuermodus für die
                        'Nachgiebigkeitsgeschwindigkeit.
CALL SetCompControl     'Aktiviert die Nachgiebigkeitssteuerung.
```

ResetCompVMode (Bibliothek) [V1.4 oder später]

Funktion

Deaktiviert den Geschwindigkeitssteuermodus unter der Nachgiebigkeitssteuerung (spezielle Funktionsbibliothek zur Nachgiebigkeitssteuerung) (dedizierter Befehl für 6 Achsen).

Format

ResetCompVMode

Erklärung

Deaktiviert den Steuermodus für die Nachgiebigkeitsgeschwindigkeit bei der Ausführung von SetCompControl.

Makrodefinition

<pacman.h> ist erforderlich.

Verwandte Begriffe

SetCompVMode

Anmerkungen

(1) Führen Sie diesen Befehl in einer Task mit Robotersteuerbedingungen aus (TAKEARM). Wenn keine Robotersteuerbedingungen gehalten werden, wird die Fehlermeldung “21F7 Arm-Semaphor kann nicht abgerufen werden.” angezeigt.

Beispiel

```
CALL ResetCompVMode     'Deaktiviert den Steuermodus für die
                        'Nachgiebigkeitsgeschwindigkeit.
CALL SetCompControl     'Aktiviert die Nachgiebigkeitssteuerung.
```

SetCompEralw (Bibliothek) [V1.4 oder später]

Funktion

Stellt die Werte für die zulässige Abweichung für Position und Haltung der Werkzeugspitze unter der Nachgiebigkeitssteuerung ein (dedizierter Befehl für 6 Achsen).

Format

SetCompEralw (<Zulässige Abweichung entlang X>, <Zulässige Abweichung Y>, <Zulässige Abweichung Z>, <Zulässige Abweichung X>, <Zulässige Abweichung Y>, <Zulässige Abweichung Z>)

Erklärung

Gibt die Werte für die zulässige Abweichung der Position und der Haltung der Tool-Spitze unter der Nachgiebigkeitssteuerung an. Die Einheit für die zulässige Abweichung entlang X, Y und Z ist mm, und die Einheit für die zulässige Abweichung um X, Y und Z ist Grad. Es ist maximal eine Dezimalstelle zulässig.

Makrodefinition

<pacman.h> ist erforderlich.

Verwandte Begriffe

ResetCompEralw, SetFrcCoord

Anmerkungen

(1) Führen Sie diesen Befehl in einer Task mit Robotersteuerbedingungen aus (TAKEARM). Wenn keine Robotersteuerbedingungen gehalten werden, wird die Fehlermeldung “21F7 Arm-Semaphor kann nicht abgerufen werden.” angezeigt.

(2) Die Anfangswerte einer zulässigen Abweichung sind 100 mm entlang X, Y und Z sowie 30 Grad um X, Y und Z. Der maximale Wert um die X-, Y- und Z-Achse beträgt 175 Grad. Wenn Sie Werte eingeben, die den Maximalwert übersteigen, tritt der Fehler “6003 Gültiger Bereich wurde überschritten” auf.

(3) Wenn die Position oder die Haltungsabweichung der Werkzeugspitze den zulässigen Wert übersteigt, tritt der Fehler “60f8 Fehler bei Positionsabweichung unter Nachgiebigkeitssteuerung” auf.

(4) Das zum Einstellen der Abweichung verwendete Koordinatensystem ist das mit SetFrcCoord angegebene. Wenn Sie es als SetFrcCoord (1) eingeben, ist das Einstellungskoordinatensystem für SetCompEralw das Werkzeugkoordinatensystem.

Beispiel

```
CALL SetFrcCoord (2)    'Stellt das Koordinatensystem für die Kraftbegrenzung auf
                        'das Arbeitskoordinatensystem ein.
CALL SetFrcLimit (100, 0, 100, 100, 100, 100)
                        'Stellt die Kraftbegrenzungsrate in Y-Richtung des
                        'Arbeitskoordinatensystems auf 0% ein.
CALL SetCompEralw (10, 150, 10, 5, 5, 5)
                        'Stellt die Werte für die zulässige Abweichung entlang X
                        'und Z der Arbeitskoordinate auf 10 mm, entlang Y auf
                        '150 mm und um X, Y und Z auf 5 Grad ein.
```

ResetCompEralw (Bibliothek) [V1.4 oder später]

Funktion

Initialisiert die Werte für die zulässige Abweichung für Position und Haltung der Werkzeugspitze unter der Nachgiebigkeitssteuerung (dedizierter Befehl für 6 Achsen).

Format

ResetCompEralw

Erklärung

Initialisiert die Werte für die zulässige Abweichung der Position und der Haltung der Tool-Spitze unter der Nachgiebigkeitssteuerung. Die Anfangswerte für die zulässige Abweichung lauten 100 mm entlang X, Y und Z und 30 Grad um X, Y und Z.

Makrodefinition

<pacman.h> ist erforderlich.

Verwandte Begriffe

SetCompEralw

Anmerkungen

(1) Führen Sie diesen Befehl in einer Task mit Robotersteuerbedingungen aus (TAKEARM). Wenn keine Robotersteuerbedingungen gehalten werden, wird die Fehlermeldung “21F7 Arm-Semaphor kann nicht abgerufen werden.” angezeigt.

Beispiel

```
CALL ResetCompEralw
```

SetDampRate (Bibliothek) [V1.4 oder später]

Funktion

Stellt die Dämpfungsraten unter der Nachgiebigkeitssteuerung ein (dedizierter Befehl für 6 Achsen).

Format

SetDampRate (<Dämpfungsrate entlang X>, <Dämpfungsrate entlang Y>, <Dämpfungsrate entlang Z>, <Dämpfungsrate um X>, <Dämpfungsrate um Y>, <Dämpfungsrate um Z>)

Erklärung

Stellt die Dämpfungsraten entlang und um die X-, Y- und Z-Achse eines mit SetFrcCoord angegebenen Koordinatensystems ein.
Die Einstellung reicht von 0 bis 100, wobei 0 einer maximalen Dämpfung entspricht. Bis zu zwei Dezimalstellen sind gültig.

Makrodefinition

<pacman.h> ist erforderlich.

Verwandte Begriffe

ResetDampRate, SetFrcCoord, SetCompRate

Anmerkungen

(1) Führen Sie diesen Befehl in einer Task mit Robotersteuerbedingungen aus (TAKEARM). Wenn keine Robotersteuerbedingungen gehalten werden, wird die Fehlermeldung “21F7 Arm-Semaphor kann nicht abgerufen werden.” angezeigt.

(2) Diese Bibliothek ist nicht verfügbar, während die Nachgiebigkeitssteuerung aktiviert ist. Wenn Sie diese Bibliothek unter der Nachgiebigkeitssteuerung ausführen, tritt der Fehler “60fa Nachgiebigkeitssteuerung ist aktiviert” auf.

(3) Alle eingestellten Werte entlang und um die X-, Y- und Z-Achse werden nach dem Einschalten des Controllers auf 100 initialisiert.

(4) Führen Sie diese Funktion mit einer Task aus, die das Recht zur Robotersteuerung abruft (TAKEARM). Wenn das Recht zur Robotersteuerung nicht abgerufen wird, wird die Fehlermeldung “21F7 Arm-Semaphor kann nicht abgerufen werden.” angezeigt.

(5) Wenn Sie SetCompRate nach SetDampRate ausführen, werden die Nachgiebigkeitsraten auf die mit SetCompRate festgelegte Einstellung geändert.

Beispiel

```
CALL SetFrcCoord (1)    'Stellt das Koordinatensystem für die Kraftbegrenzung auf
                        'das Werkzeugkoordinatensystem ein.
CALL SetCompRate (100, 0, 100, 100, 100, 100)
                        'Stellt die Nachgiebigkeitsrate ein.
CALL SetDampRate (100, 20, 100, 100, 100, 100)
                        'Stellt die Dämpfungsrate ein.
CALL SetCompControl     'Stellt die Nachgiebigkeitsrate und die Dämpfungsrate in
                        'Y-Richtung des Werkzeugkoordinatensystems auf 0% bzw.
                        '20% ein und aktiviert die Nachgiebigkeitssteuerung.
```

ResetDampRate (Bibliothek) [V1.4 oder später]

Funktion

Initialisiert die Dämpfungsraten unter der Nachgiebigkeitssteuerung (dedizierter Befehl für 6 Achsen).

Format

ResetDampRate

Erklärung

Initialisiert die Dämpfungsraten entlang und um die X-, Y- und Z-Achse auf 100%.

Makrodefinition

<pacman.h> ist erforderlich.

Verwandte Begriffe

SetDampRate

Anmerkungen

(1) Führen Sie diesen Befehl in einer Task mit Robotersteuerbedingungen aus (TAKEARM). Wenn keine Robotersteuerbedingungen gehalten werden, wird die Fehlermeldung “21F7 Arm-Semaphor kann nicht abgerufen werden” angezeigt.

(2) Diese Bibliothek ist nicht verfügbar, während die Nachgiebigkeitssteuerung aktiviert ist. Wenn Sie diese Bibliothek unter der Nachgiebigkeitssteuerung ausführen, tritt der Fehler “60fa Nachgiebigkeitssteuerung ist aktiviert” auf.

Beispiel

```
CALL ResetDampRate     'Initialisiert die Dämpfungsraten.
```

SetVibControl (Bibliothek) [V1.4 oder später]

Funktion

Stellt den Steuermodus für die Reduzierung der Restvibration ein (dedizierter Befehl für 6 Achsen).

Format

SetVibControl

Erklärung

Wechselt zum Steuermodus für die Reduzierung der Restvibration und reduziert die Restvibration, wenn der Roboter in Bewegung ist.

Makrodefinition

<pacman.h> ist erforderlich.

Verwandte Begriffe

ResetVibControl

Anmerkungen

(1) Führen Sie diesen Befehl in einer Task mit Robotersteuerbedingungen aus (TAKEARM). Wenn keine Robotersteuerbedingungen gehalten werden, wird die Fehlermeldung “21F7 Arm-Semaphor kann nicht abgerufen werden” angezeigt.

(2) Wenn Sie SetVibControl während der Verwendung von SetEralw zum Ändern der zulässigen Winkelabweichung verwenden, wird die zulässige Winkelabweichung initialisiert. Verwenden Sie SetEralw zum Einstellen der zulässigen Winkelabweichung erneut.

(3) Die Pfadgenauigkeit wird wegen der nachfolgenden Verzögerung im Steuermodus zur Reduzierung der Restvibration verringert. Vermeiden Sie diesen Modus, wenn die Pfadgenauigkeit erforderlich ist.

(4) Wenn Sie SetVibControl während einer Pfadbewegung ausführen, wird die Bibliothek nach dem Stoppen der Bewegung ausgeführt. Somit ist dies eine Ausführung nach dem Stoppen.

Beispiel

```
CALL SetVibControl      'Stellt den Steuermodus für die Reduzierung der
                        'Restvibration ein.
```

ResetVibControl (Bibliothek) [V1.4 oder später]

Funktion

Kehrt vom Steuermodus zur Reduzierung der Restvibration zurück in den normalen Steuermodus.

Format

ResetVibConrol

Erklärung

Kehrt vom Steuermodus zur Reduzierung der Restvibration zurück in den normalen Steuermodus.

Makrodefinition

<pacman.h> ist erforderlich.

Verwandte Begriffe

ResetVibControl

Anmerkungen

(1) Führen Sie diesen Befehl in einer Task mit Robotersteuerbedingungen aus (TAKEARM). Wenn keine Robotersteuerbedingungen gehalten werden, wird die Fehlermeldung “21F7 Arm-Semaphor kann nicht abgerufen werden” angezeigt.

(2) Wenn Sie ResetVibControl während der Verwendung von SetEralw zum Ändern der zulässigen Winkelabweichung verwenden, wird die zulässige Winkelabweichung initialisiert. Verwenden Sie SetEralw zum Einstellen der zulässigen Winkelabweichung erneut.

(3) Wenn Sie ResetVibControl während einer Pfadbewegung ausführen, wird die Bibliothek nach dem Stoppen der Bewegung ausgeführt. Somit ist dies eine Ausführung nach dem Stoppen.

Beispiel

```
CALL ResetVibControl   'Stellt den normalen Steuermodus ein.
```

22.6 Bildverarbeitung

viTran6 (Bibliothek)

Funktion

Wandelt die Bildverarbeitungskoordinaten in Roboterkoordinaten um (für 6 Achsen).

Format

viTran6 (<Kalibrierungsnummer>, <Bildverarbeitungskoordinate X>, <Bildverarbeitungskoordinate Y>, <Roboterkoordinate (Typ P)>)

Erklärung

Diese Anweisung wandelt die Bildverarbeitungskoordinaten (X, Y) auf die absoluten Roboterkoordinaten (X, Y, Z) um.
Zur Ausführung dieser Bibliothek muß die Bildverarbeitungskalibrierung ausgeführt und die Bildverarbeitungskalibrierungsdaten mit dem Controller übertragen werden.
Geben Sie eine Nummer an, die die Bildverarbeitungskalibrierung als Kalibrierungsnummer ausführt.
Mit dieser Koordinatenumwandlung können nur die absoluten Koordinaten (X, Y, Z) abgerufen werden Die Abbildungsdaten müssen separat eingestellt werden, um die MOVE-Anweisung ausführen zu können.

Beispiel

```
                    'Speichert die Bildverarbeitungskoordinate X in F1.
                    'Speichert die Bildverarbeitungskoordinate Y in F2.
TAKEARM
CHANGETOOL 1        'Stellt das Werkzeug ein, für das eine
                    'Bildverarbeitungskalibrierung durchgeführt wurde.
MOVE P,P10          'Bewegung zur Standby-Position.
P11 = P10           'Kopiert die Armstellung an der Standby-Position.
CALL viTran6(0, F1, F2, P11)
                    'Wandelt die Bildverarbeitungskoordinaten in
                    'Roboterkoordinaten um(“0” der Bildverarbeitungskalibrierung
                    'wird verwendet).
APPROACH P, P11, 100
MOVE L, P11         'Bewegung an die mit der Bildverarbeitung gemessenen Position.
DEPART L,100
CHANGETOOL 0
```

Anhänge

1 Zeichencodetabelle

Tabelle 1. Zeichencodetabelle

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
00																
10																
20		!	”	#	$	~	&	’	(	)	*	+	,	-	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	X	y	z	{	|	}	~	
80																
90																
A0																
B0																
C0																
D0																
E0																
F0																

2 Abbildungen von Schulter, Ellbogen und Handgelenk

[1] Verfügbare 32 Armstellungen

Ein Roboter mit 6 Achsen kann am Ende des Endwerkzeugs durch unterschiedliche Armstellungen von Schulter, Ellbogen, Handgelenk, 6. Achse und 4. Achse bestimmte Positionen und Stellungen (X, Y, Z, RX, RY und RZ) annehmen.

In den Abbildungen 1 bis 5 sind die unterschiedlichen Armstellungen, die der Roboter für Schulter, Ellbogen, Handgelenk, 6. Achse und 4. Achse annehmen kann, dargestellt.

Durch Kombination dieser unterschiedlichen Armstellungen kann der Roboter für eine bestimmte Position und Stellung 32 unterschiedliche Armstellungen annehmen, wie dies in Tabelle 2 aufgelistet ist.

In Abbildung 6 sind Beispiele von acht möglichen Kombinationen der Armstellungen bei Schulter, Ellbogen und Handgelenk beim Roboter der Serie VS-D, dargestellt.

Tabelle 2. Verfügbare Armstellungen

Wert	Armstellung 4. Achse	Armstellung 6. Achse	Handgelenkstel-lung	Ellbogenstellung	Schulterstellung
0	EINFACH 4	EINFACH	KIPPEN	OBEN	RECHTS
1	EINFACH 4	EINFACH	KIPPEN	OBEN	LINKS
2	EINFACH 4	EINFACH	KIPPEN	UNTEN	RECHTS
3	EINFACH 4	EINFACH	KIPPEN	UNTEN	LINKS
4	EINFACH 4	EINFACH	NICHT KIPPEN	OBEN	RECHTS
5	EINFACH 4	EINFACH	NICHT KIPPEN	OBEN	LINKS
6	EINFACH 4	EINFACH	NICHT KIPPEN	UNTEN	RECHTS
7	EINFACH 4	EINFACH	NICHT KIPPEN	UNTEN	LINKS
8	EINFACH 4	DOPPELT	KIPPEN	OBEN	RECHTS
9	EINFACH 4	DOPPELT	KIPPEN	OBEN	LINKS
10	EINFACH 4	DOPPELT	KIPPEN	UNTEN	RECHTS
11	EINFACH 4	DOPPELT	KIPPEN	UNTEN	LINKS
12	EINFACH 4	DOPPELT	NICHT KIPPEN	OBEN	RECHTS
13	EINFACH 4	DOPPELT	NICHT KIPPEN	OBEN	LINKS
14	EINFACH 4	DOPPELT	NICHT KIPPEN	UNTEN	RECHTS
15	EINFACH 4	DOPPELT	NICHT KIPPEN	UNTEN	LINKS
16	DOPPELT 4	EINFACH	KIPPEN	OBEN	RECHTS
17	DOPPELT 4	EINFACH	KIPPEN	OBEN	LINKS
18	DOPPELT 4	EINFACH	KIPPEN	UNTEN	RECHTS
19	DOPPELT 4	EINFACH	KIPPEN	UNTEN	LINKS
20	DOPPELT 4	EINFACH	NICHT KIPPEN	OBEN	RECHTS
21	DOPPELT 4	EINFACH	NICHT KIPPEN	OBEN	LINKS
22	DOPPELT 4	EINFACH	NICHT KIPPEN	UNTEN	RECHTS
23	DOPPELT 4	EINFACH	NICHT KIPPEN	UNTEN	LINKS
24	DOPPELT 4	DOPPELT	KIPPEN	OBEN	RECHTS
25	DOPPELT 4	DOPPELT	KIPPEN	OBEN	LINKS
26	DOPPELT 4	DOPPELT	KIPPEN	UNTEN	RECHTS
27	DOPPELT 4	DOPPELT	KIPPEN	UNTEN	LINKS
28	DOPPELT 4	DOPPELT	NICHT KIPPEN	OBEN	RECHTS
29	DOPPELT 4	DOPPELT	NICHT KIPPEN	OBEN	LINKS
30	DOPPELT 4	DOPPELT	NICHT KIPPEN	UNTEN	RECHTS
31	DOPPELT 4	DOPPELT	NICHT KIPPEN	UNTEN	LINKS

(1) Schulterstellung

Die Schulterstellung wird durch eine Kombination der Koordinatenkomponenten für die 1., 2. und 3. Achse definiert.

Der Roboter kann zwei verschiedene Schulterstellungen annehmen: Hand links (LINKS) und Hand rechts (RECHTS). (J1 bis J6 stehen für Gelenk 1 bis Gelenk 6.)

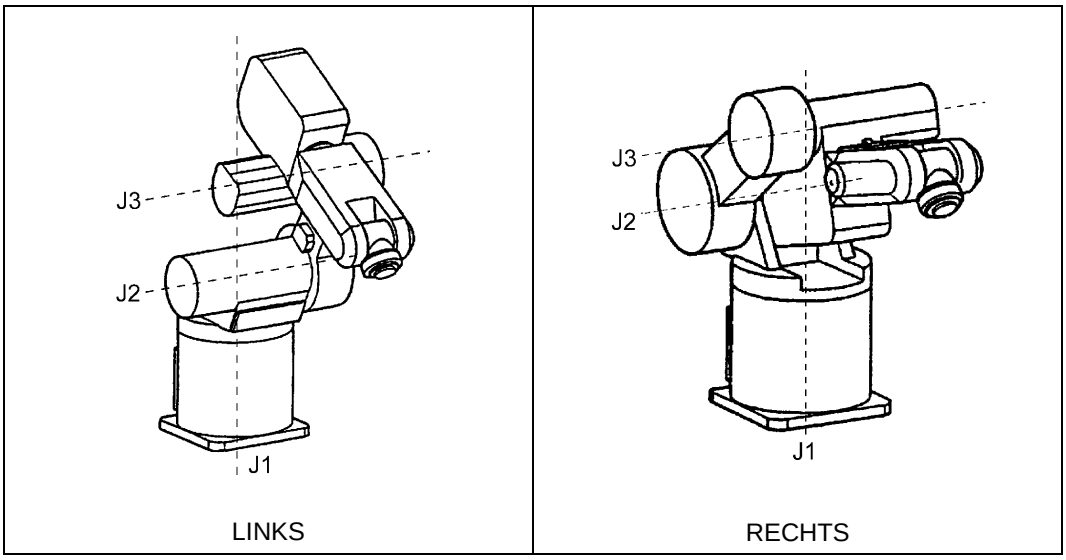


Abbildung 1. Schulterstellung

(2) Ellbogenstellung

Die Ellbogenstellung wird durch eine Kombination der Koordinatenkomponenten der 2. und 3. Achse definiert.

Der Roboter kann zwei verschiedene Ellbogen-Armstellungen annehmen: “Hand oben” (OBEN) und “Hand unten” (UNTEN). (J1 bis J6 stehen für Gelenk 1 bis Gelenk 6.)

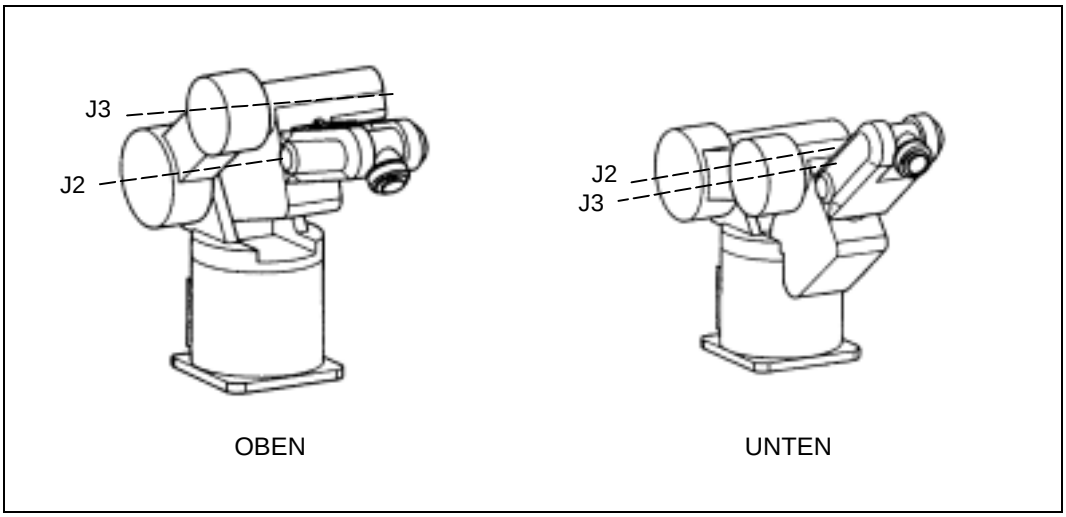


Abbildung 2. Ellbogenstellung

(3) Handgelenkstellung

Die Handgelenkstellung wird durch eine Kombination der Koordinatenkomponenten der 4. und 5. Achse definiert.

Der Roboter kann zwei verschiedene Handgelenkstellungen annehmen: “Normal” (NICHT KIPPEN) und “Invertiert” (KIPPEN). In der Armstellung NICHT KIPPEN ist die 4. Achse um 180° gedreht, ohne die Handgelenkstellung zu ändern. (J1 bis J6 stehen für Gelenk 1 bis Gelenk 6.)

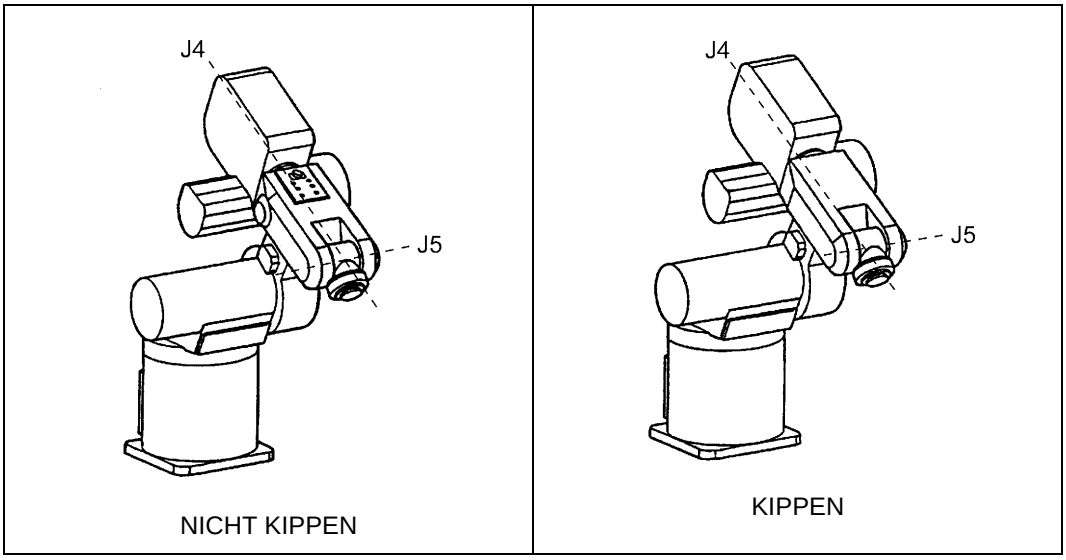


Abbildung 3. Handgelenkstellung

(4) Armstellung der 6. Achse

Die Armstellung der 6. Achse ist definiert durch den Wert der Komponente der 6. Achse.

Der Roboter kann zwei verschiedene Armstellungen der 6. Achse annehmen: EINFACH und DOPPELT. Wenn sich die 6. Achse um $-180° < \theta_6 \leq 180°$ in den mechanischen Schnittstellenkoordinaten dreht, ist die Armstellung EINFACH; dreht sie sich um $180° < \theta_6 \leq 360°$ oder $-360° < \theta_6 \leq -180°$, ist die Armstellung DOPPELT.

Der Roboter nimmt verschiedene Armstellungen an, wenn θ_6 180° oder 181° ist. Gehen Sie beim Ändern der Positionsdaten für die Armstellung der 6. Achse sehr vorsichtig vor. Angenommen, Sie wollen die Armstellung der 6. Achse an θ_6=181° ändern, verwendet der Roboter die Armstellung der 6. Achse an θ_6=-179°, wenn Sie keine Änderung der Armstellung vornehmen.

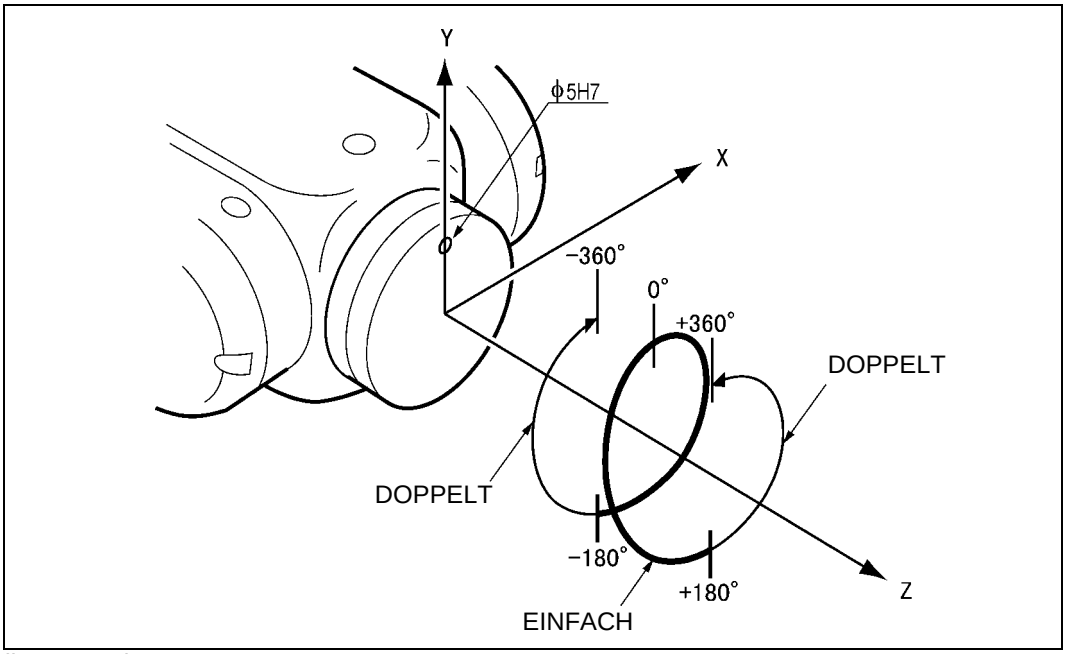


Abbildung 4. Armstellung 6. Achse

(5) Armstellung der 4. Achse

Die Armstellung der 4. Achse ist definiert durch den Wert der Komponente der 4. Achse.

Der Roboter kann zwei verschiedene Armstellungen der 4. Achse annehmen: EINFACH 4 und DOPPELT 4. Wenn sich die 4. Achse um $-180° < \theta_4 \leq 180°$ in den mechanischen Schnittstellenkoordinaten dreht, ist die Armstellung EINFACH 4; dreht sie sich um $180° < \theta_4 \leq 185°$ oder $-185° < \theta_4 \leq -180°$, ist die Armstellung DOPPELT 4.

Der Roboter nimmt verschiedene Armstellungen an, wenn θ_4 180° oder 181° ist. Gehen Sie beim Ändern der Positionsdaten für die Armstellung der 6. Achse sehr vorsichtig vor. Angenommen, Sie wollen die Armstellung der 4. Achse an θ_4=181° ändern, verwendet der Roboter die Armstellung der 4. Achse an θ_4=-179°, wenn Sie keine Änderung der Armstellung vornehmen.

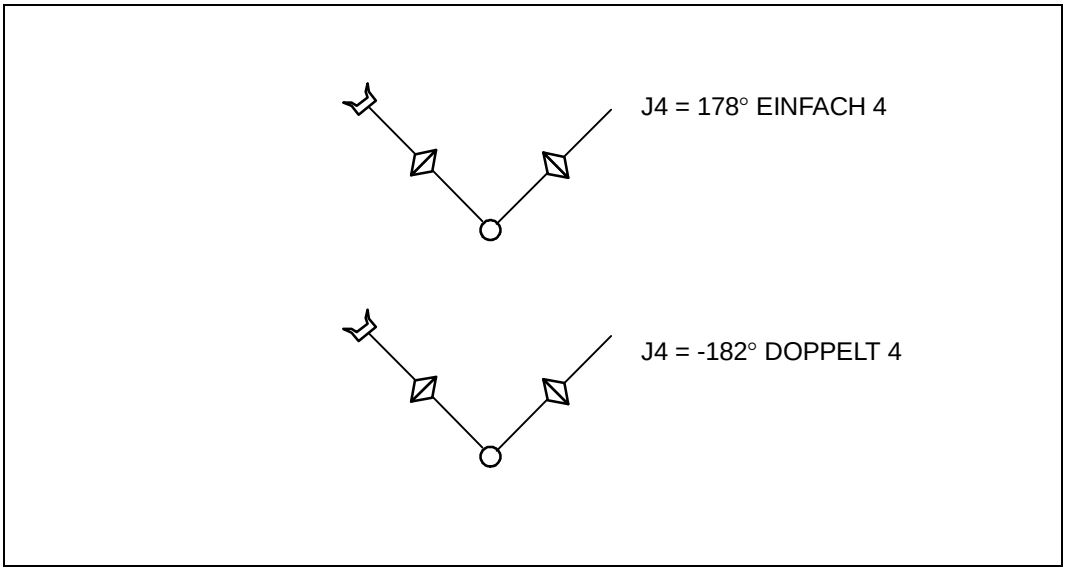
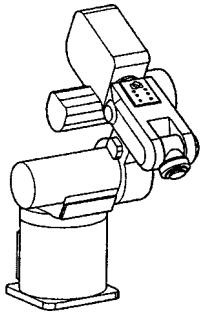
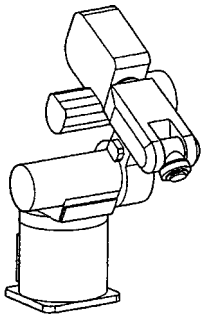


Abbildung 5. Armstellung 4. Achse

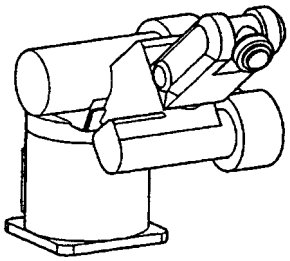
Armstellung-1
LINKS, OBEN und NICHT KIPPEN



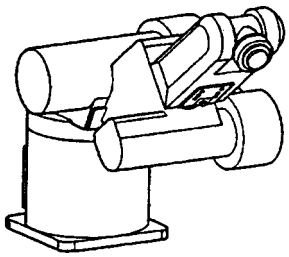
Armstellung-2
LINKS, OBEN und KIPPEN



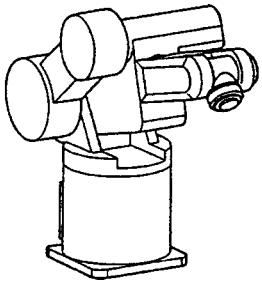
Armstellung-3
LINKS, UNTEN und NICHT KIPPEN



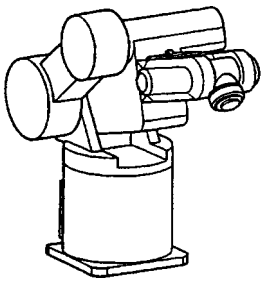
Armstellung-4
LINKS, UNTEN und KIPPEN



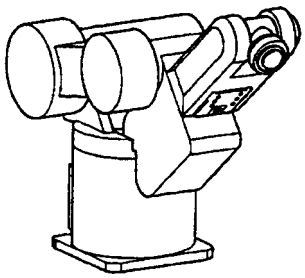
Armstellung-5
RECHTS, OBEN und NICHT KIPPEN



Armstellung-6
RECHTS, OBEN und KIPPEN



Armstellung-7
RECHTS, UNTEN und NICHT KIPPEN



Armstellung-8
RECHTS, UNTEN und KIPPEN

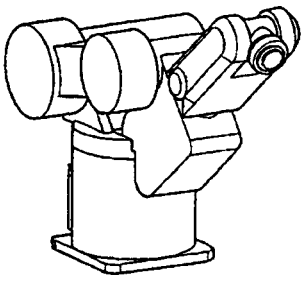
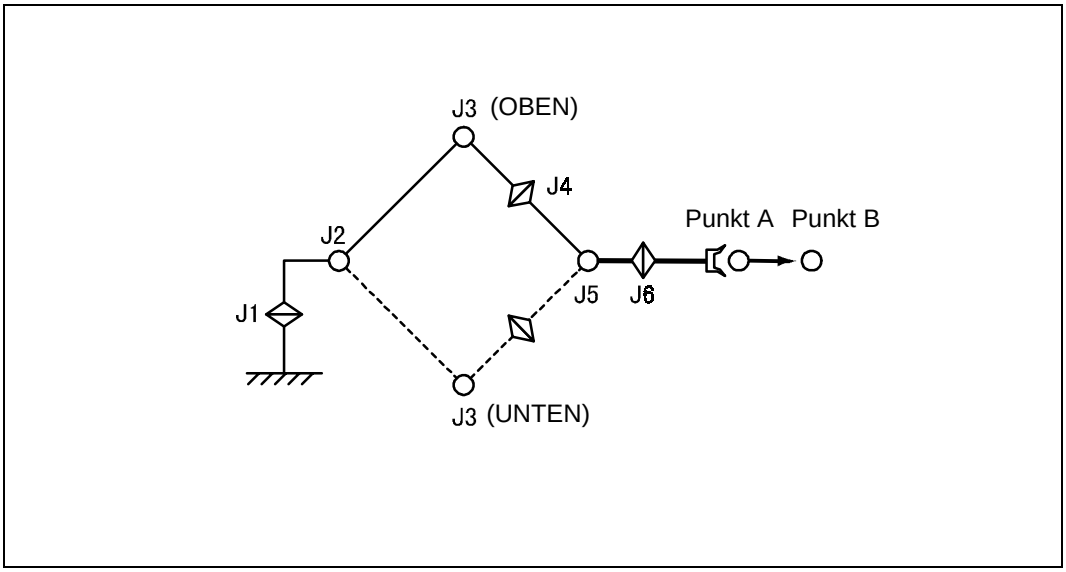


Abbildung 6. Mögliche Kombinationen von Armstellungen von Schulter, Ellbogen und Handgelenk des Roboters

⚠**VORSICHT:** Wenn Sie einen Befehl mit der Linearsteuerung ausführen und sich die Armstellungen des Roboters am Startpunkt von den bei der Programmierung oder beim Einlernen gespeicherten Werten unterscheiden, müssen Sie zunächst sicherstellen, daß kein Teil des Roboters mit umgebenden Geräten oder Einrichtungen kollidiert. Dies ist erforderlich, da jedes Gelenk des Roboters die in Abhängigkeit von den momentanen Armstellungen geeigneten Bewegungen durchführt, um die Spitze des Endwerkzeugs den Objektpunkt erreichen zu lassen, auch wenn die Roboterposition und -stellung am Startpunkt von der Stellung im Programm oder bei der Programmierung nicht abweicht. Der Pfad des Endwerkzeugs ist jedoch praktisch identisch, obwohl die Armstellungen unterschiedlich sein können.

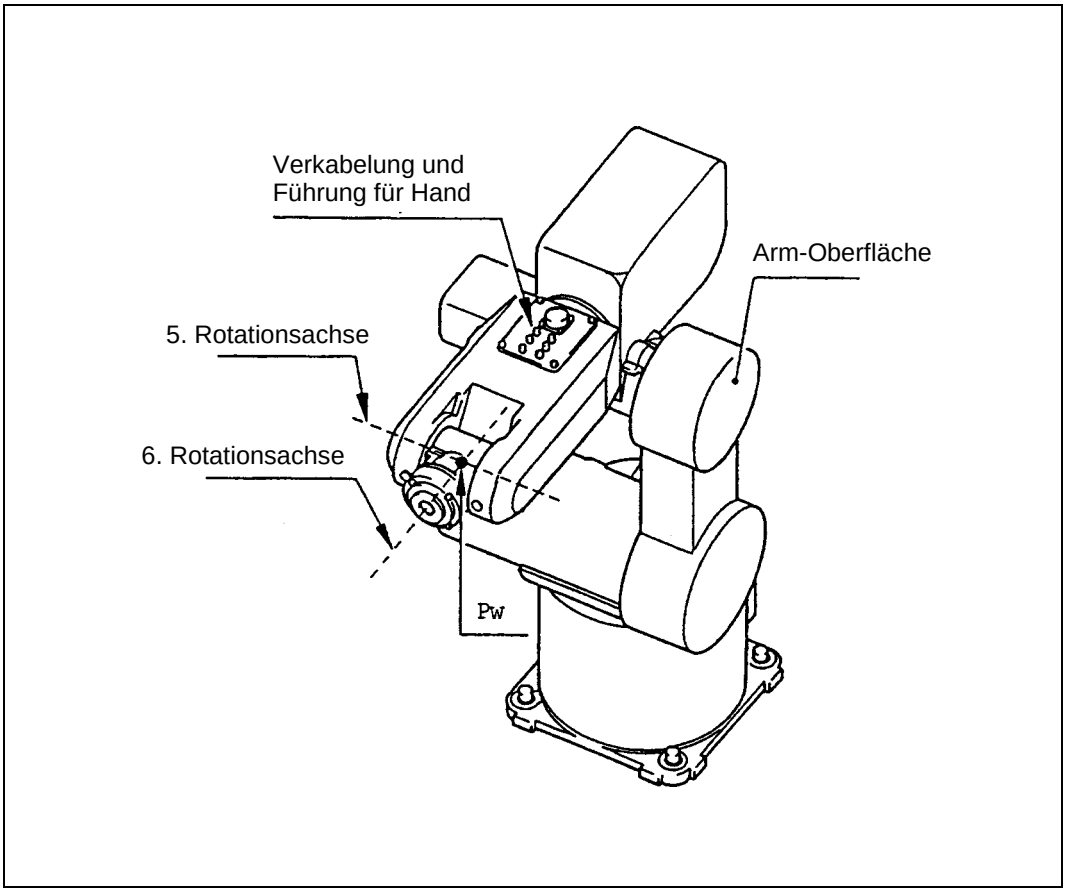


VORSICHT: Durch die Konstruktion des Roboters bedingt, sind eventuell nicht alle 32 Armstellungen in jeder Position und Stellung des Roboters möglich. In bestimmten Fällen kann – je nach Punkt und Stellung – nur die Armstellung LINKS / OBEN / NICHT KIPPEN zulässig sein. (In der Praxis kann der Roboter fast immer nicht alle theoretisch möglichen Armstellungen annehmen, sondern es sind nur zwei Armstellungen möglich: LINKS / OBEN / NICHT KIPPEN und LINKS / OBEN / KIPPEN. Für die Armstellung der 4. Achse nimmt der Roboter EINFACH 4 an.)

[2] Grenzen der Roboter-Armstellungen

Dieser Abschnitt beschreibt die Grenzen für die einzelnen Armstellungen des Roboters (Schulter, Ellbogen, Handgelenk und 6. Achse).

Bei der Beurteilung der Grenzen von Schulter, Ellbogen und Handgelenk des Roboters verwendet das System den Schnittpunkt Pw der beiden Rotationsachsen der 5. und 6. Achse (wie in Abbildung 7 dargestellt).



Armstellung 7. Position von Pw

Grenzpunkte von Armstellungen werden als “singuläre Punkte” bezeichnet.

Alle Pfade, die mit der Linearsteuerung durch Befehle (z. B. MOVE, APPROACH und DEPART) definiert werden, dürfen nicht in der Nähe eines singulären Punktes verlaufen, siehe Abschnitt 4.3 in Kapitel 4 dieses Handbuches. Wenn der Pfad in der Nähe des singulären Punktes verläuft, gibt der Roboter FEHLER6080s (Geschwindigkeit überschritten) oder FEHLER6070s (Softwarebewegungsgrenze überschritten) aus und stoppt.

(1) LINKS/RECHTS (Schulterstellung)

Die Rotationsachse der 1. Achse ist als die Grenze zwischen LINKS und RECHTS definiert.

Wenn Sie die Seite des Armgelenks aus einer senkrechten Position betrachten und der Punkt Pw auf der linken Seite der Drehachse der 1. Achse liegt, ist die Armstellung LINKS; befindet sich Pw auf der rechten Seite ist die Armstellung RECHTS. In Abbildung 8 ist die Grenze mit einer gestrichelten Linie aus abwechselnd kurzen und langen Strichen dargestellt.

Anmerkung: Wenn der Punkt Pw auf der Rotationsachse der 1. Achse liegt, d. h. auf der Grenze zwischen LINKS und RECHTS, wird er als singulärer Punkt bezeichnet.

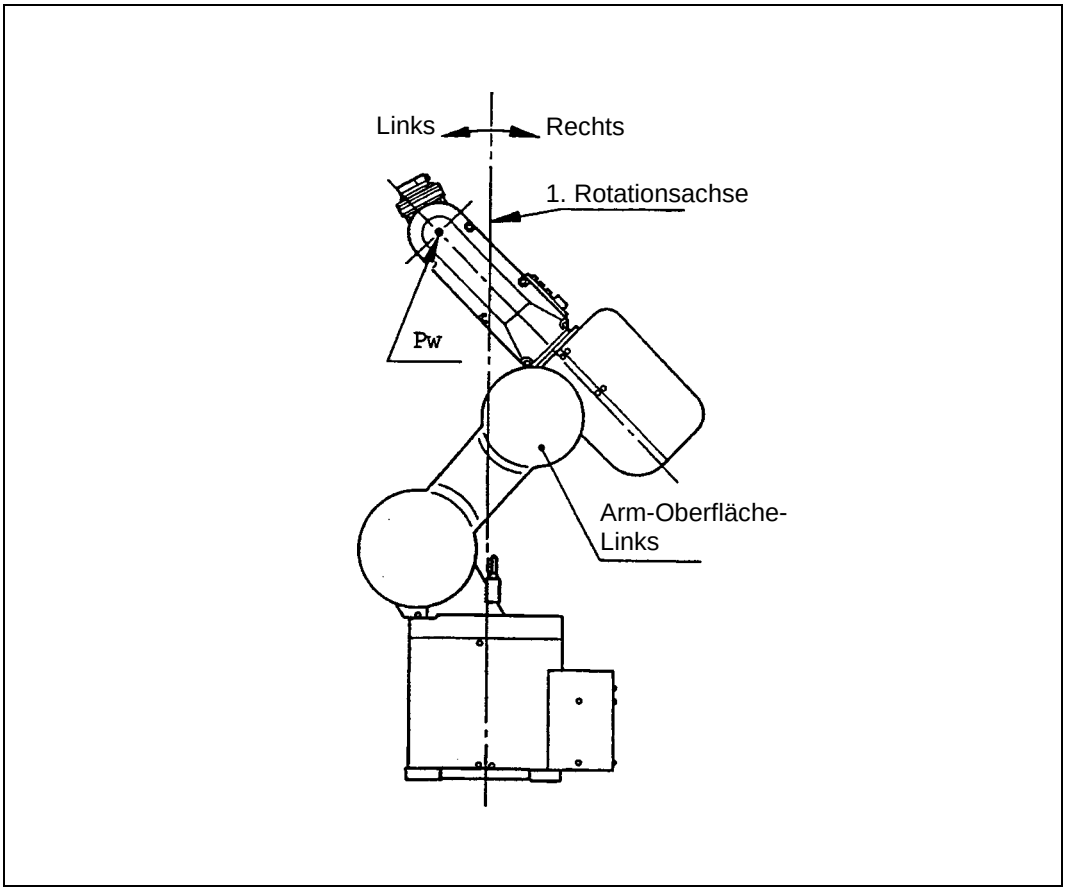


Abbildung 8. Grenze zwischen LINKS und RECHTS

(2) OBEN/UNTEN (Ellbogenstellung)

Die Mittellinie der Armverbindung (die Schulter und Ellbogen verbindet) ist als Grenze zwischen OBEN und UNTEN festgelegt.

Wenn sich Punkt Pw auf der + Seite der Mittellinie befindet, ist die Armstellung OBEN; befindet sich Pw auf der - Seite, ist die Armstellung UNTEN. In den Abbildungen 9 und 10 ist die Grenze mit einer gestrichelten Linie aus abwechselnd kurzen und langen Strichen dargestellt.

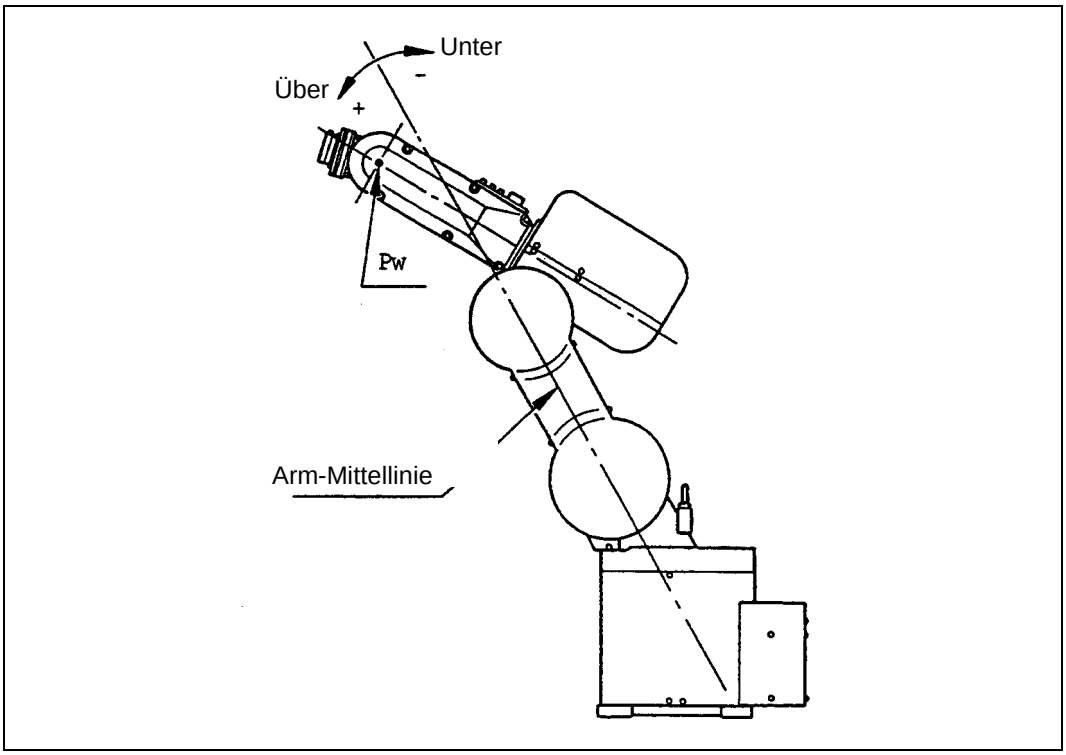


Abbildung 9. Grenze zwischen OBEN und UNTEN für LINKS

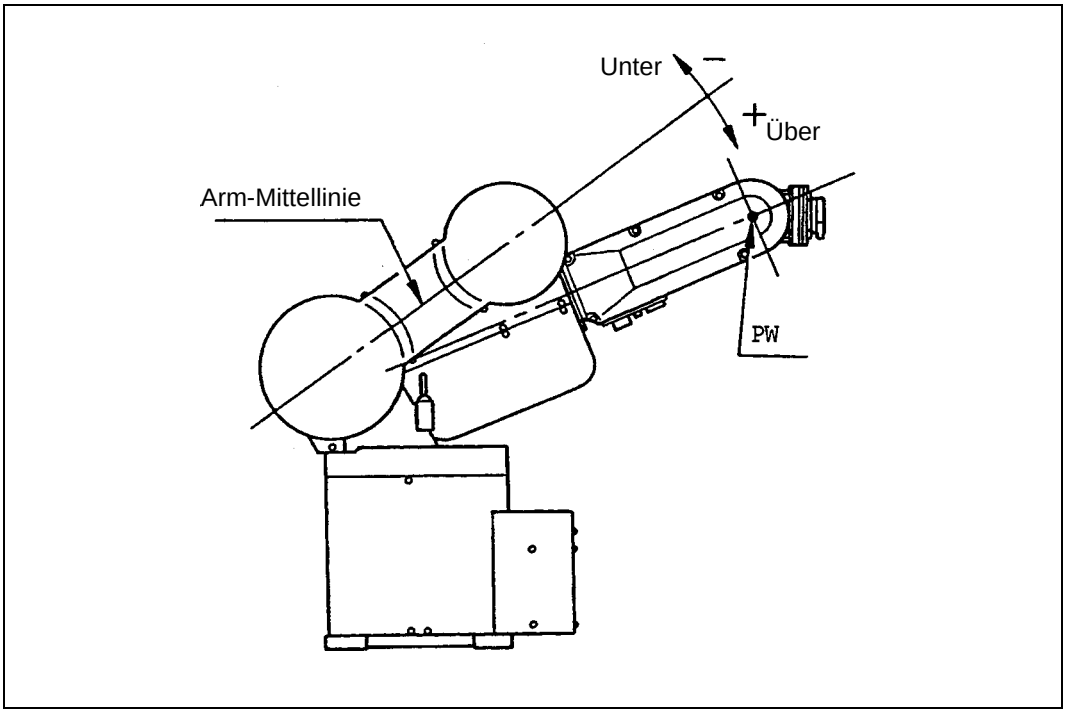


Abbildung 10. Grenze zwischen OBEN und UNTEN für RECHTS

(3) KIPPEN/NICHT KIPPEN (Handgelenkstellung)

Die Drehachse der 4. Achse ist als Grenze zwischen KIPPEN und NICHT KIPPEN definiert.

Wenn die Senkrechte zu der Flanschoberfläche die Drehachse der 4. Achse nach oben kippt, ist die Armstellung KIPPEN; kippt sie die Drehachse nach unten, ist die Armstellung NICHT KIPPEN. In den Abbildungen 11 und 12 ist die Grenze mit einer gestrichelten Linie aus abwechselnd kurzen und langen Strichen dargestellt.

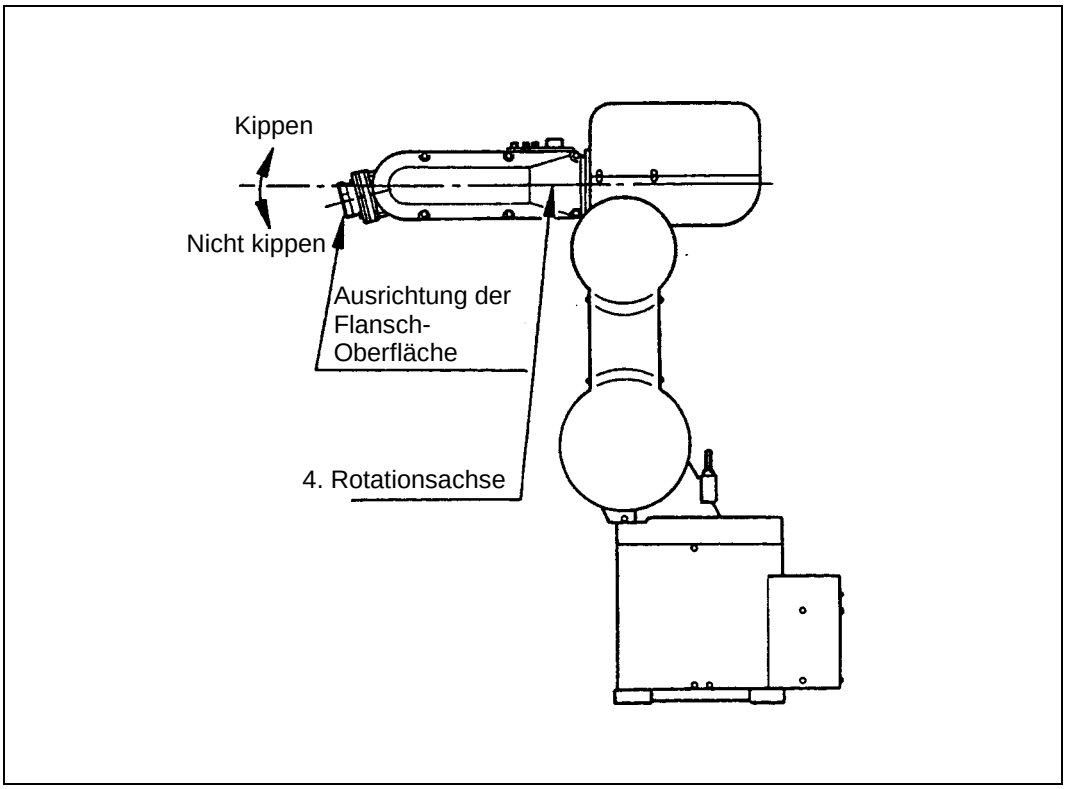


Abbildung 11. Grenze zwischen KIPPEN und NICHT KIPPEN für LINKS

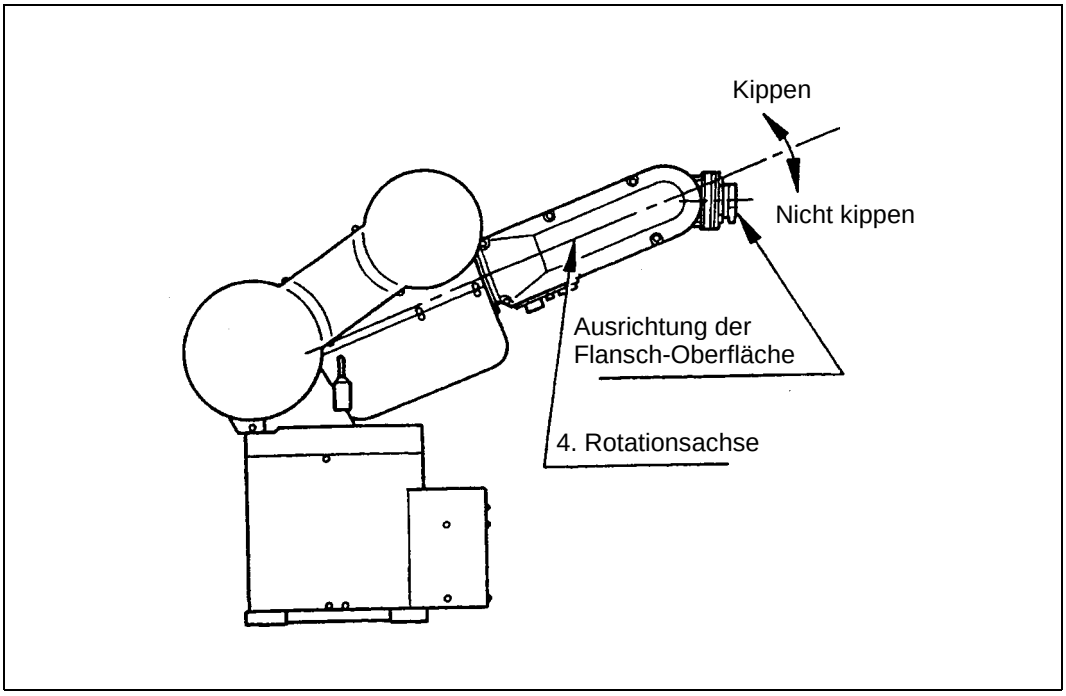


Abbildung 12. Grenze zwischen KIPPEN und NICHT KIPPEN für RECHTS

(4) EINFACH/DOPPELT (Armstellung der 6. Achse)

Wenn der Drehungswinkel ($\theta 6$) der 6. Achse innerhalb des Bereiches $-180° < \theta 6 \leq 180°$ um die Z-Achse in den mechanischen Schnittstellenkoordinaten liegt, ist die Armstellung EINFACH; liegt der Winkel im Bereich zwischen $180° < \theta 6 \leq 360°$ oder $-360° < \theta 6 \leq -180°$, ist die Armstellung DOPPELT. Grenzen existieren bei -180° und +180°.

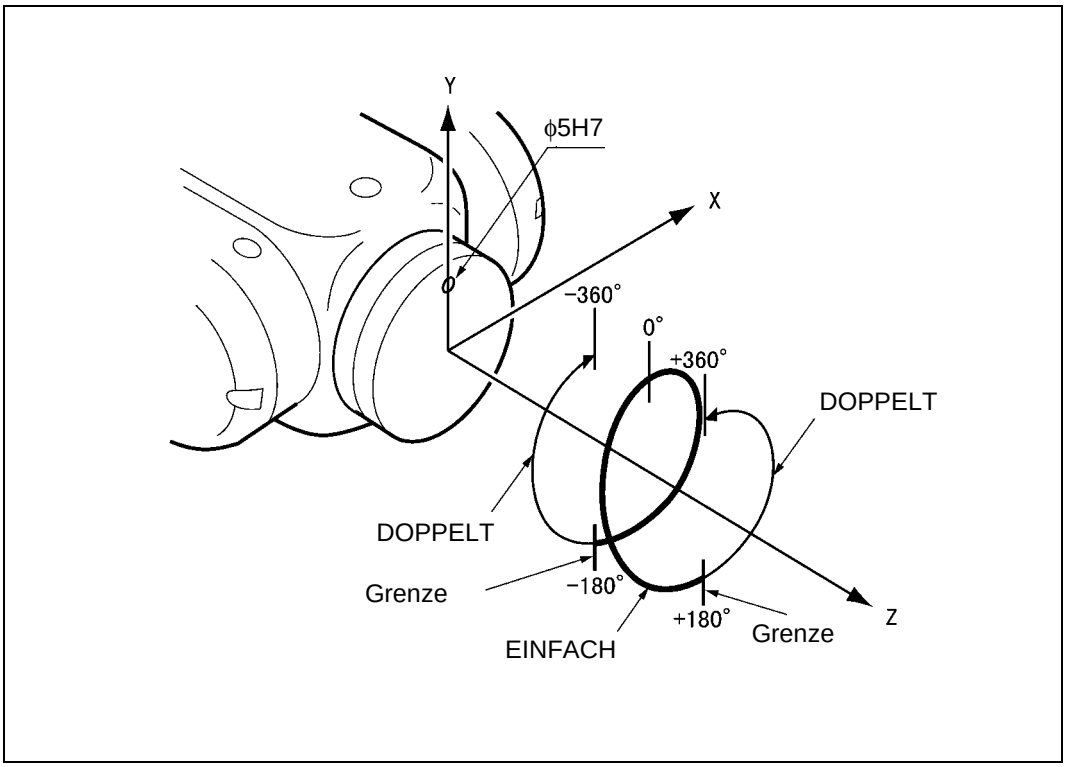


Abbildung 13. Grenze zwischen EINFACH und DOPPELT

3 Umgebungseinstellwerte

Tabellennummer	Makroname	Beschreibung	WINCAPSII
1	cnfSYS	Tabelle der Systemparameter	—
2	cnfARM	Tabelle der Bahnerstellungs-parameter	Arm Manager
3	cnfVIS	Tabelle der Bildverarbeitungs-parameter	Vision Manager
4	cnfPAC	Tabelle der PAC-Parameter	PAC Manager
5	cnfSRV	Tabelle der Servoparameter	Arm Manager
6	cnfSPD	Tabelle der Verwendung von Bedingungsparametern	Arm Manager
7	cnfITP	Tabelle der Interpreterparameter	PAC Manager
8	cnfDIO	Tabelle der DIO-Parameter	DIO Manager
9	cnfCOM	Tabelle der Kommunikations-parameter	—

Anmerkung (1): Makronamen sind in <pacman.h> definiert. Wenn Sie einen Makronamen mit den Befehlen GETENV oder LETENV verwenden, geben Sie die Datei wie folgt an:

#INCLUDE <pacman.h>

Anmerkung (2): Sie können einen Makronamen mit jedem Manager von WINCAPSII aus einer Elementnummer in einer Zufallstabelle erstellen. Schlagen Sie die entsprechende Prozedur im Benutzerhandbuch (WINCAPSII) nach.
Wenn Sie zum Beispiel eine Makrodefinitionsdatei für die Tabelle der Bahnerstellungsparameter im Arm Manager erstellen möchten, fügen Sie die Datei auf folgende Weise ein:

#INCLUDE “arm_cnf.h”

4 Verwenden von Zustandsparametern

Anzahl	Elemente	Standardwert ab Werk	Standardwert beim Einschalten	Beschreibung	Kommentare
7	Steuersatz der Bewegungsoptimierung	0	0	0: AUS 1: Nur Punkt-zu-Punkt-Bewegung 2: Nur Linearbewegung 3: Sowohl Punkt-zu-Punkt- als auch Linearbewegung (Weitere Informationen, siehe, Abschnitt 4.6, "Steuersätzen der Bewegungsoptimierung" in Kapitel 4 dieses Handbuches.)	Kann mit aspChange () eingestellt werden.
8	Bodenmontage oder Überkopfmontage	0	Letzter Wert beim Ausschalten	0: Bodenmontage 1: Überkopfmontage	
9	Masse der Nutzlast (g)	Abhängig vom Modell	Letzter Wert beim Ausschalten	Masse von Endwerkzeug und am Ende des Roboterarmes montiertem Objekt.	Kann mit aspACLD eingestellt werden.
10	Nutzlastschwerpunkt X (mm)	0	Letzter Wert beim Ausschalten	X-Komponente von Nutzlastschwerpunkt (besteht aus Endwerkzeug und Objekt) (Weitere Informationen, siehe Abschnitt 4.6 “Steuersätzen der Bewegungsoptimierung” in Kapitel 4 dieses Handbuches.)	Kann mit aspACLD eingestellt werden.
11	Nutzlastschwerpunkt Y (mm)	80	Letzter Wert beim Ausschalten	Y-Komponente von Nutzlastschwerpunkt (besteht aus Endwerkzeug und Objekt) (Weitere Informationen, siehe Abschnitt 4.6 “Steuersätzen der Bewegungsoptimierung” in Kapitel 4 dieses Handbuches)	
12	Nutzlastschwerpunkt Z (mm)	100	Letzter Wert beim Ausschalten	Z-Komponente von Nutzlastschwerpunkt (besteht aus Endwerkzeug und Objekt) (Weitere Informationen, siehe Abschnitt 4.6 “Steuersätzen der Bewegungsoptimierung” in Kapitel 4dieses Handbuches)	
13 bis 20	Codierer-Impulszähler zur Toleranz der Positionierung (J1 bis J8)	20	20	Konvergenzgenauigkeit für die angegebene Achse (eine von J1 bis J8) bei Ausführung des Bewegungsbefehls mit der Option @E	Kann mit mvSetPulseWidth () eingestellt werden.
21	Zeitüberschreitung Positionierungsabschluß (ms)	5600	5600	Wenn bei Ausführung eines Bewegungsbefehls mit der Option @E die Positionierung nicht innerhalb dieser angegebenen Zeit abgeschlossen wird, tritt eine Zeitüberschreitung auf.	Kann mit mvSetTimeOut () eingestellt werden.
22	Steuerprotokollmodus	1	Letzter Wert beim Ausschalten	Anzahl der zu speichernden Steuerprotokolle. Eingabebereich: 1 bis 3 (1250 x eingestellter Wert) = Anzahl der Steuerprotokolle	Wen viele Programme und/oder Variablen verwendet werden, kann das Einstellen vieler Steuerprotokolle beim Einschalten zu einem Fehler führen. Verringern Sie in diesem Fall die Anzahl der Steuerprotokolle.

Anzahl	Elemente	Standard wert ab Werk	Standardwert beim Einschalten	Beschreibung	Kommentare
23	Steuerprotokoll Stichprobenintervall	8	Letzter Wert beim Ausschalten	Stichprobenintervalle des Steuerprotokolls. Eingabebereich: 8, 16, 24 oder 32 ms	Wenn ein Wert angegeben wird, der kein Vielfaches von 8 ist, ändert der Controller diesen Wert automatisch auf ein Vielfaches von 8.
24	Effizienz des Schwerkrafteffektes (für Roboter mit 6 Achsen)	0	Letzter Wert beim Ausschalten	0: Schwerkraftausgleich deaktiviert 1: Schwerkraftausgleich aktiviert	Kann mit SetGravity oder ResetGravity eingestellt werden.
25	Abbruchschalter für Curlmt-Funktion	0	Letzter Wert beim Ausschalten	Wenn das niedrigste Bit 0 ist: Setzt die aktuelle Grenzwerteinstellung zurück, wenn der Motor eingeschaltet wird Wenn das zweitniedrigste Bit 0 ist: Entsperrt die Servo-Schleife, wenn der Motor eingeschaltet wird Wenn das drittniedrigste Bit 0 ist: Setzt das Abbrechen der PWM-Umschaltung zurück, wenn der Motor eingeschaltet wird	Die Freigabe der Servo-Sperre und das Abbrechen der PWM-Umschaltung wurden ausschließlich für Roboter mit 4 Achsen konzipiert.
26	Konfiguration der Servo-Sperre (für Roboter mit 4 Achsen)	0	Letzter Wert beim Ausschalten	1: Servo-Sperre freigegeben	Kann mit OffSrvLock oder OnSrvLock eingestellt werden.
27	Steuermethode (für HM/HS-D-Serie)	0	Letzter Wert beim Ausschalten	1: P-Steuerung	Stellen Sie sicher, daß “Beschleunigungsmodus ändern” auf 0 gesetzt ist.
28	Konfiguration mit großer Trägheit (für HM/HS-D-Serie)	0	Letzter Wert beim Ausschalten	1: Schleifenzunahme auf hohe Trägheit gesetzt	Stellen Sie sicher, daß “Beschleunigungsmodus ändern” auf 0 gesetzt ist und die “Masse der Nutzlast (g)” 10000 beträgt.
29	Beschleunigungsmodus ändern	0 oder 1	Letzter Wert beim Ausschalten	0: Zunahmeänderungsfunktion aktiviert 1: Zunahmeänderungsfunktion deaktiviert	Die Anfangseinstellung ist 0 oder 1 für Roboter mit 4 bzw. 6 Achsen. Ändern Sie diese Anfangseinstellung nicht.
34	Motorleistungshaltefunktion	1	Letzter Wert beim Ausschalten	Legt den Status der Motorleistung fest, wenn der Schalter “Automatisch Aktivieren” aktiviert ist. 0: Schaltet die Motorleistung AUS, wenn sie EIN war 1: Behält den aktuellen Status der Motorleistung bei	
35	Zykloidee Bewegungseinstellung	0	Letzter Wert beim Ausschalten	0: Zykloide Bewegung deaktiviert 1: Zykloide Bewegung aktiviert	Kann mit Setcycloid oder Resetcycloid eingestellt werden.

Anzahl	Elemente	Standardwert ab Werk	Standardwert beim Einschalten	Beschreibung	Kommentare
53 bis 60	Zunahmereduzierungsrate (J1 bis J8)	Wert ensprechend jedem Roboter	Letzter Wert beim Ausschalten	Zunahmereduzierungsrate für eine von J1 bis J8	Tritt in Kraft, wenn “Beschleunigungsmodus ändern”, “Steuermethode” und “Konfiguration mit hoher Trägheit” auf 0 eingestellt sind. Ändern Sie diesen Anfangswert nicht.
61 bis 68	Zunahmereduzierungsrate für Lade-operation mit hoher Trägheit (J1 bis J8) (für HM/HS-D-Serie)	0	Letzter Wert beim Ausschalten	Zunahmereduzierungsrate für eine von J1 bis J8, wenn die Ladeoperation mit hoher Trägheit ausgewählt ist	Tritt in Kraft, wenn “Beschleunigungsmodus ändern” und “Steuermethode” auf 0 und “Konfiguration mit hoher Trägheit” auf 1 eingestellt sind. Ändern Sie diesen Anfangswert nicht.
69	Neuer oder alter Robotertyp (für Roboter mit 4 Achsen)	0	Letzter Wert beim Ausschalten	0: Neuer Typ (D-Serie) 1: Alter Typ (C-Serie)	Wenn Sie den Controller nur anschaffen, um ihn an die HM/HS/HC-C-Serie anzuschließen, stellen Sie dieses Element auf 1 ein.
70	Übergangsbewegungseinstellung	0	Letzter Wert beim Ausschalten	Nach dem Neustart einer Stopoperation während einer Übergangsbewegung macht der Roboter eine Bewegung vorwärts: 0: Nach der Übergangsbewegung angegebene Zielposition (Standardwert) 1: Vor der Übergangsbewegung angegebene Zielposition	
71	Positionstoleranz nach dem Übergangsende	5	Letzter Wert beim Ausschalten	Bedingung, mit der verhindert wird, daß der Roboter beim Neustart vor der Übergangsbewegung eine Bewegung zur Zielposition hin macht. Die Bedingung sollte als Distanz von der Zielposition eingestellt werden.	Die Bedingung bezieht sich auf die Distanz von der Zielposition auf der Befehlsebene, nicht auf die tatsächliche Distanz von der aktuellen Roboter-endposition.
78	Dämpfereinstellungsrate (X) (für Roboter mit 6 Achsen)	10000	10000	Dämpfungsrate entlang der X-Achse unter Nachgiebigkeitssteuerung	Kann mit SetDampRate oder ResetDampRate eingestellt werden. Kann mit dem Programmiergerät nicht geändert werden. (Ver. 1.4 oder neuer)
79	Dämpfereinstellungsrate (Y) (für Roboter mit 6 Achsen)	10000	10000	Dämpfungsrate entlang der Y-Achse unter Nachgiebigkeitsteuerung	
80	Dämpfereinstellungsrate (Z) (für Roboter mit 6 Achsen)	10000	10000	Dämpfungsrate entlang der Z-Achse unter Nachgiebigkeitssteuerung	

Anzahl	Elemente	Standard wert ab Werk	Standardwert beim Einschalten	Beschreibung	Kommentare
81	Dämpfereinstellungsrate (RX) (für Roboter mit 6 Achsen)	10000	10000	Dämpfungsrate um die X-Achse unter Nachgiebigkeitssteuerung	Kann mit SetDampRate oder ResetDampRate eingestellt werden. Kann mit dem Programmiergerät nicht geändert werden. (Ver. 1.4 oder neuer)
82	Dämpfereinstellungsrate (RY) (für Roboter mit 6 Achsen)	10000	10000	Dämpfungsrate um die Y-Achse unter Nachgiebigkeitssteuerung	
83	Dämpfereinstellungsrate (RZ) (für Roboter mit 6 Achsen)	10000	10000	Dämpfungsrate um die Z-Achse unter Nachgiebigkeitssteuerung	
84	Nachgiebigkeits-steuermodus (für Roboter mit 6 Achsen)	1	1	Wenn das niedrigste Bit 0 ist: Nachgiebigkeitsgeschwindigkeit Steuermodus Wenn das zweitniedrigste Bit 1 ist: Deaktiviert die Schwerkraftausgleichsfunktion unter der Nachgiebigkeitssteuerung	Kann mit SetCompVMode, ResetCompVMode, SetCompControl oder SetCompFControl eingestellt werden. Kann mit dem Programmiergerät nicht geändert werden. (Ver. 1.4 oder neuer)
86	Antivibrationseinstellung (für Roboter mit 6 Achsen)	0	Letzter Wert beim Ausschalten	1: Steuermodus für die Reduzierung der Restvibration	Kann mit SetVibControl oder ResetVibControl eingestellt werden.
87	Nachgiebigkeits-steuerung EIN/AUS (für Roboter mit 6 Achsen)	0	0	1: Unter Nachgiebigkeitssteuerung	Kann mit SetCompControl, SetCompF Steuerung oder ResetCompControl eingestellt werden Kann mit dem Programmiergerät nicht geändert werden. (Ver. 1.4 oder neuer)
88	Koordiniert die Nachgiebigkeitssteuerung (für Roboter mit 6 Achsen)	0	0	0: Basiskoordinaten 1: Werkzeugkoordinaten 2: Arbeitskoordinaten	Kann mit SetFrcCoord eingestellt werden. Kann mit dem Programmiergerät nicht geändert werden. (Ver. 1.4 oder neuer)
89	Erzwungene Begrenzungsrate (+X) (für Roboter mit 6 Achsen)	10000	10000	Erzwungene Steuerungsrate entlang der +X-Achse unter Nachgiebigkeitssteuerung	Kann mit SetFrcCoord eingestellt werden. Kann mit dem Programmiergerät nicht geändert werden. (Ver. 1.4 oder neuer)
90	Erzwungene Begrenzungsrate (+Y) (für Roboter mit 6 Achsen)	10000	10000	Erzwungene Steuerungsrate entlang der +Y-Achse unter Nachgiebigkeitssteuerung	
91	Erzwungene Begrenzungsrate (+Z) (für Roboter mit 6 Achsen)	10000	10000	Erzwungene Steuerungsrate entlang der +Z-Achse unter Nachgiebigkeitssteuerung	

Anzahl	Elemente	Standard wert ab Werk	Standardwert beim Einschalten	Beschreibung	Kommentare
92	Erzwungene Begrenzungsrate (+RX) (für Roboter mit 6 Achsen)	10000	10000	Erzwungene Steuerungsrate um die +X-Achse unter Nachgiebigkeitssteuerung	Kann mit SetFrcCoord eingestellt werden. Kann mit dem Programmiergerät nicht geändert werden. (Ver. 1.4 oder neuer)
93	Erzwungene Begrenzungsrate (+RY) (für Roboter mit 6 Achsen)	10000	10000	Erzwungene Steuerungsrate um die +Y-Achse unter Nachgiebigkeitssteuerung	
94	Erzwungene Begrenzungsrate (+RZ) (für Roboter mit 6 Achsen)	10000	10000	Erzwungene Steuerungsrate um die +Z-Achse unter Nachgiebigkeitssteuerung	
95	Erzwungene Begrenzungsrate (-X) (für Roboter mit 6 Achsen)	10000	10000	Erzwungene Steuerungsrate entlang der -X-Achse unter Nachgiebigkeitssteuerung	Kann mit SetFrcCoord eingestellt werden. Kann mit dem Programmiergerät nicht geändert werden. (Ver. 1.4 oder neuer)
96	Erzwungene Begrenzungsrate (-Y) (für Roboter mit 6 Achsen)	10000	10000	Erzwungene Steuerungsrate entlang der -Y-Achse unter Nachgiebigkeitssteuerung	
97	Erzwungene Begrenzungsrate (-Z) (für Roboter mit 6 Achsen)	10000	10000	Erzwungene Steuerungsrate entlang der -Z-Achse unter Nachgiebigkeitssteuerung	
98	Erzwungene Begrenzungsrate (-RX) (für Roboter mit 6 Achsen)	10000	10000	Erzwungene Steuerungsrate um die -X-Achse unter Nachgiebigkeitssteuerung	Kann mit SetFrcCoord eingestellt werden. Kann mit dem Programmiergerät nicht geändert werden. (Ver. 1.4 oder neuer)
99	Erzwungene Begrenzungsrate (-RY) (für Roboter mit 6 Achsen)	10000	10000	Erzwungene Steuerungsrate um die -Y-Achse unter Nachgiebigkeitssteuerung	
100	Erzwungene Begrenzungsrate (-RZ) (für Roboter mit 6 Achsen)	10000	10000	Erzwungene Steuerungsrate um die -Z-Achse unter Nachgiebigkeitssteuerung	
101	Nachgiebigkeitseinstellungsrate (X) (für Roboter mit 6 Achsen)	10000	10000	Nachgiebigkeitsrate entlang der X-Achse unter Nachgiebigkeitssteuerung	Kann mit SetCompRate eingestellt werden. Kann mit dem Programmiergerät nicht geändert werden. (Ver. 1.4 oder neuer)
102	Nachgiebigkeitseinstellungsrate (Y) (für Roboter mit 6 Achsen)	10000	10000	Nachgiebigkeitsrate entlang der Y-Achse unter Nachgiebigkeitssteuerung	
103	Nachgiebigkeitseinstellungsrate (Z) (für Roboter mit 6 Achsen)	10000	10000	Nachgiebigkeitsrate entlang der Z-Achse unter Nachgiebigkeitssteuerung	
104	Nachgiebigkeitseinstellungsrate (RX) (für Roboter mit 6 Achsen)	10000	10000	Nachgiebigkeitsrate um die X-Achse unter Nachgiebigkeitssteuerung	Kann mit SetCompRate eingestellt werden. Kann mit dem Programmiergerät nicht geändert werden. (Ver. 1.4 oder neuer)
105	Nachgiebigkeitseinstellungsrate (RY) (für Roboter mit 6 Achsen)	10000	10000	Nachgiebigkeitsrate um die Y-Achse unter Nachgiebigkeitssteuerung	
106	Nachgiebigkeitseinstellungsrate (RZ) (für Roboter mit 6 Achsen)	10000	10000	Nachgiebigkeitsrate um die Z-Achse unter Nachgiebigkeitssteuerung	
107	Nachgiebigkeits-/ Positions-fehlertoleranz (X) (für Roboter mit 6 Achsen)	100	100	Zulässige Abweichung entlang der X-Achse unter Nachgiebigkeitssteuerung	Kann mit SetCompEralw eingestellt werden. Kann mit dem Programmiergerät nicht geändert werden. (Ver. 1.4 oder neuer)
108	Nachgiebigkeits-/ Positions- fehlertoleranz (Y) (für Roboter mit 6 Achsen)	100	100	Zulässige Abweichung entlang der Y-Achse unter Nachgiebigkeitssteuerung	
109	Nachgiebigkeits-/ Positions fehlertoleranz (Z) (für Roboter mit 6 Achsen)	100	100	Zulässige Abweichung entlang der Z-Achse unter Nachgiebigkeitssteuerung	

Anzahl	Elemente	Standard wert ab Werk	Standardwert beim Einschalten	Beschreibung	Kommentare
110	Nachgiebigkeits-/ Positions fehlertoleranz (RX) (für Roboter mit 6 Achsen)	300	300	Zulässige Abweichung um die X-Achse unter Nachgiebigkeitssteuerung	Kann mit SetCompEralw eingestellt werden. Kann mit dem Programmiergerät nicht geändert werden. (Ver. 1.4 oder neuer)
111	Nachgiebigkeits-/ Positions fehlertoleranz (RY) (für Roboter mit 6 Achsen)	300	300	Zulässige Abweichung um die Y-Achse unter Nachgiebigkeitssteuerung	
112	Nachgiebigkeits-/ Positionsfehlertoleranz (RZ) (für Roboter mit 6 Achsen)	300	300	Zulässige Abweichung um die Z-Achse unter Nachgiebigkeitssteuerung	
113	Erzwungener Offset (X) (für Roboter mit 6 Achsen)	0	0	Erzwungener Offset entlang der X-Achse unter Nachgiebigkeitssteuerung	Kann mit SetFrcAssist eingestellt werden. Kann mit dem Programmiergerät nicht geändert werden. (Ver. 1.4 oder neuer)
114	Erzwungener Offset (Y) (für Roboter mit 6 Achsen)	0	0	Erzwungener Offset entlang der Y-Achse unter Nachgiebigkeitssteuerung	
115	Erzwungener Offset (Z) (für Roboter mit 6 Achsen)	0	0	Erzwungener Offset entlang der Z-Achse unter Nachgiebigkeitssteuerung	
116	Erzwungener Offset (RX) (für Roboter mit 6 Achsen)	0	0	Offset-Moment um die X-Achse unter Nachgiebigkeitssteuerung	Kann mit SetFrcAssist eingestellt werden. Kann mit dem Programmiergerät nicht geändert werden. (Ver. 1.4 oder neuer)
117	Erzwungener Offset (RY) (für Roboter mit 6 Achsen)	0	0	Offset-Moment um die Y-Achse unter Nachgiebigkeitssteuerung	
118	Erzwungener Offset (RZ) (für Roboter mit 6 Achsen)	0	0	Offset-Moment um die Z-Achse unter Nachgiebigkeitssteuerung	
120	Optimierung initialisieren	0	Letzter Wert beim Ausschalten	0: Setzt das Steuerungsset der Bewegungsoptimierung auf 0, wenn der Strom aus- und eingeschaltet wird (Standardwert) 1: Setzt das Steuerungsset der Bewegungsoptimierung nicht zurück, wenn der Strom aus- und eingeschaltet wird	(Ver. 1.4 oder neuer)
121 bis 128	Drehmomentbegrenzung für Nachgiebigkeitssteuerung (J1 bis J8) (für Roboter mit 6 Achsen)	0	0	Aktueller Grenzwert für eine von J1 bis J8 unter Nachgiebigkeitssteuerung	Kann mit SetCompJLimit oder ResetCompJLimit eingestellt werden. Kann mit dem Programmiergerät nicht geändert werden. (Ver. 1.4 oder neuer)

5 Liste der reservierten Wörter

Diese reservierten Wörter dürfen nicht als Variablennamen oder Kennungsnamen verwendet werden.

A	ABOVE ABS ACCEL ACOS ALL AND APPROACH AREA AREAPOS AREASIZE AS ASC ASIN ATN ATN2 AVEC
B	B BELOW BIN BIT BLOB BLOBCOPY BLOBLABEL BLOBMEASURE BREAK BUZZER BYTE
C	C CALL CAMASPECT CAMIN CAMLEVEL CAMMODE CAMTIN CAMTOFF CAMTON CASE CHANGE CHANGETOOL CHANGEWORK CHR CLEARLOG CODE COM COMMON CONT COS CREATESEM CURACC CURDEC CUREJNT CURFIG CURJACC CURJDEC CURJNT CURJSPD CURPOS CURSPD CURTRN CYCLE
D	D DATE DECEL DEF DEFDBL DEFEND DEFINT DEFIO DEFJNT DEFPOS DEFSNG DEFSTR DEFTRN DEFVEC DEG DEGRAD DELAY DELETESEM DEPART DESTJNT DESTPOS DESTTRN DIM DIST DO DOUBLE DRAW DRIVE DRIVEA
E	E EJ ELSE ELSEIF END ENDIF EQJ EQP EQS EQT EQU EQV ERL ERR ERRMSG ERROR EXIT EXP
F	F FALSE FIG FIGARPL FIGAPRP FLIP FLOAT FLUSH FLUSHSEM FOR FORMAT FREE FUNCTION
G	GETENV GIVEARM GIVESEM GIVEVIS GO GOHOME GOSUB GOTO GRASP
H	H HALT HAND HEX HOLD HOME
I	I IF IN INPUT INPUTB$ INT INTEGER INTERRUPT IO IOBLOCK IS
J	J J2P J2T JOINT JSPEED JACCEL JDECEL
K	KILL
L	L LEFT LEFTY LEN LET LETA LETENV LETF LETJ LETJ1 LETJ2 LETJ3 LETJ4 LETJ5 LETJ6 LETO LETP LETR LETRX LETRY LETRZ LETX LETY LETZ LINEINPUT LOG LOG10 LOOP
M	MAGNITUDE MAX MID MIN MIRROR MOD MOVE MPS
N	NEJ NEP NEQ NES NET NEV NEXT NONFLIP NOT
O	OFF ON OR ORD OUT OVEC
P	P P2J P2T PI POSE POSITION POSRX POSRY POSRZ POSX POSY POSZ POW PRINT PRINTDBG PRINTLBL PRINTMSG PRIORITY PROGRAM PTP PVEC
R	R RAD RADDEG RELEASE REM REPEAT RESET RESETAREA RESETVALVE RESUME RETURN RIGHT RIGHTY RND ROBOT ROTATE ROTATEH RUN RVEC
S	S SEC SELECT SENDKEY SET SETAREA SETVALVE SGN SHADDGROUP SHCIRCLE SHCLRGROUP SHCLRMODEL SHCOPYMODEL SHCORNER SHDEFCIRCLE SHDEFCORNER SHDEFMODEL SHDISPMODEL SHDROPGROUP SHGROUP SHLOADMODEL SHMODEL SHREFGROUP SHREFMODEL SHSAVEMODEL SIN SINGLE SPEED SQR SQRT START STARTLOG STATUS STEP STOP STOPEND STOPLOG STR STRING STRPOS SUB SUSPEND

T	T T2J T2P TAKEARM TAKESEM TAKEVIS TAN THEN TIME TIMER TINV TO TOOL TOOLPOS TRNS TRUE
U	UNTIL
V	V VAL VECTOR VER VISBINA VISBINAR VISBRIGHT VISCAL VISCAMOUT VISCIRCLE VISCLS VISCOPY VISCROSS VISDEFCHAR VISDEFTABLE VISEDGE VISELLIPSE VISFILTER VISGETNUM VISGETP VISGETSTR VISHIST VISLEVEL VISLINE VISLOC VISMASK VISMEASURE VISOVERLAY VISPLNOUT VISPOSX VISPOSY VISPRINT VISPROJ VISPTP VISPUTP VISREADBAR VISREADQR VISRECT VISREFCAL VISREFHIST VISREFTABLE VISSCREEN VISSECT VISSTATUS VISWORKPLN
W	WAIT WEND WHILE WINDCLR WINDCOPY WINDDISP WINDMAKE WINDREF WORD WORK WORKPOS WRITE
X	X XOR XY XYH
Y	YZ YZH
Z	ZX ZXH

6 Befehlsentsprechungstabelle zu herkömmlichen Sprachbefehlen (VS)

■ Entsprechungstabelle für Bewegungsbefehle

Konventionelle Sprachbefehle	PAC-Befehl	MOVE-Makro
MVE J*n*	MOVE P, J*n*	——
MVP J*n*	MOVE P, @P J*n*	——
MVSE P*n*	MOVE L, T*n*	——
MVSP P*n*	MOVE L, @P T*n*	——
DRVE *(j 1 j 2 j 3 j 4 j 5 j 6*)	DRIVE (1*j1*),…	*Jn = Jc* + (*j 1 j 2 j 3 j 4 j 5 j 6*) MOVE P, *Jn*
DRVE J*n*	DRIVE (1, JOINT (1, J*n*)),…	*jj = Jc* +(JOINT (1, J*n*)),…, JOINT (6, J*n*) MOVE P, @P J*n*
DRVP (*j 1 j 2 j 3 j 4 j 5 j 6*)	DRIVE @P (1 *j* 1),…	*Jn =Jc* + (*j 1 j 2 j 3 j 4 j 5 j 6*) MOVE P, @P J*n*
DRVP J*n*	DRIVE @P (1, JOINT (1*j1*)),…	*jj=Jc* + (JOINT (1, J*n*)),…, JOINT (6, J*n*) MOVE P, @P *jj*
DRWE (*x, y, z*)	DRAW L, (*x, y, z*)	MOVE L, *Pc* + (*x, y, z*)
DRWE P*n*	DRAW L, PVEC (T*n*)	MOVE L, *Pc* + (POSX (P*n*), POSY (P*n*), POSZ (Pn))
DRWP (*x, y, z*)	DRAW L, @P (*x, y, z*)	MOVE L, @P *Pc* + (*x, y, z*)
DRWP P*n*	DRAW L, @P PVEC (T*n*)	MOVE L, @P *Pc* + (POSX (P*n*), POSY (P*n*), POSZ (P*n*))
DEPE *n*	DEPART L, *n*	MOVE L, *Pc* + (0, 0, -*n*) H
DEPP *n*	DEPART L, @P *n*	MOVE L, @P *Pc* + (0, 0, -*n*) H
APRE *n*	APPROACH L, *Px*, *n*	MOVE L, *Px* + (0, 0, -*n*) H
APRP *n*	APPROACH L, @P *Px*, *n*	MOVE L, @P *Px* + (0, 0, -*n*) H
APRJE *n*	APPROACH P, *Px*, *n*	MOVE P, *Px* + (0, 0, -*n*) H
APRJP *n*	APPROACH P, @P *Px*, *n*	MOVE P, @P *Px* + (0, 0, -*n*) H
ROTE *n*	ROTATEH *n*	MOVE L, *Pc* + (*0, 0, 0, 0, 0, n*) H
ROTP *n*	ROTATEH @*P n*	MOVE L, @P *Pc* + (*0, 0, 0, 0, 0, n*) H
MVRE P*n2*, P*n3*	MOVE C, T*n2*, T*n3*	——
MVPR P*n2*, P*n3*	MOVE C, T*n2*, @P T*n3*	——

(!) *Pc/Jc* ist eine aktuelle Position und *Px/Jx* ist eine Zielposition.
(!) DRVE einem Argument in der DRIVE-Bibliotheksspezifikation hinzuzufügen wird untersucht.
(!) *j j* ist eine temporäre lokale Variable.

■ Entsprechungstabelle für Geschwindigkeitszielbefehle

Konventionelle Sprachbefehle	PAC-Befehl	Anmerkung
ISP *n*	SPEED *n*	
ACC n	ACCEL *n, n*	
AACC *n*	ACCEL *n*	
RACC *n*	DECEL *n*	

■ Entsprechungstabelle für Sprungbefehle

Konventionelle Sprachbefehle	PAC-Befehl	Anmerkung
JI *m-n*	IF IO [m]=1 THEN *label*n	
JZ *m-n*	IF IO [m]=0 THEN *label*n	
JMP *n*	GOTO *label*n	
CMP *l s m*	IF *l s m* THEN *label*n	s ist ein Vergleichssymbol.

Konventionelle Sprachbefehle	PAC-Befehl	Anmerkung
LABL *n*	*label*n.*	
LABL *n*	*label*n.*	
IPCLR *n*	CALL pltResetAll (*n*)	
INTRPT	INTERRUPT ON	Wechselt nach dem nächsten Befehl auf INTERRUPT OFF.
REM	REM|'	
ON *n* ON *n-m*	SET IO [*n*] SET IO [*n* TO *m*]	
OFF *n* OFF *n-m*	RESET IO [*n*] RESET IO [*n* TO *m*]	
ONT *n-m* TIME=t	SET IO [*n* TO *m*], t*10	Die Einheit ist ms. (Anmerkung) Bei PAC geht eine Steuerung nicht zum nächsten Prozeß weiter.
VON *n* VON *n-m*	SET IO [*n*] SET IO [*n* TO *m*]	
VOFF *n* VOFF *n-m*	RESET IO [*n*] RESET IO [*n* TO *m*]	
ON PLT1END	SET IO [*n*]	Siehe DIO-Zuordnungstabelle.
OFF PLT1END	RESET IO [*n*]	Siehe DIO-Zuordnungstabelle.
ON PLTEND	SET IO [*n*]	Siehe DIO-Zuordnungstabelle.
OFF PLTEND	RESET IO [*n*]	Siehe DIO-Zuordnungstabelle.
INB *L m-n*	CALL ndInb (*L, m, n*)	Bibliotheksaufruf
ONB *L m-n*	CALL ndOnb (*L, m, n*)	Bibliotheksaufruf
ONB l*x* m-n	CALL ndOnbl (*x, m, n*)	Bibliotheksaufruf

■ Entsprechungstabelle für Stopbefehle

Konventionelle Sprachbefehle	PAC-Befehl	Anmerkung
END	END	
STOP	HOLD	
STOPEND	STOPEND	
TIM *n*	DELAY *n** 10	

■ Entsprechungstabelle SETI-Befehle

Konventionelle Sprachbefehle	PAC-Befehl	Anmerkung
Variable 6 Achsen P*n*	T [*n*]	
Indirekte Referenz I*n*. P	P [*n*]	
$	CURPOS|*	
\	CURJNT|*	
ISP	CURSPD	
AACC	CURACC	
RACC	CURDEC	
SETI lx = N_*n*	CALL pltGetN (*n, x*)	Bibliotheksaufruf
SETI l*x* =M_*n*	CALL pltGetM (*n, x*)	Bibliotheksaufruf
SETI l*x* =K_*n*	CALL pltGetK (*n, x*)	Bibliotheksaufruf
SETI l*x* =N1_*n*	CALL pltGetN1 (*n, x*)	Bibliotheksaufruf
SETI l*x*=M1_*n*	CALL pltGetM1 (*n, x*)	Bibliotheksaufruf
SETI l*x*=K1_*n*	CALL pltGetK1 (*n, x*)	Bibliotheksaufruf

■ Entsprechungstabelle für Operationsbefehle

Konventionelle Sprachbefehle	PAC-Befehl	Anmerkung
+	+	Addition
-	-	Subtraktion
*	*	Multiplikation
/	/	Division
%	MOD	Rest
.	*	Innenprodukt (für Vektoren)
X	X	Außenprodukt (für Vektoren)
ABS (*n*)	ABS (*n*)	
SIN (*n*)	SIN (*n*)	
COS (*n*)	COS (*n*)	
TAN (*n*)	TAN (*n*)	
ATAN (*n*)	ATAN (*n*)	
ATN2 (*y, x*)	ATN2 (*y, x*)	
SQRT (*n*)	SQRT (*n*)	
FWRD (J*n*)	J2P (J [*n*])	
REV2 (P*n*)	P2J (P [*n*])	
TRNS (P*m*, J*k*)	Pm + (*x, y, z,* α*,* β*,* γ) H	Siehe nächste Seite.
TINV (P*n*)	TINV (T*n*)	
DATE	CALL ndDate	Bibliotheksspezifikation ist in Untersuchung.
TIME (0)	CALL ndTime	Bibliotheksspezifikation ist in Untersuchung.
TIME (1)	TIMER	

■ Entsprechungstabelle für Kommunikationsbefehle

Konventionelle Sprachbefehle	PAC-Befehl	Anmerkung
VIS *n*	CALL ndVis (*n*)	Bibliotheksaufruf
JF *n-m*	CALL ndJf (*n, k*) IF I [*k*]=0 THEN *label	Bibliotheksaufruf
VSET *n*	CALL ndVset ()	Bibliotheksaufruf
VDT	CALL ndVdt	Bibliotheksaufruf
VPUT $ VPUT P*n*	CALL ndVput ()	Bibliotheksaufruf
VRST	CALL ndVrst	Bibliotheksaufruf

■ Entsprechungstabelle definierte Befehle

Konventionelle Sprachbefehle	PAC-Befehl	Anmerkung
SUB *n*	CALL SUB*n*	SUB*n* ist ein Programmname.
PALT *n*	CALL pltMove (*n*)	Bibliotheksaufruf
TOOL *n*	CHANGETOOL *n*	

Funktionen zum Verschieben des Koordinatensystems

Obwohl die PAC-Sprache die Koordinatensystem-Verschiebungsfunktionen (Befehl TRANS) herkömmlicher Sprachbefehle nicht unterstützt, können Sie das Koordinatensystem mit der Positionsoperation verschieben. Weitere Information, siehe Abschnitt 1.9.7 “Positionsoperation” in Kapitel 1 dieses Handbuches.
Für das Koordinatensystem stehen Methoden zur Benutzung von Variablen der Typen P und T zur Verfügung, mit denen die Werkstückkoordinatenbezüge und die Werkzeugkoordinatensystembezüge verändert werden können.
Verschieben Sie das Koordinatensystem zum Variablentyp J, wird er mit JP2 oder J2T in eine Variable vom Typ P oder T umgewandelt.

(1) Wenn Variablen vom Typ P verwendet werden

1. Bezeichnen Sie die Relativbewegungsdistanz mit dem Arbeitskoordinatensystem als Bezug.
   Pn = Pm + (X, Y, Z, Rx, Ry, Rz)
   Ermitteln Sie die Position Pn: Von der Bezugsstellung Pm bewegt sich der Standort in Millimetern (mm) um X, Y und Z in die Richtungen der X-, Y- und Z-Achsen der entsprechenden Arbeitskoordinatensysteme. Die Haltung dreht sich in Grad um Rx, Ry und Rz um die X-, Y- und Z-Achsen der entsprechenden Werkstückkoordinaten. Der Roboter nimmt die Armstellung Pm an.
2. Bezeichnen Sie die Relativbewegungsdistanz mit dem Werkzeugkoordinatensystem als Bezug.
   Pn=Pm + (X, Y, Z, Rx, Ry, Rz) H
   Ermitteln Sie die Position Pn: Von der Bezugsstellung Pm aus bewegt sich der Standort in Millimetern (mm) um X, Y und Z in Richtung der X- (Normalenvektor), Y- (Orientierungsvektor) und Z-Achse (Anfahrvektor) des entsprechenden Arbeitskoordinatensystems. Die Haltung dreht sich in Grad von Rx, Ry und Rz um die X- (Normalenvektor), Y- (Orientierungsvektor) und Z-Achse (Anfahrvektor) der entsprechenden Werkstückkoordinaten. Der Roboter nimmt die Armstellung Pm an.

(2) Wenn Variablen vom Typ T verwendet werden

1. Bezeichnen Sie die Relativbewegungsdistanz mit dem Arbeitskoordinatensystem als Bezug.
   DEFTRN LT1, LT2
   DEFPOS LP1
   LP1= (0, 0, 0, Rx, Ry, Rz)
   LT1=P2T(LP1)
   LT2=Tm
   LETP LT2 = (0, 0, 0)
   Tn=LT1*LT2
   LETP Tn = PVEC(Tm) + (X, Y, Z)
   LETF Tn = FIG (Tm)
   Ermitteln Sie eine Position Tn: Von der Bezugsstellung Pm bewegt sich der Standort in Millimetern (mm) um X, Y und Z in die Richtungen der X-, Y- und Z-Achsen der entsprechenden Arbeitskoordinatensysteme. Die Haltung dreht sich in Grad um Rx, Ry und Rz um die X-, Y- und Z-Achsen der entsprechenden Werkstückkoordinaten. Der Roboter nimmt die Armstellung Tm an.
2. Bezeichnen Sie die Relativbewegungsdistanz mit dem Werkzeugkoordinatensystem als Bezug.
   DEFTRN LT1
   DEFPOS LP1
   LP1=(X, Y, Z, Rx, Ry, Rz)
   LT1=P2T(LP1)
   Tn=LT1*LT2
   LETF Tn = FIG (Tm)
   Ermitteln Sie die Position Pn: Von der Bezugsstellung Pm aus bewegt sich der Standort in Millimetern (mm) um X, Y und Z in Richtung der X- (Normalenvektor), Y- (Orientierungsvektor) und Z-Achse (Anfahrvektor) des entsprechenden Arbeitskoordinatensystems. Die Haltung dreht sich in Grad von Rx, Ry und Rz um die X- (Normalenvektor), Y- (Orientierungsvektor) und Z-Achse (Anfahrvektor) der entsprechenden Werkstückkoordinaten. Der Roboter nimmt die Armstellung Tm an.

(Anmerkung) Der Befehl TRANS ist ein Koordinatensystem-Verschiebungsbefehl, der das Werkzeugkoordinatensystem als Bezug verwendet. Die Haltungsrotationsmethode unterscheidet sich jedoch vom Werkzeugkoordinatensystem-Verschiebungsbefehl für Variablen der Typen P und T wie folgt.

	Haltungsrotationsmethode des Koordinatensystems
Js = (X, Y, Z, Rx, Ry, Rz) Pn = TRANS (Pm, Js)	Aus der Haltung an der Bezugsstellung Pm heraus dreht es um Rx (Grad) um die X-Achse (Normalenvektor) des Werkzeugkoordinatensystems und um Ry (Grad) um die Y-Achse der gedrehten Werkzeugachse. Es dreht sich auch um Rz (Grad) um die Z-Achse des gedrehten Werkzeugkoordinatensystems.
Pn=Pm + (X, Y, Z, Rx, Ry, Rz) H	Aus der Haltung an der Bezugsstellung Pm heraus dreht es um Rx (Grad) um die X-Achse (Normalenvektor) des Werkzeugkoordinatensystems und um Ry (Grad) um die Y-Achse der Werkzeugachse vor der Drehung. Es dreht sich auch um Rz (Grad) um die Z-Achse des Werkzeugkoordinatensystems vor der Drehung.

Für eine Haltungsrotation ähnlich dem Befehl TRANS der herkömmlichen Sprache verwenden Sie folgende Benennungen.

(1) Wenn Variablen vom Typ P verwendet werden
```
Pn = Pm + (X, Y, Z, 0, 0, 0) H
Pn = Pn + (0, 0, 0, Rx, 0, 0) H
Pn = Pn + (0, 0, 0, 0, Ry, 0) H
Pn = Pn + (0, 0, 0, 0, 0, Rz) H
```
(2) Wenn Variablen vom Typ T verwendet werden
```
DEFTRN LT
DEFPOS LP
LP = (X, Y, Z, 0, 0, 0)
LT = P2T (LP)
Tn = Tm*LT
LP = (0, 0, 0, Rx, 0, 0)
LT = P2T (LP)
Tn = Tn*LT
LP = (0, 0, 0, 0, Ry, 0)
LT = P2T (LP)
Tn = Tn*LT
LP = (0, 0, 0, 0, 0, Rz)
LT = P2T (LP)
Tn = Tn*LT
LETF Tn = FIG (Tm)
```

7 Kompatibilitätstabelle

Die mit der Funktion VER$ verwendeten Ausdruckswerte und die entsprechenden Module sind in der nachstehenden Tabelle aufgeführt.
Weitere Informationen zur Funktion VER$, siehe Abschnitt “12.1.3 VER$ (Funktion)” in Kapitel 12 dieses Handbuches.

Ausdruckswert	Modulname
0	ROM
1	Hauptplatine
2	DIO-Karte
3	Bildverarbeitungskarte
4	DeviceNet-Karte
5	Programmiergerät
6	Bedienfeld
7	PAC-Sprachen-Prozeßmodul
8	Zwischencode-Prozeßmodul
9	Pfad-Prozeßmodul
10	Servo-Prozeßmodul
11	Kommunikations-Prozeßmodul
12	Programmiergerät-Prozeßmodul
13	Bildverarbeitungs-Prozeßmodul
14	Bildverarbeitungs-Kommunikationsprozeßmodul
15	DeviceNet-Prozeßmodul
16	PAC-Sprachspezifikation
17	Zwischencodespezifikation
18	Servo-Kommunikationsspezifikation
19	Bildverarbeitungs-Kommunikationsspezifikation
20	ROBOTalk-Kommunikationsspezifikation
21	Programmiergeräte-Kommunikationsspezifikation
22	DeviceNet-Kommunikationsspezifikation

8 Parametereinstelltabelle

■ Pac Manager - Programm

Parametername	Makroname	Beschreibung
Anzahl Variablen vom Typ I	PC_NO_INT	Die Anzahl der numerierten langen ganzzahligen Variablen.
Anzahl Variablen vom Typ F	PC_NO_SNG	Die Anzahl der numerierten Variablen vom Typ Real mit einfacher Genauigkeit.
Anzahl Variablen vom Typ D	PC_NO_DBL	Die Anzahl der numerierten Variablen vom Typ Real mit doppelter Genauigkeit.
Anzahl Variablen vom Typ V	PC_NO_VEC	Die Anzahl der numerierten Vektorvariablen.
Anzahl Variablen vom Typ P	PC_NO_POS	Die Anzahl der numerierten Positionsvariablen.
Anzahl Variablen vom Typ J	PC_NO_JNT	Die Anzahl der numerierten Gelenkvariablen.
Anzahl Variablen vom Typ T	PC_NO_TRN	Die Anzahl der numerierten Variablen vom Typ Homogene Transformation.
Anzahl Variablen vom Typ S	PC_NO_STR	Die Anzahl der numerierten Variablen vom Typ Zeichenfolge.
Genauigkeit Näherungsvergleich (*10^6)	PC_EPSILON	Näherungsvergleichsoperator (=) zulässige Abweichung (×1000000)
Verfolgungscode löschen	PC_NO_TRACE	Löscht den Debugcode aus dem Kompilierungscode. (0: nicht löschen, 2: ganz löschen.)
Berechnungscode für Zykluszeit löschen	PC_NO_CYC_TIME	Löscht den Berechnungscode für die Zykluszeit aus dem Kompilierungs-code. (0: nicht löschen, 1: teilweise löschen, 2: ganz löschen)
Feldbereich-Überprüfungscode löschen	PC_NO_CHK_INDEX	Löscht den Array-Prüfungscode aus dem Kompilierungscode. (0: nicht löschen, 1: teilweise löschen, 2: ganz löschen)
Fehlerunterbrechungsprozeßcode löschen	PC_NO_ERR_TRAP	Löscht den Unterbrechungsprozeßcode aus dem Kompilierungscode. (0: nicht löschen, 1: löschen)

■ Dio Manager - Hardware

Parametername	Makroname	Beschreibung
Zuordnungsmodus (0: kompatibel, 1: Standard)	IO_ASSIGN	E/A-Zuordnungsmodus Einstellung (0: kompatibler Modus, 1: Standardmodus)
Paritätsbit (0: ungültig, 1: gültig)	IO_PARITY	Einstellen der Programmnummern-Auswahlparität im Programmstart-Paritätsbit (0: ungültig, 1: gültig)
HandIO. Unterbrechungseinstellung (0: ungültig, 1: gültig)	IO_HD_ENABLED	Hand I/O Unterbrechungseinstellung (0: ungültig, 1: gültig)
DeviceNet. Eingabe-Steckplatznummer	IO_DN_IN_SLOT	DeviceNet Eingabe-Steckplatznummer
DeviceNet. Ausgabe-Steckplatznummer	IO_DN_OUT_SLOT	DeviceNet Ausgabe-Steckplatznummer
DeviceNet. Vorbereitungsfehlererkennung (0: ungültig, 1: gültig)	IO_DN_EXERRCHK	DeviceNet Vorbereitungsfehlererkennung (0: ungültig, 1: gültig)

■ Arm Manager - Pfaderstellung

Parametername	Makroname	Beschreibung
Positive Software-Bewegungsbegrenzung (J1,deg*10^3)	AM_JPRM_PLIM1	Positive Software-Bewegungsbegrenzung 1. Achse (×1000, Einheit: Grad)
Positive Software-Bewegungsbegrenzung (J2,deg*10^3)	AM_JPRM_PLIM2	Positive Software-Bewegungsbegrenzung 2. Achse (×1000, Einheit: Grad)
Positive Software-Bewegungsbegrenzung (J3,deg*10^3)	AM_JPRM_PLIM3	Positive Software-Bewegungsbegrenzung 3. Achse (×1000, Einheit: Grad)
Positive Software-Bewegungsbegrenzung (J4,deg*10^3)	AM_JPRM_PLIM4	Positive Software-Bewegungsbegrenzung 4. Achse (×1000, Einheit: Grad)
Positive Software-Bewegungsbegrenzung (J5,deg*10^3)	AM_JPRM_PLIM5	Positive Software-Bewegungsbegrenzung 5. Achse (×1000, Einheit: Grad)
Positive Software-Bewegungsbegrenzung (J6,deg*10^3)	AM_JPRM_PLIM6	Positive Software-Bewegungsbegrenzung 6. Achse (×1000, Einheit: Grad)
Positive Software-Bewegungsbegrenzung (J7,deg*10^3)	AM_JPRM_PLIM7	Positive Software-Bewegungsbegrenzung 7. Achse (×1000, Einheit: Grad)
Positive Software-Bewegungsbegrenzung (J8,deg*10^3)	AM_JPRM_PLIM8	Positive Software-Bewegungsbegrenzung 8. Achse (×1000, Einheit: Grad)
Negative Software-Bewegungsbegrenzung (J1,deg*10^3)	AM_JPRM_NLIM1	Negative Software-Bewegungsbegrenzung 1. Achse (×1000, Einheit: Grad)
Negative Software-Bewegungsbegrenzung (J2,deg*10^3)	AM_JPRM_NLIM2	Negative Software-Bewegungsbegrenzung 2. Achse (×1000, Einheit: Grad)
Negative Software-Bewegungsbegrenzung (J3,deg*10^3)	AM_JPRM_NLIM3	Negative Software-Bewegungsbegrenzung 3. Achse (×1000, Einheit: Grad)
Negative Software-Bewegungsbegrenzung (J4,deg*10^3)	AM_JPRM_NLIM4	Negative Software-Bewegungsbegrenzung 4. Achse (×1000, Einheit: Grad)
Negative Software-Bewegungsbegrenzung (J5,deg*10^3)	AM_JPRM_NLIM5	Negative Software-Bewegungsbegrenzung 5. Achse (×1000, Einheit: Grad)
Negative Software-Bewegungsbegrenzung (J6,deg*10^3)	AM_JPRM_NLIM6	Negative Software-Bewegungsbegrenzung 6. Achse (×1000, Einheit: Grad)
Negative Software-Bewegungsbegrenzung (J7,deg*10^3)	AM_JPRM_NLIM7	Negative Software-Bewegungsbegrenzung 7. Achse (×1000, Einheit: Grad)

Parametername	Makroname	Beschreibung
Negative Software-Bewegungsbegrenzung (J8,deg*10^3)	AM_JPRM_NLIM8	Negative Software-Bewegungsbegrenzung 8. Achse (×1000, Einheit: Grad)
BER (J1,deg*10^5)	AM_JPRM_RANG1	BER-Wert 1. Achse (×100000, Einheit: Grad)

■ Arm Manager - Bedingung verwenden

Parametername	Makroname	Beschreibung
Steuersatz der Bewegungsoptimierung	CND_ASPACC	Steuersatz der Bewegungsoptimierung (0: normal, 1: nur Punkt-zu-Punkt ist gültig, 2: nur Linear ist gültig, 3: Punkt-zu-Punkt und Linear sind gültig)
Bodenmontage oder Überkopfmontage	CND_SPACE	Einstellen von Bodenmontage oder Überkopfmontage (0: Bodenmontage, 1: Überkopfmontage)
Masse der Nutzlast (g)	CND_LOAD	Masse der Nutzlast (Einheit: g)
Nutzlastschwerpunkt X6 (mm)	CND_LDPOSGX	Komponente X des Nutzlastschwerpunktes (Einheit : mm)
Nutzlastschwerpunkt Y6 (mm)	CND_LDPOSGY	Komponente Y des Nutzlastschwerpunktes (Einheit : mm)
Nutzlastschwerpunkt Z6 (mm)	CND_LDPOSGZ	Komponente Z des Nutzlastschwerpunktes (Einheit : mm)
Zulässige Impulsbreite bei Stop (J1)	CND_PLSWDTH1	1. Achse, zulässige Impulsbreite bei Stop (Einheit: Impuls).
Zulässige Impulsbreite bei Stop (J2)	CND_PLSWDTH2	2. Achse, zulässige Impulsbreite bei Stop (Einheit: Impuls).
Zulässige Impulsbreite bei Stop (J3)	CND_PLSWDTH3	3. Achse, zulässige Impulsbreite bei Stop (Einheit: Impuls).
Zulässige Impulsbreite bei Stop (J4)	CND_PLSWDTH4	4. Achse, zulässige Impulsbreite bei Stop (Einheit: Impuls).
Zulässige Impulsbreite bei Stop (J5)	CND_PLSWDTH5	5. Achse, zulässige Impulsbreite bei Stop (Einheit: Impuls).
Zulässige Impulsbreite bei Stop (J6)	CND_PLSWDTH6	6. Achse, zulässige Impulsbreite bei Stop (Einheit: Impuls).
Zulässige Impulsbreite bei Stop (J7)	CND_PLSWDTH7	7. Achse, zulässige Impulsbreite bei Stop (Einheit: Impuls).
Zulässige Impulsbreite bei Stop (J8)	CND_PLSWDTH8	8. Achse, zulässige Impulsbreite bei Stop (Einheit: Impuls).
Bewegungsende-Zeitüberschreitung (ms)	CND_MVTIMOUT	Bewegungsende-Zeitüberschreitung bei Angabe von @E (Einheit: ms)

■ Vision Manager - Allgemeine Einstellung

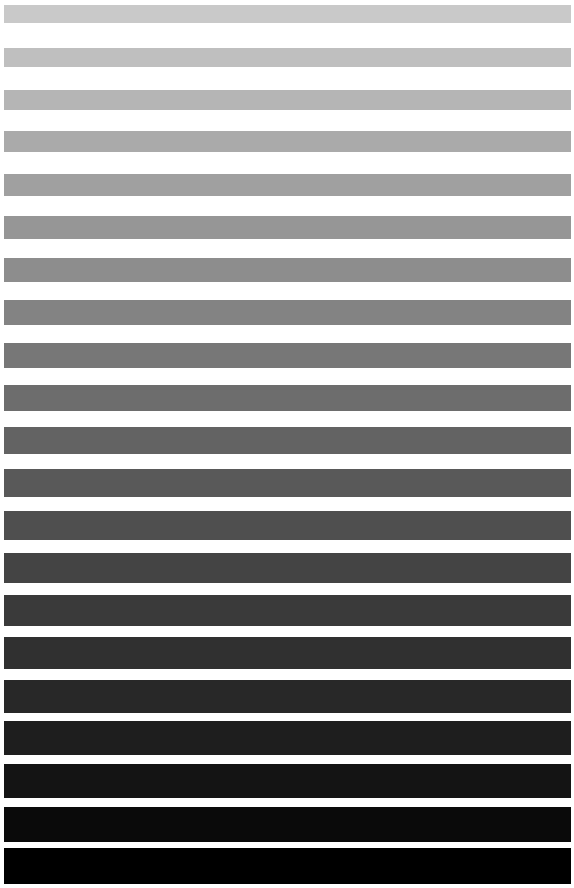
Parametername	Makroname	Beschreibung
Kamera 1 - Verschlußsystem	CA_SHUT1	Kamera 1 - Verschlußsystem (0: Feld, 1: Rahmen)
Kamera 1 - unterer Eingabegrenzwert	CA_LEVEL_L1	Kamera 1 - unterer Eingabegrenzwert
Kamera 1 - oberer Eingabegrenzwert	CA_LEVEL_H1	Kamera 1 - oberer Eingabegrenzwert
Kamera 1 - Kameraverfügbarkeit	CA_CONNECT1	Kamera 1- Verfügbarkeit zur Verwendung (0: zur Verwendung verfügbar, 1: nicht angeschlossen, 2: Option, 3: keine)
Kamera 2 - Verschlußsystem	CA_SHUT2	Kamera 2 - Verschlußsystem (0: Feld, 1: Rahmen)
Kamera 2 - unterer Eingabegrenzwert	CA_LEVEL_L2	Kamera 2 - unterer Eingabegrenzwert
Kamera 2 - oberer Eingabegrenzwert	CA_LEVEL_H2	Kamera 2 - oberer Eingabegrenzwert
Kamera 2 - Kameraverfügbarkeit	CA_CONNECT2	Kamera 2 - Verfügbarkeit zur Verwendung (0: zur Verwendung verfügbar, 1: nicht angeschlossen, 2: Option, 3: keine)
Kamerasynchronisierungssystem	CA_SYNC	Kamerasynchronisierungssystem (0: Kamera interne Synchronisation, 1: Kamera externe Synchronisation)
ShCorn - Distanz	SH_CORN_DIST1	ShCorner Befehl Meßbedingung (Distanz zwischen Kissen und Ecke) 1 bis 10
ShCorn - Steigung	SH_CORN_DIST2	ShCorner Befehl Meßbedingung (Steigung zwischen zwei Kissen) 1 bis 10
ShCorn - Breite	SH_CORN_WIDTH	ShCorner Befehl Meßbedingung (Pfadbreite) 1 bis 10
ShCorn - Höhe	SH_CORN_HEIGHT	ShCorner Befehl Meßbedingung (Pfadhöhe) 1 bis 10
ShCirc - Steigung	SH_CIRC_DIST	ShCircle Befehl Meßbedingung (Steigung zwischen zwei Kissen) 1 bis 10
ShCirc - Breite	SH_CIRC_WIDTH	ShCircle Befehl Meßbedingung (Pfadbreite) 1 bis 10
Such-Zeitüberschreitungswert (ms)	SH_TIMEOUT	Suchmessung-Zeitüberschreitungswert
Vision Monitor Anzeigeposition	VS_DISP_LOC	Eine Position zum Anzeigen des Prozeßfensters im Vision Monitor (0: Mitte, 1: links, 2: rechts)

■ Log Manager - Kommunikationsspezifikation

Parametername	Makroname	Beschreibung
Zeitüberschreitungswert (ms)	COM_RTK_TOUT	Zeitüberschreitungswert bei der Kommunikation (Einheit: ms)
Anzahl Wiederholungsversuche bei Zeitüberschreitung.	COM_RTK_TRETRY	Anzahl Wiederholungsversuche bei Kommunikationszeitüberschreitung.
Anzahl Wiederholungsversuche beim Auftreten einer NAK.	COM_RTK_RETRY	Die Anzahl der Wiederholversuche, wenn NAK (Paketfehler) auftritt.

Parametername	Makroname	Beschreibung
RS232C(2). Kommunikationspriorität	COM_RS2_ACCESS	Kommunikationprioritätseinstellung von RS232C-Kanal 2 des Controllers (0: nicht verfügbar, 1: nur Lesen, 2: Lesen/Schreiben verfügbar)
RS232C(2). Baudrate	COM_RS2_SPEED	Einstellung der Baudrate von RS232C-Kanal 2 des Controllers (Einheit: bps)
RS232C(2). Parität (0: gerade, 1: keine, 2: ungerade)	COM_RS2_PARITY	Paritätseinstellung des RS232C-Kanals 2 des Controllers (0: gerade, 1: keine, 2: ungerade)
RS232C(2).Datenlänge	COM_RS2_DATELEN	Einstellung der Datenlänge von RS232C-Kanal 2 des Controllers (Einheit: Bit)
RS232C(2). Stopbit (0: 1Bit, 1: 1.5 Bit, 2: 2Bit)	COM_RS2_STOPBIT	Stopbit-Einstellung des RS232C-Kanals 2 des Controllers (0: 1Bit, 1: 1.5Bit, 2: 2Bit)
RS232C(2).Begrenzer (0: CR, 1: CR + LF)	COM_RS2_DELIMITER	Begrenzereinstellung des RS232C-Kanals 2 des Controllers (0: CR, 1: CR + LF)
RS232C(3). Kommunikationspriorität	COM_RS3_ACCESS	Kommunikationsprioritätseinstellung von RS232C-Kanal 3 des Controllers (0: nicht verfügbar, 1: nur Lesen, 2: Lesen/Schreiben verfügbar)
RS232C(3). Baudrate	COM_RS3_SPEED	Einstellung der Baudrate von RS232C-Kanal 3 des Controllers (Einheit: bps)
RS232C(3). Parität (0: gerade, 1: keine, 2: ungerade)	COM_RS3_PARITY	Paritätseinstellung des RS232C-Kanals 3 des Controllers (0: gerade, 1: keine, 2: ungerade)
RS232C(3). Datenlänge	COM_RS3_DATELEN	Einstellung der Datenlänge von RS232C-Kanal 3 des Controllers (Einheit: Bit)
RS232C(3). Stopbit (0: 1Bit, 2: 1.5Bit, 3: 2Bit)	COM_RS3_STOPBIT	Stopbit-Einstellung des RS232C-Kanals 3 des Controllers (0: 1Bit, 1: 1.5Bit, 2: 2Bit)
RS232C(3). Begrenzer (0: CR, 1: CR + LF)	COM_RS3_DELIMITER	Begrenzereinstellung des RS232C-Kanals 3 des Controllers (0: CR, 1: CR + LF)
RS232C(4). Kommunikationspriorität	COM_RS4_ACCESS	Kommunikationsprioritätseinstellung von RS232C-Kanal 4 des Controllers (0: nicht verfügbar, 1: nur Lesen, 2: Lesen/Schreiben verfügbar)
RS232C(4). Baudrate	COM_RS4_SPEED	Einstellung der Baudrate von RS232C-Kanal 4 des Controllers (Einheit: bps)
RS232C(4). Parität (0: gerade, 1: keine, 2: ungerade)	COM_RS4_PARITY	Paritätseinstellung des RS232C-Kanals 4 des Controllers (0: gerade, 1: keine, 2: ungerade)
RS232C(4). Datenlänge	COM_RS4_DATELEN	Einstellung der Datenlänge von RS232C-Kanal 4 des Controller (Einheit: Bit)
RS232C(4). Stopbit (0: 1Bit, 1: 1.5Bit, 2: 2Bit)	COM_RS4_STOPBIT	Stopbit-Einstellung des RS232C-Kanals 4 des Controllers (0: 1Bit, 1: 1.5Bit, 2: 2Bit)
RS232C(4). Begrenzer (0: CR, 1: CR + LF)	COM_RS4_DELIMITER	Begrenzereinstellung des RS232C-Kanals 4 des Controllers (0: CR, 1: CR + LF)
EthernetKommunikationspriorität	COM_ET_ACCESS	Controller-Ethernet-Kommunikationseinstellung (0: nicht verfügbar, 1: nur Lesen, 2: Lesen/Schreiben verfügbar)
Ethernet. IP-Adresse	COM_ET_IP_ADDR	Controller-Ethernet IP-Adreßeinstellung
Ethernet. Subnetzmaske	COM_ET_SUBNET_MASK	Controller-Ethernet Subnetzmaskeneinstellung
Ethernet. Verbindungsziel 1. lokaler Anschluß	COM_ET_LPORT1	Verbindungsziel der Einstellung des Controllers Ethernet 1. lokaler Anschluß
Ethernet. Verbindungsziel 2. lokaler Anschluß	COM_ET_LPORT2	Verbindungsziel der Einstellung des Controllers Ethernet 2. lokaler Anschluß
Ethernet. Verbindungsziel 3. lokaler Anschluß	COM_ET_LPORT3	Verbindungsziel der Einstellung des Controllers Ethernet 3. lokaler Anschluß
Ethernet. Verbindungsziel 4. lokaler Anschluß	COM_ET_LPORT4	Verbindungsziel der Einstellung des Controllers Ethernet 4. lokaler Anschluß
Ethernet. Verbindungsziel 5. lokaler Anschluß	COM_ET_LPORT5	Verbindungsziel der Einstellung des Controllers Ethernet 5. lokaler Anschluß

Index



A

Abschlußziffer 8-6
Absolute Bewegung 3-1
Achsenkomponente 15-38
AchsenkomponenteZuweisung 10-11
Angabe der Geschwindigkeit 4-1
Anpassung des Palettierungsprogrammes 2-22
Anweisung 8-1, 13-1
Anweisung für Koordinatentransformation 8-18, 12-51
Arbeitskoordinatensystemdeklaration 8-9, 9-7
Arithmetische Funktion 15-1
Arithmetischer Operator 7-16
Armbewegung (Bibliothek) 22-50
Arm-Semaphor 14-14
Armstellungsfunktion 15-45
Armstellungskomponente 15-36
Array-Deklaration 8-8, 9-17
Array-peration 7-20
Art der Kante 21-60
Aufruf eines definierten Prozesses 8-15
Aufruf mit Bezug 8-26, 11-4
Aufruf mit einem Wert 8-25, 11-4
Aufrufen einer Unterroutine 2-1, 2-2
Aufrufen eines Programmes 2-1, 2-3
Ausdruck 7-3, 7-16
Ausführen eines Programmes 2-5
Auswahl 8-13

B

Bedingte Verzweigung 8-13, 11-17
Befehl 7-3
Befehl zur Kalibrierung der Bildverarbeitung 8-30
Befehle zum Abrufen von Ergebnissen 8-30
Befehle zur Bildverarbeitung 8-28
Befehle zur Fenstereinstellung 8-27
Befehle zur Suchfunktion 8-29
Benutzerkoordinatensystem-Deklaration 8-9
Bereich (Bildverarbeitungssteuerung) 5-2
Bereich für die Interferenzprüfung 9-2
Bereinigen des Kommunikationspuffers 2-8
Beschleunigungseinstellung 4-2
Bestätigen der erreichten Position 3-3
Bezeichnung 5-8
Bezeichnungsbefehle 8-29
Bibliothek 2-13
Bild-Ein-/Ausgabe 21-1
Bildverarbeitung 21-41
Bildverarbeitungssteuerung 5-1, 8-27, 21-1
Binärformat 7-12
Binärisierung (Bildverarbeitungssteuerung) 5-4
Bogeninterpolationssteuerung 3-12

C

Codeerkennung 21-64
Codeerkennungsbefehl 8-29
Codiererwert-Prüfbewegung 3-3

D

Datei einbinden 20-4
Datentyp 7-10
Datentypumwandlung 7-11
Deklaration einer Werkzeugkoordinatensystemnummer 9-6
Deklarationsanweisung 8-6, 9-1
Dezimalformat 7-12
DI/DO-Steuerungsanweisung 8-19
Die Robotersprache PAC 7-1
Distanzextraktion 15-35
Doppeltes Anführungszeichen 8-4
Durchlaufbewegung 3-3
Durchlaufstartposition 3-9
Durchlaufstartverschiebung 3-9
Durchschnittlicher Helligkeitswert 21-61

E

E/A (EIN/AUS) 7-10
E/A-Anschluß 13-1
E/A-Variablen-Deklaration 8-8
Einfaches Anführungszeichen 8-4
Eingabe/Ausgabe(Bibliothek) 22-49
Einheiten für die PAC-Sprache 7-21
Einstellen der Roboterinstallationsbedingung 4-17
Ellipsenfenster 21-16
Endebewegung 3-3
Endesignal des Palettierungsprogrammes 2-20
Entsprechungstabelle (VS) Anhang-22
Externe Beschleunigung/Verzögerung 4-1
Externe Geschwindigkeit 4-1

F

Fehlerinformationen 18-1
Fehlerprüfung 8-22, 18-1
Fehlerunterbrechung 18-4
Fenster (Bildverarbeitungssteuerung) 5-1
Fenstereinstellung 21-14
Festlegen der internen Beschleunigung und Verzögerung 4-2
Flußsteuerungsanweisung 8-13, 11-1
Funktion 7-8, 15-1
Funktionsdeklaration 8-9

G

Gelenk-Operation 7-20
Geschwindigkeitsumwandlung 15-22
Globale Variable 7-4, 7-5
Globale Variable Indirekter Bezug 7-6

H

Haltung 10-7
Handgelenk Anhang-2
Helligkeit 5-1
Helligkeitsintegralwert (Bildverarbeitungssteuerung) 5-7
Hexadezimalformat 7-13
Histogramm (Bildverarbeitungssteuerung) 5-4

I

Indirekter Bezug 7-6
Integrierte Konstanten 16-1
Interferenzprüfung 9-2
Interne Beschleunigung/Verzögerung 4-1
Interne Geschwindigkeit 4-1
Interpolationssteuerung 3-11

K

Kanalnummer 2-8
Kante (Bildverarbeitungssteuerung) 5-7
Kennung 7-3, 7-4, 7-8, 8-3
Kennzeichnung 21-67
Komma 8-4
Kommentar 11-23
Kommunikation zwischen Tasks 2-6
Kommunikationsbefehl 2-8
Kompatibilitätstabelle Anhang-27
Konstante 7-3, 7-12, 16-1
Konstante vom Typ Ganze Zahl 7-12
Konstante vom Typ Gelenk 7-15
Konstante vom Typ Numerischer Wert ... 7-12
Konstante vom Typ Position 7-14
Konstante vom Typ Real mit doppelter Genauigkeit 7-13
Konstante vom Typ Real mit einfacher Genauigkeit 7-13
Konstante vom Typ Vektor 7-14
Konstante vom Typ Zeichenfolge 7-14
Konventionelle Sprache 22-2
Koordinaten des Interferenzbereichs 9-2
Korrespondenzwinkel (Bildverarbeitungssteuerung) 5-9
Kreisfenster 21-14
Kreuzsymbol 21-36

L

Leerzeichen 8-4
Linearsteuerung 3-12
Linienfenster 21-15
Liste der reservierten Worte Anhang-20
Logischer Operator 7-18
Lokale Variable 7-4, 7-7
Lokale Variablen 9-8

M

Makrodefinitionen 20-1
Modusmethode (Bildverarbeitungssteuerung) 5-5
Multitasking 2-6
Multitasking-Steuerungsanweisung 14-1

N

Name 7-3
Numerischer Wert 7-10
Nutzlast 14, 15
Nutzlastschwerpunkt Anhang-14

O

Operator 7-3, 7-16
Optimierung 20-5

P

PAC (Programming Language for Assembly Cell) 7-1
PAC-Bibliothek 22-1
Palettierung 2-14, 22-18
Palettierungsbibliothek 2-14, 22-18
Palettierungsprogramm 2-21
Palettierungszähler 2-19
Parametereinstelltabelle Anhang-28
Pixel 5-1
P-Kachelungsmethode (Bildverarbeitungssteuerung) 5-6
Position-Operation 7-19
Präprozessor 8-24, 20-1
Priorität 2-6
Programm 7-9
Programmbank 2-13
Programmbibliothek 2-13, 22-1
Programmdeklaration 8-9
Programmname 7-9
Programmname und Deklaration 8-2, 9-1
Programmstop 11-1
Protokoll 19-4
Punkt-zu-Punkt-Steuerung 3-11

Q

QR-Code 21-64

R

Rechteckfenster 21-17
Rekursiver Programmaufruf 2-4
Relation zwischen Stufe und Kante..... 21-61
Relationaler Operator 7-17
Reserviertes Wort 8-5
Roboterarmsteuerung 8-17
Robotersteuerungsanweisung 5-1, 8-17, 12-1
Rotationskomponente 10-8, 15-41
RS232C-Steuerungsanweisung 8-19
Rundungsform 5-8

S

Schaltkreisnummer 2-8
Schwerpunkt 5-2, Anhang-14
Sektorfenster 21-16
Semaphor 2-6, 14-6
Semaphor-ID 14-6
Semaphor-Steuerungsanweisung 8-20
Semikolon 8-4
Serielle Kommunikation 2-8
Sicherheitsbereichsdeklaration 8-9
SLIM 7-2
Sprachelemente 7-3
Stellung 7-10
Sternchen 8-4
Steueranweisung 8-20
Steuerbefehle zur Bildein-/Ausgabe 8-27
Steuersatz 0 4-5
Steuersatz 1 4-8
Steuersatz 2 4-9
Steuersatz 3 4-10
Steuersatz der Bewegungsoptimierung Anhang-14
Steuerung der Armstellung 12-31
Steuerung von Zeit und Datum 8-21, 17-1
Steuerungsanweisung für Bewegung 5-1
Steuerungsanweisung für Bewegung 8-17
Steuerungsanweisung für die Ein/Ausgabe 8-19
Steuerungsanweisung für Geschwindigkeit... ..... 8-18, 12-38
Steuerungsanweisung für Programmiergerät 8-19, 13-13
Steuerungsanweisung für Stop 8-17
Steuerungsanweisung für Zeit 8-18
Suchen (Bildverarbeitungssteuerung) 5-9
Suchmodell (Bildverarbeitungssteuerung) 5-9
Symbolkonstanten 20-1
System 19-1
Systeminformationen 8-23, 19-1
Systemvariable 7-8

T

Task-Steuerung 14-1
Transformation von Daten des Typs Stellung 15-28
Trigonometrische Funktion 15-12
Typ Konstante 7-15
Typdeklaration 8-6
Typdeklarationsanweisung 8-7
Typdeklarationszeichen (Abschlußziffer) .. 8-6

U

Umgebungseinstellwerte Anhang-13
Unbedingte Verzweigung 11-21
Unterprogramm 2-1, 8-15
Unterscheidungsanalysemethode (Bildverarbeitungssteuerung) 5-5

V

Variable 7-4
Variablen 7-3, 10-1
Vektor 7-10, 15-24
Vektor-Operation 7-19
Verbindungswinkel 10-6
Verwenden von Zustandsparametern Anhang-14

W

Werkzeugkoordinatensystemdeklaration .. 8-9
Werkzeugoperation 22-44
Wiederholung 8-14, 11-9
Winkel der großen Achse (Bildverarbeitungssteuerung) 5-3
Winkelkomponente 15-37
Winkelumwandlung 15-19

Z

Zeichencodetabelle Anhang-1
Zeichenfolge 7-10
Zeichenfolgefunktion 15-50
Zeichenfolge-Operator 7-18
Zeichensatz 8-4
Zeichnen 21-23
Zeichnungsbefehle 8-28
Zeile 8-1
Zeitfunktion 15-23
Zuweisung für Typ Gelenk 8-11
Zuweisung für Typ Position 8-11
Zuweisung zum Anfahrvektor 8-12
Zuweisung zum Drehungsvektor 8-11
Zuweisung zum Orientierungsvektor 8-12
Zuweisung zum Positionsvektor 8-12
Zuweisung zur Achsenkomponente 8-11
Zuweisung zur Armstellungskomponente 8-11, 8-12
Zuweisung zur Armstellungs-Komponente 10-5
Zuweisung zur RX-Komponente 8-11
Zuweisung zur RY-Komponente 8-11
Zuweisung zur RZ-Komponente 8-11
Zuweisung zur X-Komponente 8-11
Zuweisung zur Y-Komponente 8-11
Zuweisung zur Z-Komponente 8-11
Zuweisungsanweisung 10-1
Zuweisungsanweisung 8-10
Zuweisungsanweisung für Ausgangsposition 8-12
Zuweisungsanweisung für Gelenk 8-11
Zuweisungsanweisung für Homogene Transformation 8-12
Zuweisungsanweisung für Homogene Transformation 8-12
Zuweisungsanweisung für numerischen Wert 8-10
Zuweisungsanweisung für Position 8-11
Zuweisungsanweisung für Stellung 8-11
Zuweisungsanweisung für Vektor 8-10, 10-2
Zuweisungsanweisung für Zeichenfolge 8-10
Zuweisungs-Operator 7-16

Vertikal ausgerichtete V*-D-SERIE
Horizontal ausgerichtete H*-D-SERIE
Kartesische Koordinaten XYC-4D-SERIE
Bildverarbeitungsgerät µVision-21-SERIE

PROGRAMMIERERHANDBUCH (V1.4)

Erste Auflage

April 2000

Zweite Auflage

Juni 2000

DENSO CORPORATION
Industrial Systems Product Division

6B20C

Zweck dieses Handbuchs ist es, genaue Informationen für die Handhabung und Bedienung des Roboters bereitzustellen. Falls Sie Fehler oder Auslassungen feststellen, teilen Sie uns dies bitte mit; ebenso sind Vorschläge zur allgemeinen Verbesserung dieses Handbuchs gerne willkommen.

DENSO übernimmt keinerlei Haftung für direkte oder indirekte Schäden, die durch die Anwendung der Informationen in diesem Handbuch entstehen.