Programmierung von Robotern

Die wichtigste Aufgabe des Anwenders:

Den Bewegungsablauf in den Roboter eingeben = den Roboter *programmieren*.

Online-Programmierung, erfolgt am eingeschalteten Roboter und benutzt dessen Steuerung

Offline-Programmierung, ohne Roboter, das Programm wird dann später in die Steuerung kopiert.

Online-Programmierung

Teach-In-Programmierung:
Der Roboter wird über einen Steuerknüppel und Tasten geführt, ausgewählte Bahnpunkte werden gespeichert.

Folgeprogrammierung, Vormachen-Nachmachen:
Programmierer führt den Effektor des Roboters, Steuerung kompensiert Kräfte und zeichnet Bahn auf.

Master-Slave-Programmierung:
Programmierer bedient kleines Modell des Roboters (Master), großer Roboter (Slave) folgt und zeichnet auf.

Teach-In-Programmirung

Häufigste Methode,
Steuerung besitzt *Handprogrammiergerät* , auch *Teach-Box, Teach Pendant*.

Teach-Box besitzt Steuerknüppel oder Steuerkugel oder 3D-Maus.

Beim Verfahren mit der Teach-Box können viele Parameter eingestellt werden, z.B.

- Koordinatensystem
- Werkzeug
- Bewegungsart

Praktische Hinweise (1)

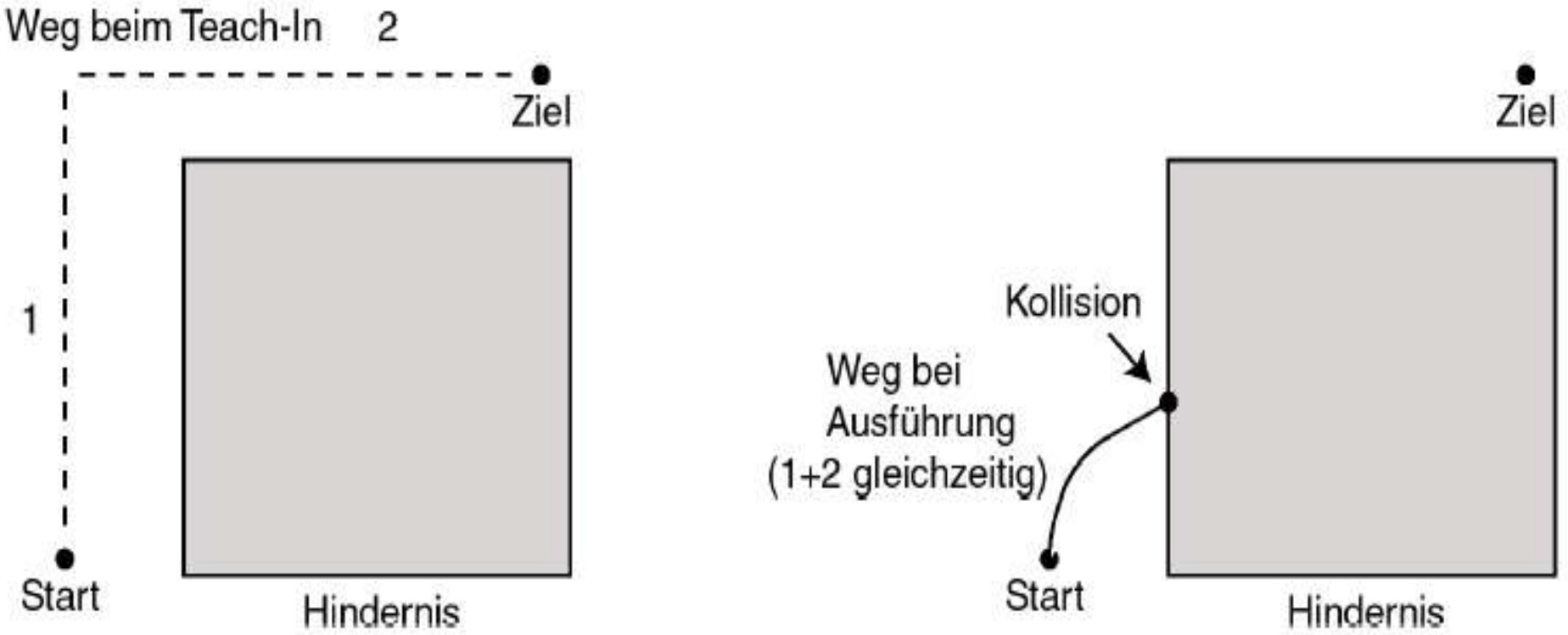


Abbildung 3.1: Der angefahrene Zwischenpunkt links oben wurde nicht eingespeichert, der Roboter steuert den Zielpunkt direkt an.

Praktische Hinweise (2)

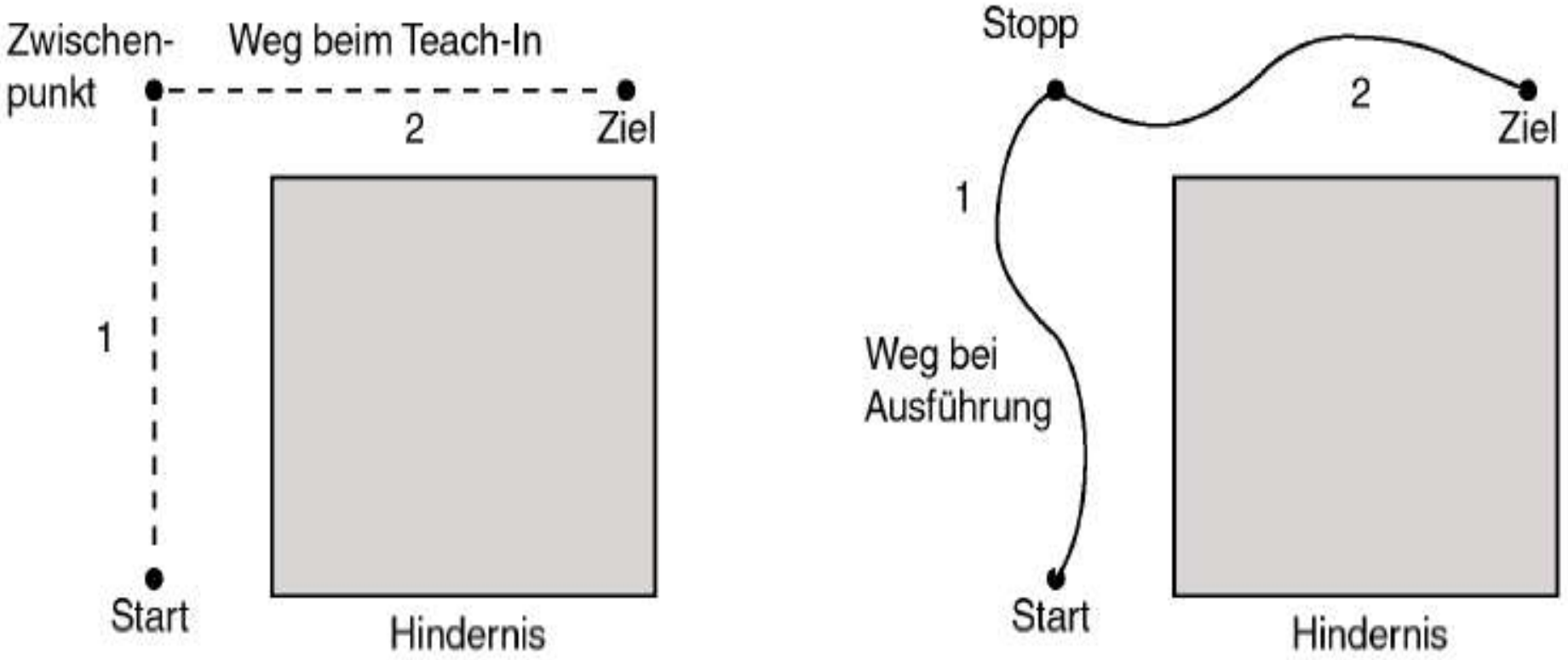
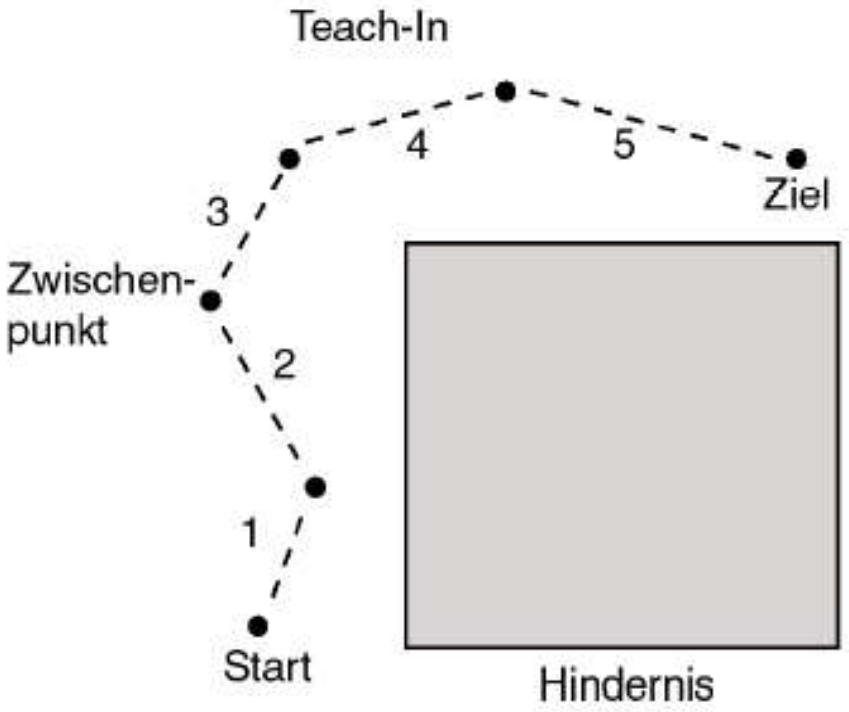
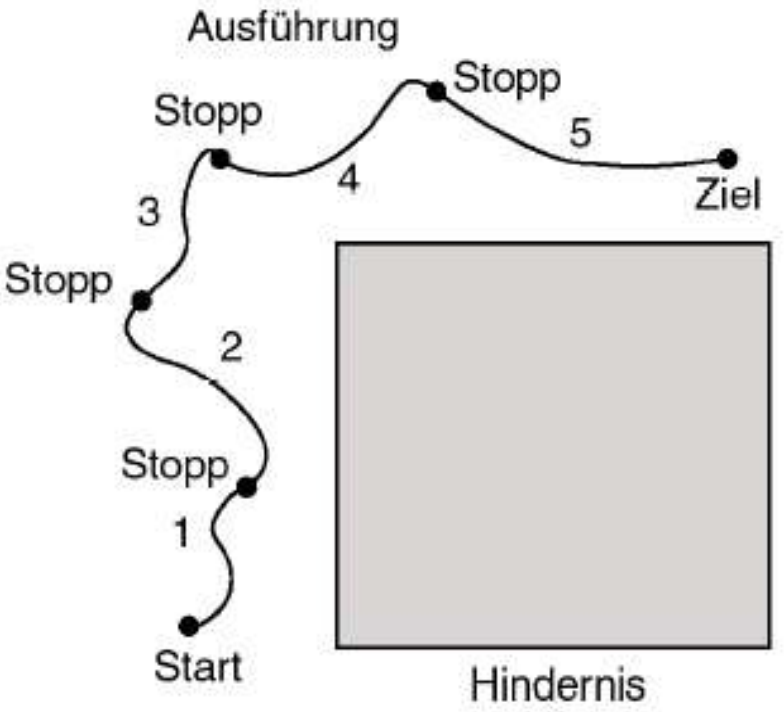


Abbildung 3.2: Der angefahrene Zwischenpunkt links wird eingespeichert, der Roboter umfährt nun wie gewünscht das Hindernis.

Praktische Hinweise (5)





Zusammenfassung praktische Tipps (1)

- Bei Punkt-zu-Punkt-Steuerung wird die Teach-In-Bahn nicht exakt wiederholt, dies erfolgt nur bei CP-Steuerung.
- Die Achsensteuerung fährt immer nur die eingespeicherten Achsenkoordinaten nächsten Zwischenpunktes direkt an. Beim Teach-In zwischenzeitlich eingenommene Stellungen werden dabei nicht mehr berücksichtigt.
- Für die Bewegung im freien Arbeitsraum werden wenig Zwischenpunkte gebraucht.
- In der Nähe der Werkobjekte oder in der Nähe von Hindernissen werden viele Zwischenpunkte gebraucht.

Zusammenfassung praktische Tipps (2)

- Vor dem Arbeiten an einem Werkobjekt empfiehlt es sich, eine *Annäherungsposition (Vorposition)* einzunehmen und sich von dort aus langsam und kontrolliert (CP-Bewegung) dem Werkobjekt zu nähern.
- Zu viele Zwischenpunkte erhöhen die Bewegungsdauer des Roboters.
- Zwischenpunkte ohne Überschleifen machen die Bewegung stockend.

Benutzung eines Werkzeugkoordinatensystems

Alle Bewegungsbefehle beziehen sich mit Bahnangabe, Geschwindigkeit und Beschleunigung auf den TCP (Werkzeugarbeitspunkt, Ursprung des Werkzeug-KS).

Bewegungen beim Teach-In können in Richtung der Werkzeugachsen geschehen, auch bei abgeknickten Werkzeugen.

Nach einem Werkzeugwechsel wird nur das neue Werkzeug eingemessen, das Programm wird nicht geändert.

Benutzung von Werkstückkoordinaten

Die Verwendung ist vorteilhaft, wenn sich die Bewegungen am Werkstück ausrichten.

Das Programm bleibt unverändert, wenn die Lage des Werkstückes sich ändert.

Verwendung von Werkstückkoordinaten am Beispiel

Aufgabe: Herausziehen eines Zylinders aus einer Bohrung. Zunächst: z-Achse des Werkstückkoordinaten-systems parallel zu Längsachse der Bohrung legen. Danach mehrere Möglichkeiten:

- Greifposition einteachen, Greifer schließen, auf Werkstückkoordinaten schalten, in z-Richtung verfahren bis Zylinder frei, diese Position teachen, Bewegungsart CP-linear zwischen dem ersten und dem zweiten Punkt.
- Werkzeug nach den Werkstückachsen ausrichten, entlang der z-Achse des Werkzeuges bewegen.
- Greifposition einteachen, rechnerisch einen zweiten Punkt erzeugen, der in z-Richtung der Werkstückkoordinaten gegen den ersten Punkt verschoben ist.

Benutzung eines Koordinatensystems an der Werkstückaufnahme

Dieses Koordinatensystem ist ein Zwischensystem, auf das sich die Werkstückkoordinatensysteme beziehen. (z .B. Mit dem Tisch verbunden)

Wenn die Position der Werkstückaufnahme verändert wird, braucht nur deren Koordinatensystem neu eingeteacht werden, alle Werkstücke sind weiterhin relativ dazu definiert.

Grenzen der Online-Programmierung

Situationen sind mit Online-Programmierung schlecht oder gar nicht zu realisieren, z. B. :

- Mathematische oder logische Berechnungen (z. B. Transformationen)
- Verarbeitung von Sensorsignalen
- Wiederholungsschleifen
- Programmverzweigungen
- Eingabe exakter Punkte, z. B. aus CAD-Daten
- Eingabe von Kommentaren

Robotersprachen

Die zusätzlichen Anforderungen erfüllen die *Robotersprachen.*

Die Roboterbahnen sind Programme (Anweisungslisten) die formal in der Robotersprache der jeweiligen Steuerung formuliert sind.

Die Verwendung von Robotersprachen ergab sich zum Teil zwanglos aus der Nachbearbeitung von Bahndaten.

Entstehung von Robotersprachen

Entstehung historisch:

1. Vollständiger Neuentwurf der Sprache unter besonderer Berücksichtigung der Robotik.
Bsp.: AL, VAL,

2. Ergänzung einer vorhandenen Programmiersprache, Bsp.: AUTOPASS, PASRO,

3. Erweiterung einer Automatisierungs- oder Steuersprache, Bsp.: RAPT.

Normung von Robotersprachen

Es gab Versuche, genormte Robotersprachen durchzusetzen, die aber alle gescheitert sind (IRL, PASRO, AL,...).

Herstellerspezifische Sprachen dominieren die Praxis:

- ABB: RAPID,
- Unimation/Stäubli: VAL, V+,
- KUKA: KRL.

Grund: Der starke Bezug der Robotersprachen auf die Steuerungshardware.

Die Handhabung von Roboterprogrammen

Die in der Online-Programmierung eingelernten Bahnen werden im Arbeitsspeicher als Roboterprogramm gespeichert und bilden somit eine logische Einheit für die Handhabung. Es ist möglich Roboterprogramme

- auf Datenträger zu sichern,
- von Datenträgern zu laden,
- einzusehen,
- zu modifizieren, z. B. Am PC.

Zugriff auf Roboterprogramme

Der Zugriff auf Roboterprogramme kann mit verschiedenen Werkzeugen erfolgen:

- Programmeditor
- Grafisches Werkzeug
- Simulationsprogramm
- Programmiergerät der Steuerung

Strukturierung von Roboterprogrammen

- Mehrere Befehle bilden eine Routine (Unterprogramm)
- Routinen können andere Routinen aufrufen
- Mehrere Routinen bilden eine Einheit (Unit oder Modul)
- Die Einheiten enthalten neben den Routinen auch Daten

Grundelemente von Robotersprachen

Befehle zur ...

- Steuerung des Programmablaufes
- Bewegung
- Bewegungseinstellung
- Digitale und analoge Signale
- Benutzerführung (User Interface)
- Kommunikation
- Mathematik

Befehle zur Steuerung des Programmablaufes

- Unterprogrammaufruf (CALL / CALLPROC / PROCCALL)
- Rücksprung aus Unterprogramm (RETURN ...)
- Einfacher Sprung (GOTO ...)
- Verzweigungen (IF ...)
- Zählschleifen (FOR ...)
- Bedingungsschleifen (WHILE)

Befehle zu Bewegungseinstellungen

- Höchstgeschwindigkeit
- Beschleunigung
- Verwaltung von Roboterkonfigurationen
- Nutzlastdefinition
- Verhalten in der Nähe von singulären Punkten
- Verschiebung von Positionen
- Feineinstellwerte für die Servokreise
- Weicheinstellung der Servokreise

Einfache Bewegungsbefehle

- Bewegungsart, PTP, CP-linear und CP-zirkular
- Endposition für Manipulator (und ggf. externe Achsen)
- gewünschte Geschwindigkeit
- Überschleifen: Nein oder Radius der Zone
- Werkzeugdaten (Datensatz zur Definition der Position des TCP)
- Werkstückdaten (Datensatz zur Definition des aktuellen Koordinatensystems)

Weitere Bewegungbefehle

Absolute achsenweise Bewegung

Relativbewegung unter Bezug auf einen bekannten Punkt

Bewegung des Roboters und Setzen eines digitalen Ausgang bei Erreichen der Überschleifzone

Bewegung des Roboters und Abarbeiten einer Routine bei Erreichen der Überschleifzone

Ausgabe digitaler Signale

- Wert eines digitalen Ausgangssignals festsetzen auf 1=HIGH oder 0=LOW,
- Alternativ: Ausgang „Setzen“ (HIGH) und „Rücksetzen“ (LOW)
- Invertieren des Wertes eines digitalen Ausgangssignals
- Generieren eines Impulses auf einem digitalen Ausgangssignal
- Änderung des Wertes einer Gruppe von digitalen Ausgangssignalen

Eingabe digitaler Signale

- Direktes Lesen eines digitalen Eingangs:
IF di1 = 1 THEN ...

- Direktes Lesen eines digitalen Gruppeneingangs
IF gi1 = 5 THEN ...

- Auf das Setzen oder Rücksetzen eines digitalen Eingangssignals warten.

- Testen, ob ein digitales Eingangssignal gesetzt ist.

Benutzerführung und Kommunikation

Kommunikation mit dem Programmiergerät:
Text einblenden, Funktionstaste lesen, Zahlenwert lesen,
Textbereich löschen etc.

Kommunikation mit einer seriellen Schnittstelle:
Konfigurieren und Öffnen, Schließen, Senden, Empfangen

Kommunikation über Netzwerkschnittstelle:
IP-Adresse vergeben, Senden, Empfangen,

Analoge Signale

- Direktes Lesen eines analogen Eingangs
IF ai1 > 5.2 THEN ...

- Änderung des Wertes eines analogen Ausgangssignals

Grundlegende mathematische Operationen

- Grundrechenarten,
- Winkelfunktionen,
- Inverse Winkelfunktionen,
- Exponentialwert,
- Quadratwurzel,
- Absolutbetrag,
- Rundung.

Spezielle mathematische Operationen

- Berechnung von Eulerschen Winkeln aus einer Orientierung
- Berechnung der Orientierung aus Eulerschen Winkeln
- Transformationsmatrizen invertieren
- Transformationsmatrizen multiplizieren
- Eine Transformationsmatrix und einen Vektor multiplizieren
- Berechnung der Länge eines Vektors.
- Berechnung des Skalarproduktes von zwei Vektoren.
- Berechnung des Vektorprodukts zweier Vektoren

Allgemeine Datentypen

- Logisch (Boole)
- Ganzzahl
- Fließkommazahl
- Zeichenkette

Spezielle Datentypen

Spezielle Datentypen für Geometrische Zusammenhänge

- Positionsvektor (x,y,z)
- Orientierung
- Koordinatensystem (=Frame, z. B. für Werkobjekt)
- Koordinatentransformation
- Roboterstellung (Position und Orientierung, Konfiguration)

Spezielle Datentypen

Spezielle Datentypen die Bewegungssteuerung

- Achsenposition und Achsenkonfiguration
- Bewegungseinstellungen
- Lastdatensatz
- Geschwindigkeitsdatensatz
- Endpunktdatensatz
- Werkzeugdatensatz
- Datensatz für Einstellung der Servokreise

Beispiel: Datentyp pose (=Frame, Transformationsmatrix) (RAPID)

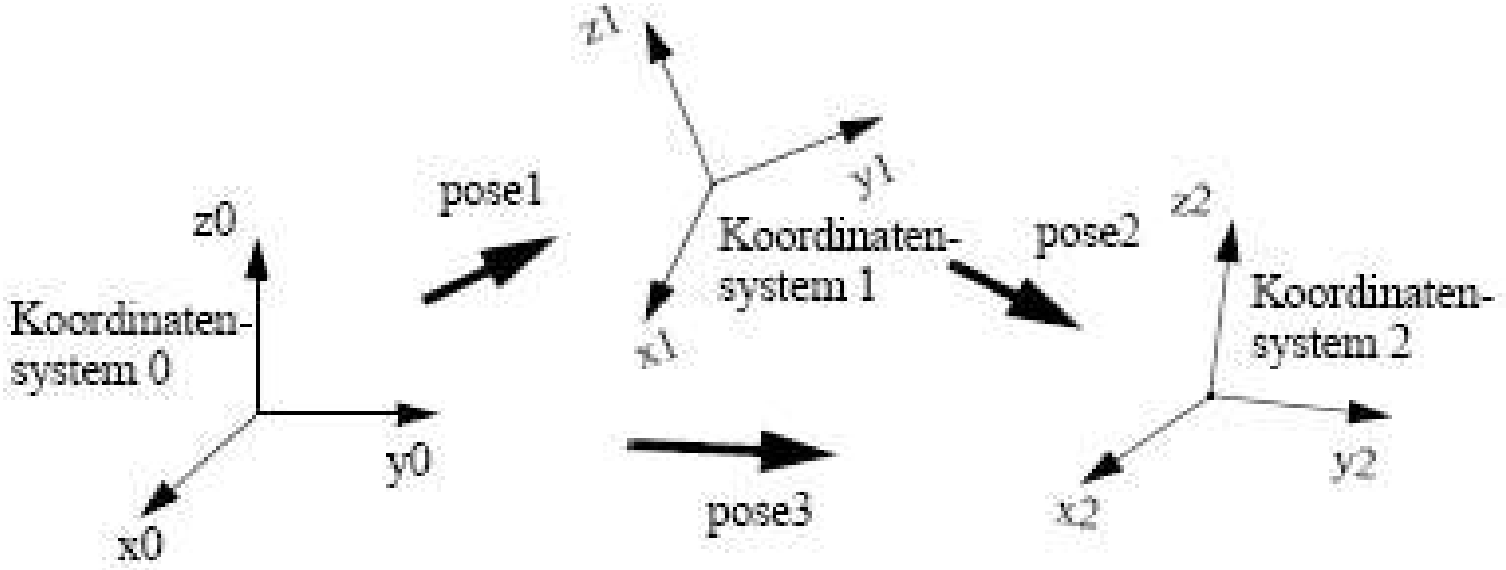
pose Koordinatentransformationen
Pose wird für den Übergang von einem Koordinatensystem auf ein anderes verwendet.

Beschreibung
Die Datentyp *pose* beschreibt die Verschiebung und Drehung eines Koordinatensystems in ein anderes Koordinatensystem. Die Daten können zum Beispiel angeben, wie das Werkzeug-Koordinatensystem mit Bezug auf das Handgelenk-Koordinatensystem positioniert und orientiert ist.

Komponenten
trans *(Verschiebung)* Datentyp: *pos*
Die Verschiebung der Position (X, Y und Z) des Koord. systems.
rot *(Drehung)* Datentyp: *orient*
Die Drehung des Koordinatensystems.

Quelle: Rapid Handbuch,ABB

Beispiel: Multiplikation von Transformationsmatrizen



pose1 stellt die Koordinatenmaße von Koordinatensystem 1 bezogen auf Koordinatensystem 0 dar.
pose2 stellt die Koordinatenmaße von Koordinatensystem 2 bezogen auf Koordinatensystem 1 dar.

Die zu pose3 führende Umformung, d.h. die Koordinatenmaße von Koordinatensystem 2 bezogen auf Koordinatensystem 0 wird durch das Produkt der beiden Umformungen erreicht:

```
VAR pose pose1;
VAR pose pose2;
VAR pose pose3;
.
.
pose3 := PoseMult(pose1, pose2);
```

Abbildung:
ABB Robotics GmbH

Beispiel: Datentyp Robtarget (RAPID)

robtarget Positionsdaten
Robtarget (*robot target*) wird zur Definition der Position von Manipulator und externen Achsen verwendet. Komponenten:

trans *(Verschiebung)* Datentyp: *pos*
Die Position (X, Y und Z) des TCP

rot *(Drehung)* Datentyp: *orient*
Die Werkzeugorientierung als Quaternion ausgedrückt (q1, q2, q3 und q4).

robconf *(Achskonfiguration)* Datentyp: *confdata*
Die Konfiguration der Manipulatorachsen (cf1, cf4, cf6 und cfx). Als aktueller Winkelquadrant von Achse 1, Achse4 und Achse6.

extax *(Externe Achsen)* Datentyp: *extjoint*
Die Position der externen Achsen.

Beispiel: Datentyp Robtarget (RAPID) (Fortsetzung)

```
CONST robtarget p15 := [ [600, 500, 225.3], [1, 0, 0, 0],
                         [1, 1, 0, 0], [ 11, 12.3, 9E9, 9E9, 9E9, 9E9] ];
```

Eine Position p15 wird wie folgt definiert:

- Als Position des Manipulators: x = 600, y = 500 und z = 225,3 mm im Werkstück-Koordinatensystem.

- Die Werkzeugorientierung erfolgt in der gleichen Richtung wie im Werkstück-Koordinatensystem.

- Die Achsenkonfiguration des Manipulators: Achsen 1 und 4 befinden sich in Position 90 bis180°, Achse 6 in Position 0 bis 90°.

Beispiel: Datentyp tooldata (RAPID)

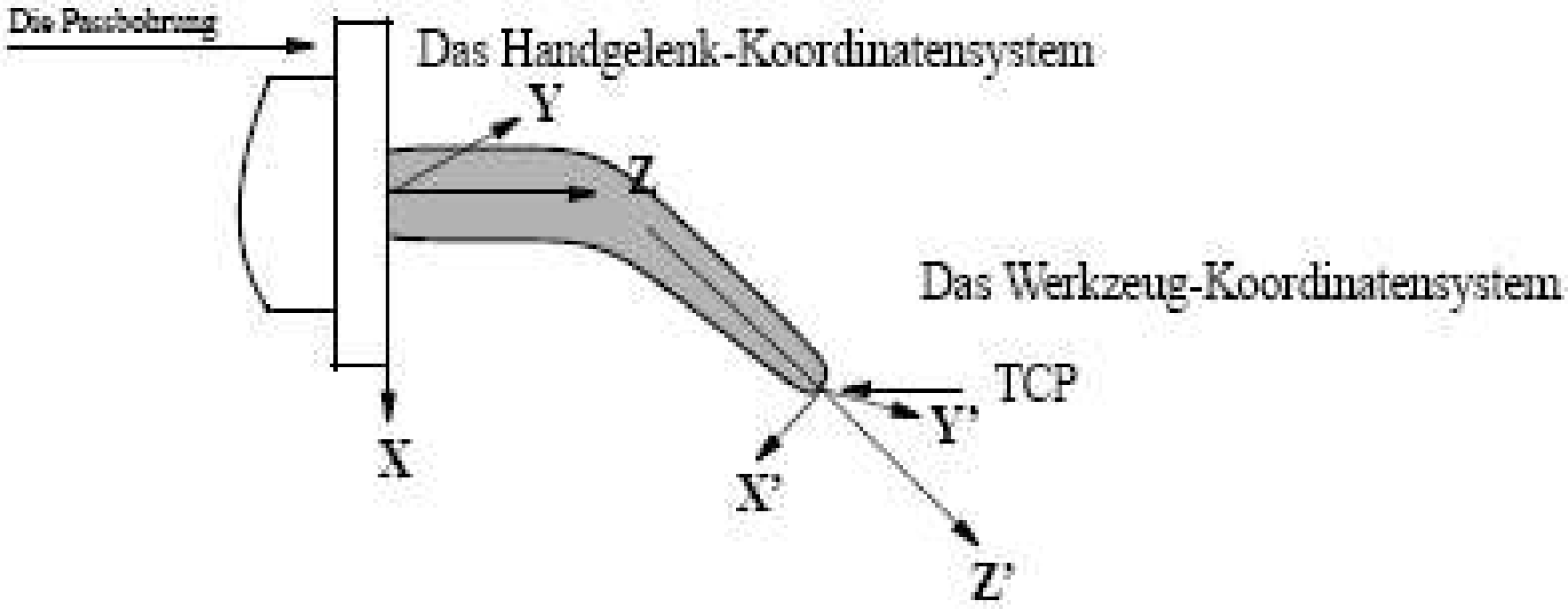
tooldata Werkzeugdaten
Tooldata wird verwendet, um die Kenngrößen eines Werkzeugs, zu beschreiben.Komponenten:

robhold *(robot hold)* Datentyp: *bool*
Definiert, ob der Manipulator das Werkzeug hält oder nicht:
- *TRUE ->* ja, der Manipulator hält das Werkzeug.
- *FALSE ->* nein, d.h. ein feststehendes Werkzeug.

tframe *(tool frame)* Datentyp: *pose*
Das Werkzeug-Koordinatensystem, d.h.:
- die Position des TCP´s (X, Y und Z) in mm, ausgedrückt im Handgelenks-Koordinatensystem.
- die Orientierung des Werkzeug-KS, ausgedrückt im Handgelenks-KS als Quaternion (q1, q2, q3 und q4)

tload *(tool load)* Datentyp: *loaddata*
Die Werkzeuglast.

Beispiel: Datentyp tooldata (RAPID) (Forts.)



Die Definition des Werkzeug-Koordinatensystems relativ zum Handgelenk-Koordinatensystem.

Abbildung: ABB Robotics GmbH

Datentyp loaddata (RAPID)

loaddata Lastdaten
Loaddata wird verwendet, um die an der mechanischen Schnittstelle des Manipulators (dem Handflansch des Manipulators) befestigte Last zu beschreiben. Komponenten:

mass Datentyp: *num*
Das Gewicht der Last in kg.

cog *(centre of gravity)* Datentyp: *pos*
Der Massenschwerpunkt der Nutzlast (x, y, z in mm), ausgedrückt im Werkzeug-Koordinatensystem.

aom *(axes of moment)* Datentyp: *orient*
Die Orientierung des Koordinatensystems, definiert durch die Trägheitsachsen der Nutzlast, ausgedrückt im Werkzeug-KS.

Spezielle Befehlsgruppen

Installation von Interruptroutinen (Interruptroutine definieren, Interrupts aktivieren/deaktivieren)

Routinen zur Fehlerbehandlung (Aufruf wiederholen, Aufruf übergehen, Fehler hochreichen, etc.)

Multitasking

Weltzonen

Synchronisation von Routinen mit Bahnpunkten

Behandlung von Singularitäten

Kollisionserkennung

Zusatz-Softwarepakete

Stark herstellerabhängig, Beispiele sind Zusatzpakete für:

Punktschweißen

Bogenschweißen

Schweißnahtverfolgung

Kleben

Lackieren