Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig angefertigt, alle benutzten Quellen und Hilfsmittel vollständig und genau angegeben habe. Ich habe alles kenntlich gemacht, was ich aus Arbeiten anderer unverändert oder mit Änderungen übernommen habe.

Mannheim, 15. April 2002

Christoph Hyllus

Vorwort

An dieser Stelle möchte ich mich bei all jenen bedanken, die mich bei der Durchführung dieser Diplomarbeit unterstützt haben.
Mein erster Dank gilt Prof. Dr. Matthias Seitz, der die Durchführung dieser Diplomarbeit ermöglicht hatte, sowie für die Betreuung und Korrektur der Diplomarbeit
Besonders danken möchte ich Herrn Hans Peter für all seine Lösungstipps für angefallene Probleme der Visual – Basic – Programmierung.
Ein weiterer Dank gilt der Firma EUROBTEC, die mich bei Fragen im Bereich des Low – Level - Protokolls unterstützt hat.
Ein weiterer Dank geht an Prof. Dr. R.- D. Brückbauer für die Zweitkorrektur dieser Arbeit.

Mannheim, 15. April 2002

Inhaltsverzeichnis

1 Einleitung

1.1 Problemumfeld

Der Anfang der Roboter geht ins 17. und 18. Jahrhundert zurück. In diesem Zeitraum gab es eine Anzahl genialer Erfindungen. Die meisten davon galten der Unterhaltung.

Etwa 1750 wurden verschiedene mechanische Musikautomaten in menschlicher Größe gebaut. 15 Jahre später wurde eine mechanische Puppe konstruiert, die zeichnen konnte [3]. Damals, ähnlich wie heute, nahmen diese Erfindungen die Tierwelt oder uns Menschen als Grundidee und Vorbild. Die Bewegung der Menschen und Tiere wurde auch bei der Entwicklung der Roboter genutzt. Viele mobile Roboter bewegen sich heute ähnlich wie Insekten und in automatisierten Fabriken sieht man oft Bewegung der Industrieroboter, die sehr der des menschlichen Armes ähnelt (Abbildung 1.1).

Der Begriff „Roboter“ wurde 1920 von dem tschechoslowakischen Schriftsteller Karel Capek geprägt. Die ersten Roboter danach kennen die meisten Menschen aus den Science - Fiction – Erzählungen und Filmen. Der erste kommerzielle Roboter wurden erst 1959 vorgestellt. Die Steuerung erfolgte durch Begrenzungsschalter und Kurvenscheiben. Ein Jahr später wurde ein Roboter des Typs „Unimate“ vorgestellt. Er benutzte zur Steuerung die Prinzipien der numerischen Steuerung und wurde hydraulisch angetrieben.

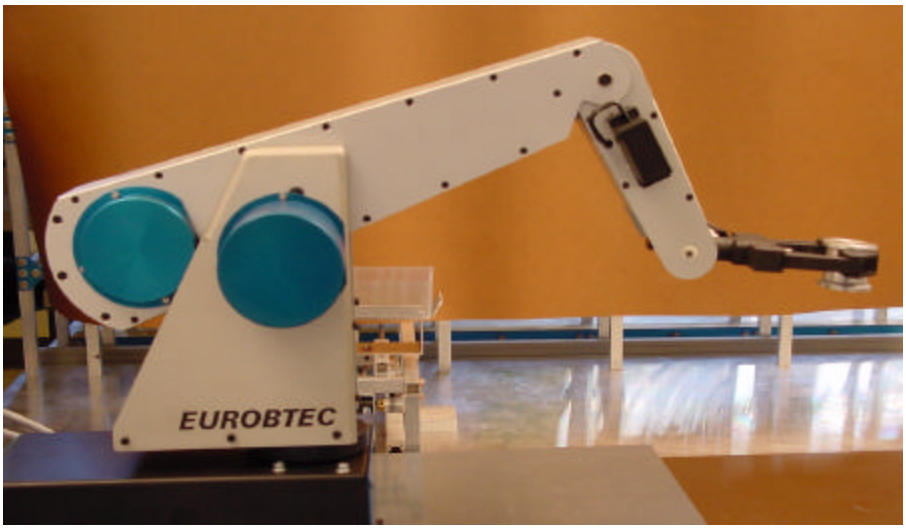


Abbildung 1.1 : Trainingsroboter ROB3 der Firma EUROBTEC

Im darauffolgenden Jahr wurde ein Roboter des gleichen Typs bei Ford Motor Company installiert. Er arbeitete mit einer Druckgussmaschine zusammen.
Die meiste Verwendung fanden die Roboter bei der Montage bei der Autoindustrie.
So auch 1978 der PUMA - Roboter (Programmierbare Universalmaschine für Montagezwecke), der aus den Entwürfen von General Motors stammt [3].
In der heutigen Realität sind Roboter hochautomatisierte Handhabungsgeräte, die von Computern gesteuert, aber immer noch von uns Menschen gebaut und programmiert werden.
Die Handhabungsgeräte können nach folgendem Schema unterteilt werden:

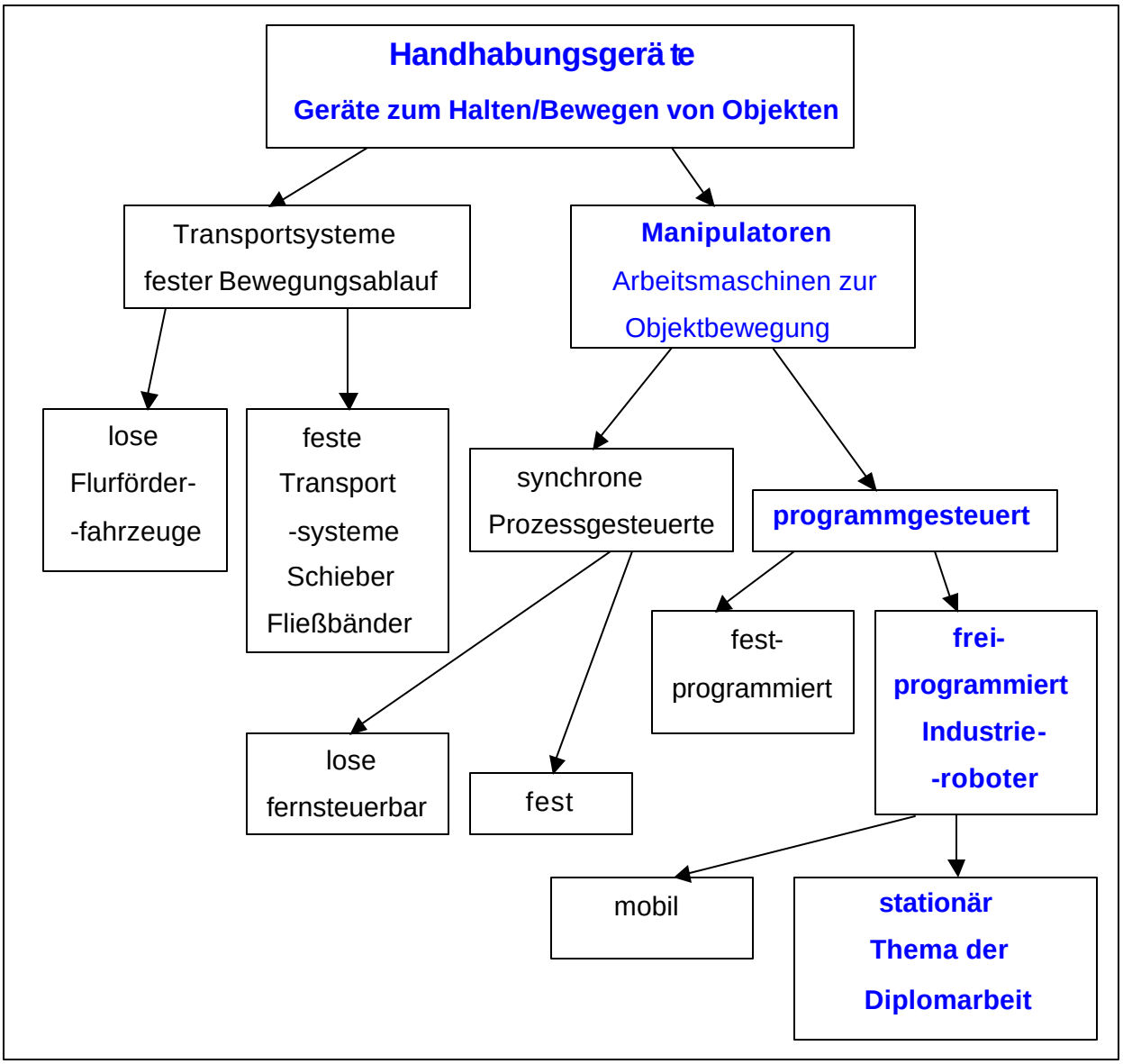


Abbildung 1.2 : Unterteilung der Handhabungsgeräte [5]

1.1.1 Entwicklung der Roboter

Bleiben wir aber bei der heutigen Wirklichkeit, bei dem Industrieroboter.
Nach VDI-Richtlinie 2860 (von 1981) sind Industrieroboter universal einsetzbare Bewegungsautomaten mit mehreren Achsen, deren Bewegung hinsichtlich Bewegungsfolge und Wegen bzw. Winkeln frei programmierbar und gegebenenfalls sensorgeführt wird. Sie sind mit Greifern, Werkzeugen oder anderen Fertigungsmitteln ausrüstbar und können Handhabungs- und/oder Fertigungsaufgaben ausführen [4].

Die industrielle Entwicklung und das Verlangen nach Erzeugnissen in großen Stückzahlen führten zur Automatisierung der Maschinen. Unser industrielles Zeitalter, angefangen bei der Energietechnik, ist geprägt von Produktivitätssteigerungen und damit verbundenen Fragen: wie schnell und wie günstig, kann man ein Produkt herstellen ohne dabei an der Qualität zu verlieren.
Um diese Möglichkeiten zu steigern, muss eine Maschine mit mehr Kraft, unermüdlich und schnell menschliche Tätigkeiten nachahmen.
In vielen Industriezweigen ist das fast gelungen. Durch ausgefallene Sensorik schaffte man Hallen voller Industrieroboter, je nach Bestellung und Produkt variabel programmiert, überwacht von einer handvoll Techniker, wo früher Dutzende Arbeiter einer monotonen Beschäftigung nachgingen.
Die wichtigsten Gründe auf Handhabungstechnik zurückzugreifen sind:

- **Wirtschaftlichkeit**, Schnelligkeit der Roboter mit ihrer Gewichtskraft kann von keinem Menschen ersetzt werden. Dazu kommen die ständig steigenden Lohnkosten.
- **Arbeitsschutz**, in vielen Industriebereichen kann es durch schädliche Stoffe wie Säure oder Dämpfe sehr schnell zu Verletzungen der Arbeiter kommen.
- **Qualitätssicherung**, viele Werkstücke sollen während der Bearbeitung nicht mit dem Handschweiß in Berührung kommen, damit es nicht zur unerwünschten Korrosion kommt.
- **Miniaturisierung**, die Größe mancher Objekte (Chip Fertigung) überschreitet die Grenzen menschlicher Präzision, was die Positioniergenauigkeit und die Arbeitsgeschwindigkeit betrifft [1].

So steigt die Anzahl der gebauten und eingesetzten Roboter im Industriebereich seit 1980 exponenzial. Nach United Nations Economic Commission for Europe in Zusammenarbeit mit der International Federation of Robotics (FR) im Bericht **"World Robotics 2000"** über den Stand und die Zukunft der Robotik soll es in der Industrie weltweit 742500 Roboter geben [16].

1.1.2 Anwendungsbereiche der Roboter

Einer der wichtigsten Gründe Roboter einzusetzen ist die Kosteneinsparung. Das beste Beispiel wirtschaftlich zu arbeiten ist der größte Service-Roboter der Welt. Er heißt „Skywash“ und wurde im September 1995 an die Lufthansa ausgeliefert. Wenn es um das Flugzeugwaschen geht, ist der „Skywash“ unschlagbar. Ohne Murren und mit gleich bleibender Präzision verrichtet er jeden Tag seine Arbeit.
Der Manipulatorarm kann mit seinen elf programmierbaren Achsen das Bürstensystem mit einer halben Tonne Nutzlast sicher und millimetergenau um den Rumpf des Jumbos führen. Dabei braucht er für die Reinigung 3 Stunden statt bislang 90 Mannstunden. Die bedeutet Einsparung von 12500 € bei jedem Waschen [18].

Abbildung 1.3 : Skywash beim Flugzeugputzen [17]

Ein anderer mobiler Roboter, „KURT II“ verrichtet seine Dienste bei der autonomen Inspektion kommunaler Abwasser. Trotzdem dominieren die in der Industrie eingesetzten Roboter gegenüber den Dienstleistungsrobotern.
Neben den oben genannten Gebieten werden Roboter auch in Spezialbereichen eingesetzt, wie die bei Expeditionen im All oder bei schwer zugänglichen Bergungsarbeiten. Mobile Roboter werden bei Militär und Polizei z.B. zur Bombenentschärfung eingesetzt, wo bei direkten Einsätzen menschliches Leben in Gefahr ist.
Nach Delphi – Prognosen (zusammengefasste Schätzungen von mehreren Wissenschaftlern) werden verstärkt Roboter in nächster Zukunft auf folgenden Gebieten eingesetzt:

- Roboter für Pflege (Alten, Kinder, Kranken, Blinden),
- Löschroboter für Brandbekämpfung und Menschenrettung,
- Roboter für Bergbau, für Tunnelprojekte, und
- Roboter als Außenskelette für Gelähmte [1].

1.2 Aufgabenstellung

1.2.1 Die Schwerpunkte der Diplomarbeit

Die Hauptziele der Diplomarbeit können folgendermaßen unterteilt werden.

1. Es soll die Entwicklung und die Notwendigkeit der Roboter erläutert werden.

2. Der Roboterarm soll mit seinem Endeffektor Positionen anfahren können (Pick&Place Demo).

2.1 Um den Roboterarm anzusteuern muss die Schnittstelle zum Antriebscontroller programmiert werden.

2.2 Die Vorgabe der Sollwerte soll entweder von der erstellten Bedien- und Beobachtungsoberfläche (B&B) oder durch ein SPS – Programm erfolgen.

2.3 Um die Sollwerte vom SPS – Programm zu vergeben, muss eine weitere Schnittstelle zur B&B programmiert werden (DDE - Schnittstelle).

2.4 Sollwerte sollen als Weltkoordinate vorgegeben werden. Diese sollen mit Hilfe der Rückwärtstransformation in Winkelkoordinate umgerechnet werden. Umgekehrt, sollen die Istwerte vom Roboter mit Vorwärtstransformation von Winkelkoordinaten in Weltkoordinate umgerechnet werden.

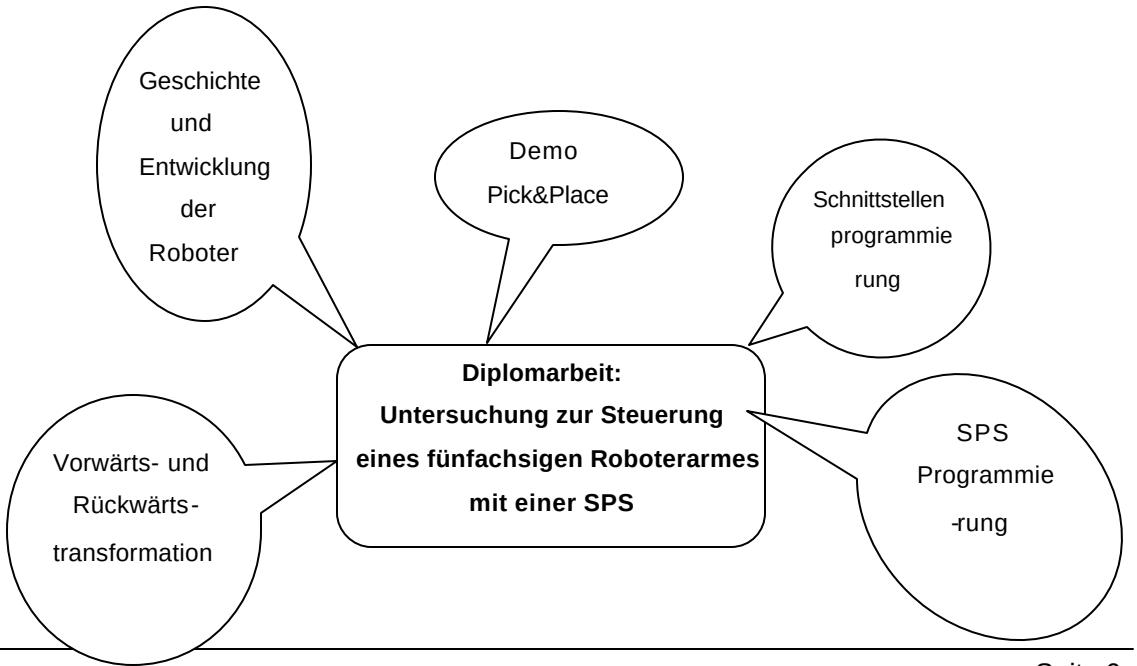


Abbildung 1.4 : Hauptziele der Diplomarbeit

Das Ziel der Diplomarbeit ist die Ansteuerung des Roboterarmes ROB 3 und Verbindung dieser Ansteuerung mit einer speicherprogrammierbaren Steuerung (Software SPS ProSys 1131 der Firma Deltalogic).

Die eigentliche Problematik dabei ist die Programmierung der Schnittstelle, sowie die technischen Möglichkeiten dieser Anbindungen.

Zur Ansteuerung des Roboterarmes steht das Low – Level - Protokoll der Firma EUROBTEC zu Verfügung. Dabei wird der Antriebscontroller (Prozessor 8031) des Roboterarmes über die serielle Schnittstelle RS 232 (COM1) angesprochen, um dann die einzelnen Achsen mit Hilfe der Gleichstromservomotoren zu positionieren.

Nach Betrachten der technischen Möglichkeiten, die Software – SPS mit einem Programm zu verbinden, das wiederum die serielle Schnittstelle ansprechen kann, blieb als Lösung die DDE - Verbindung. Die DDE – Verbindung erlaubt gleichzeitig den Datentausch zwischen mehreren Konversationspartnern.

Die Programmierumgebung Visual Basic (Version 6.0), die in der Diplomarbeit benutzt wurde, ermöglicht sowohl die Verbindung zu seriellen Schnittstelle als auch von einer Windows Applikation zu anderen.

Des Weiteren soll die SPS die Position des Endeffektors in Weltkoordinatensystem (x, y, z) an den Antriebscontroller vorgeben. Da der Antriebscontroller die Werte nur Achsweise in Byteform verarbeitet, müssen die Sollwerte von der SPS zuerst in die Achsenwerte zurücktransformiert und dann an den Antriebscontroller weitergeleitet werden.

Neben den oben genannten Aufgaben gibt die Arbeit einen Einblick in das Umfeld Robotik. Der Einblick, zu dem die Kinematik, die Ansteuerungsmöglichkeiten sowie Programmierungsmöglichkeiten eines Roboters gehören, soll dem Leser helfen, die Zusammenhänge und die Vorgehensweise bei der Arbeit zu verstehen.

1.3 Überblick über die Arbeit

Folgender Überblick soll dem Leser die Möglichkeit geben, sich über die Diplomarbeit einen Überblick zu verschaffen. Die einzelnen Kapitel werden hier kurz beschrieben.

Innerhalb des zweiten Kapitels werden das Prinzip und der Hardwareaufbau der Ansteuerung eines Roboters beschrieben.

In dem Kapitel 3 wird der mechanische Aufbau des Roboterarms beschrieben. Neben der Konstruktion wird auch der Arbeitsraum anhand des in der Diplomarbeit verwendeten Roboterarmes erläutert.
Zum Inhalt des Kapitels 3 gehören die Vor– und Rückwärtstransformationen von Gelenkkoordinaten in die Weltkoordinaten. Die Transformationen beziehen sich auf den fünfachsigen Roboterarm ROB 3.

Im Kapitel 4 werden die allgemeinen Methoden zur Programmierung eines Manipulators erläutert.

Nachdem die allgemeine Programmierungsmethoden eines Roboters im Kapitel 4 erläutet wurden, bezieht sich die Programmierung im Kapitel 5 spezifisch auf den Trainingsroboter ROB 3. Im weiteren Teil des Kapitels werden das Low – Level –Protokoll und die zugehörigen Kommandos erklärt.

Das Kapitel 6 soll dem Leser einen Überblick über die SPS - Programmierung verschaffen. Als Programmiersprache werden hier die Ablaufsprache und Strukturierter Text erklärt.

Kapitel 7 beschäftigt sich mit Visual – Basic - Programmierung.
Im weiteren Verlauf des Kapitels wird die Programmierung von DDE – und COM 1(RS 232)- Schnittstelle erläutet.
Im Kapitel 8 werden die Visual - Basic – und SPS -Programme anhand von Bilder und Struktogrammen erläutet.
Die Zusammenfassung der Ergebnisse erfolgt im Kapitel 9. Hier werden auch die Ergebnisse bewertet und ein Ausblick auf weiterführende Arbeiten gegeben.

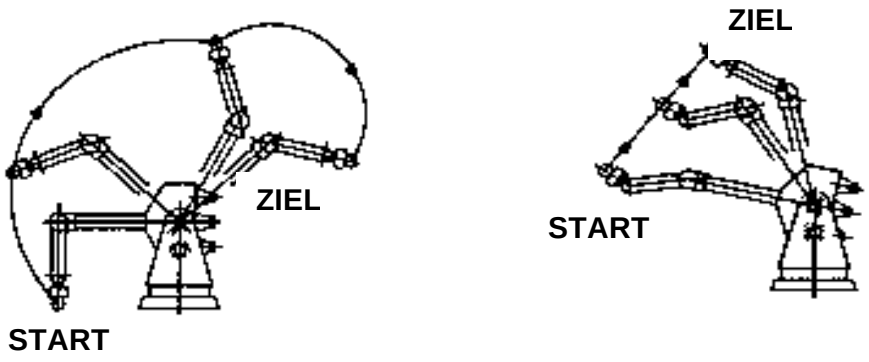
2 Die Ansteuerung eines Roboters

Bei der Ansteuerung eines Roboters gibt es 2 Arten von Steuerungen:

- Punkt-zu-Punkt-Steuerung (PtP – Steuerung, PtP = point to point) und
- Bahnsteuerung (CP – Steuerung, CP = continuous path).

Die beiden Arten von Steuerungen unterscheiden sich danach, wie die Endpunkte angefahren werden. Bei PtP – Steuerung fährt jede Achse mit maximaler Geschwindigkeit auf ihre Endposition. Die Bahn des Endeffektors ist bei dieser Methode durch die unterschiedlichen maximalen Bewegungsgeschwindigkeiten der Achsen sehr schwer abzuschätzen.

Bei der Bahnsteuerung kommt es bis zum erreichen des Endpunktes zu einer geradlinigen Bewegung. In der Praxis kann mit Hilfe einer CP – Steuerung ein Roboter längst seiner Bahn Schweißen oder Entgraten. Die Steuerung berechnet dabei die Zwischenpunkte (Interpolation). Der hohe Rechenaufwand führt zu einer Verlangsamung der Achsenbewegungen.



PtP - Steuerung CP - Steuerung

Abbildung 2.1 : Bewegungsarten eines Roboters[1]

Die Komplexität der Steuerung ist auf der Abbildung 2.2 auf der nächsten Seite zu sehen.

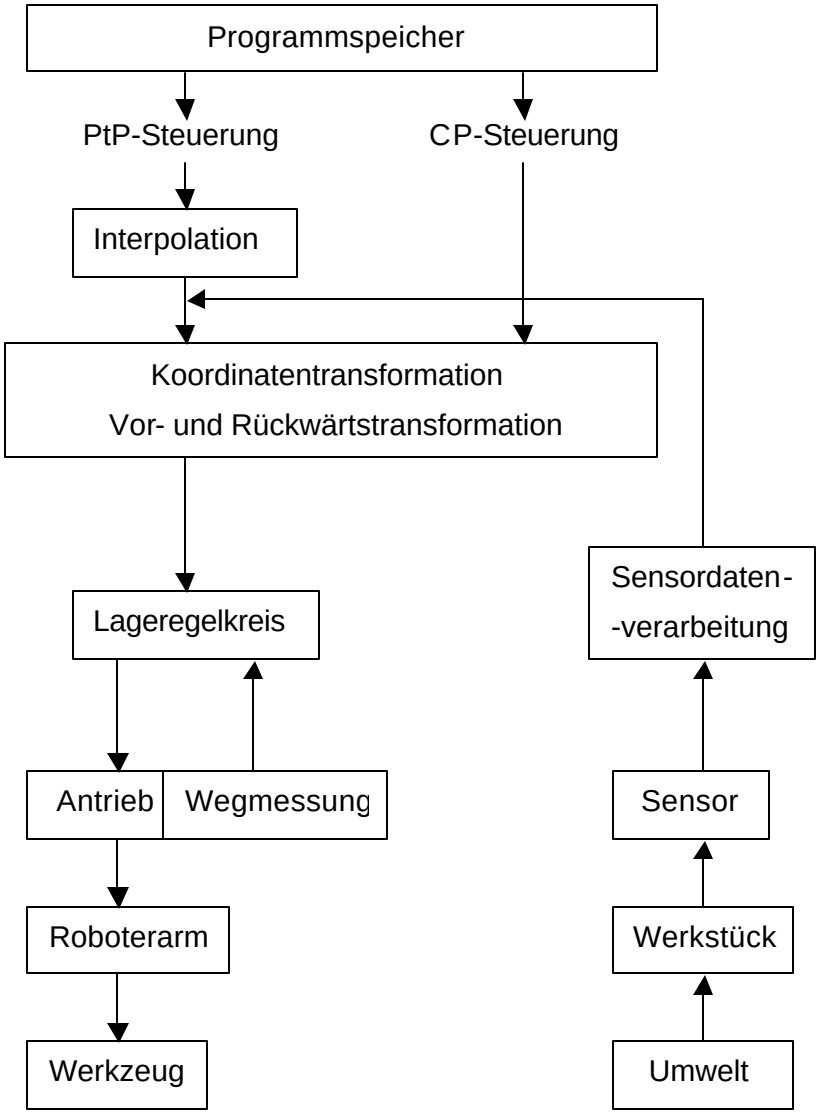


Abbildung 2.2 : Informationsfluss bei einer Robotersteuerung [1]

3 Kinematik eines Manipulators

Die Ansteuerung eines Roboters ist verbunden mit der Bewegungslehre (Kinematik), der Antriebstechnik, der Sensorik, der damit verbundenen Messtechnik und mit der Programmierung (wird umfangreichend in nächstem Kapitel beschrieben).

Dieses Kapitel beschreibt die Kinematik des in der Diplomarbeit verwendeten Roboterarmes ROB 3.

In der Robotik beschreibt der Begriff „Kinematik“ die Bewegungsmöglichkeiten und tatsächlichen Bewegungen eines Körpers mit Hilfe geeigneter Koordinaten.

Ein Industrieroboter setzt sich aus mehreren Gelenken zusammen. Die Gelenke können entweder um die Bewegungsachse gedreht oder entlang verschoben werden (z.B. Schlitten).

Man unterscheidet dadurch zwischen zwei Arten von Grundbewegungen:

- translatorische Bewegung
- rotatorische Bewegung

Die Bewegungen einzelner Achsen bilden zusammen so genannte kinematische Ketten. Es gibt offene und geschlossene kinematische Ketten. Im Fall des Roboterarmes handelt es sich um eine offene kinematische Kette auch „Freiarm“ genannt, da das letzte Glied nicht im geschlossenen Kreis mit dem ersten Glied verbunden ist (Abb. 3.1 links). Die große Beweglichkeit des letzten Gelenkes bringt auch ein Nachteil mit: das eigene Schwerkraftmoment muss kompensiert werden.

Bei der geschlossenen kinematischen Ketten stützten sich die Gewichtskräfte günstiger ab (Abb. 3.1 rechts)[1]. Anstelle eines Greifers können hier auch zwei Roboterarme von beiden Seiten ein Werkstück festhalten und somit eine geschlossene kinematische Kette bilden.

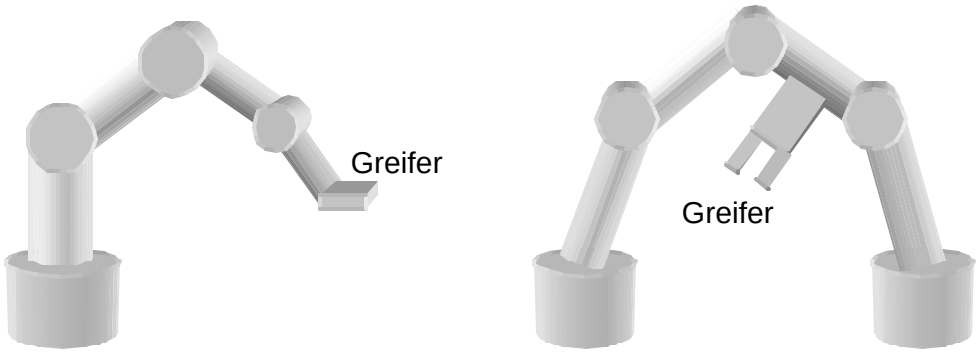


Abbildung 3.1 : Kinematische Ketten

3.1 Die Anatomie eines Roboters

Die Anatomie eines Roboters beschäftigt sich mit der physikalischen Konstruktion der Maschine. Ein Industrieroboter besteht immer aus dem mechanischen Teil, der Leistungselektronik und der Steuerung.
Die meisten der heute eingesetzten Industrieroboter sind fest auf einer Basis montiert (fest montierte Manipulatoren). Die Bewegung der einzelnen Teile eines Roboters geschieht über 6 (manchmal 5) Gelenke. Jedes Gelenk stellt einen **Freiheitsgrad** dar.

Die unterschiedlichen Anforderungen, die an Roboter gestellt werden, führen zu zusätzlichen Einrichtungen, die an einem Roboter montiert und konfiguriert werden müssen. Zu diesen Peripheriegeräten gehören die Endeffektoren und Sensoren (z.B. Tastsensoren oder Grenzschalter)
Der **Endeffektor** ist eine am letzten Gelenk befestigte Einrichtung. Dieser Endpunkt der ganzen Konstruktion wird manchmal auch als die „Hand“ des Roboters bezeichnet. Man unterscheidet zwischen zwei Gruppen von Endeffektoren: Greifer und Werkzeuge. Greifer werden zum Fassen eines Objektes verwendet. Als Werkzeug kann je nach Aufgabe z.B. eine Spritzpistole, ein Bohrer oder eine Schweißvorrichtung montiert werden.

3.1.1 Aufbau des Roboters ROB 3

Der Roboterarm ROB 3 (siehe Abbildung 3.2) ist aus 5 – Gelenken (Achsen) und einem Greifer aufgebaut und besitzt somit 5 Freiheitsgrade. Der mechanische Teil des Roboters wird nach dem Vorbild des menschlichen Skeletts aufgebaut. Analog dazu sind auch die Bezeichnungen der Achsen 1 bis 5 mit:

- Achse 1 – Körperdrehung
- Achse 2 – Schulter
- Achse 3 – Ellbogen
- Achse 4 – Handgelenk
- Achse 5 – Handdrehung [8].

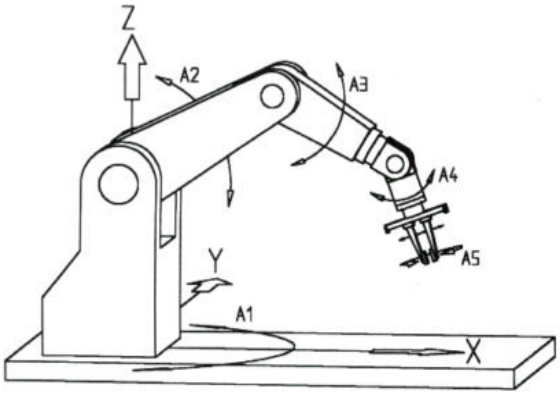


Abbildung 3.2 : Fünfachsige Trainingsroboter ROB 3

Die Anatomie des ROB3 erlaubt, ausgehend von der Grundposition (Abb. 3.3) folgende Winkelbereiche:

Gelenk	Winkel	erreichbare Bereiche	Auflösung
Achse 1	q1	+80°..0°..-80°	0…255
Achse 2	q2	+70°..0°..-30°	0…255
Achse 3	q3	0°..-100°	0…255
Achse 4	q4	+100°..0°..-100°	0…255
Achse 5	q5	+100°..0°..-100°	0…255
Greifer	-	0....60mm	0…255

Tabelle 3.1 : Winkelbereiche des ROB3

Der Winkelbereich der Achse 2 wurde vom Verfasser im Visual Basic – Programm +70°..-15° begrenzt. Der Grund war die mögliche Kollisionsgefahr des Roboterarmes (siehe dazu Abbildung 3.4).

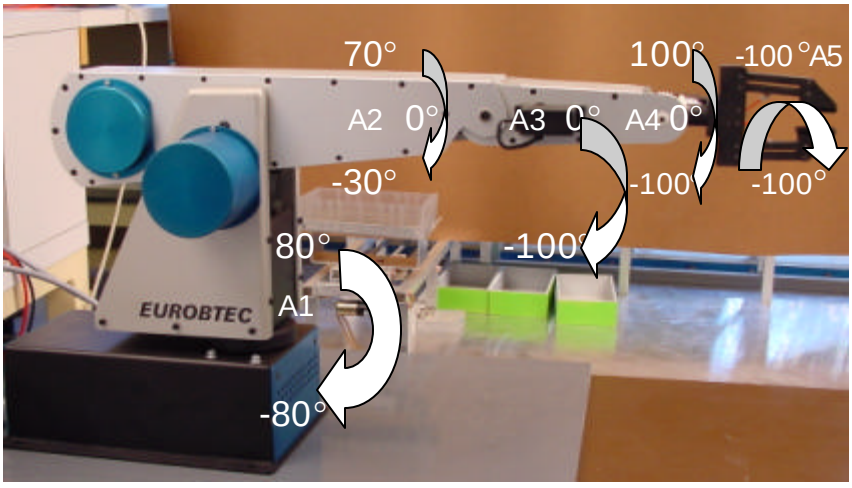
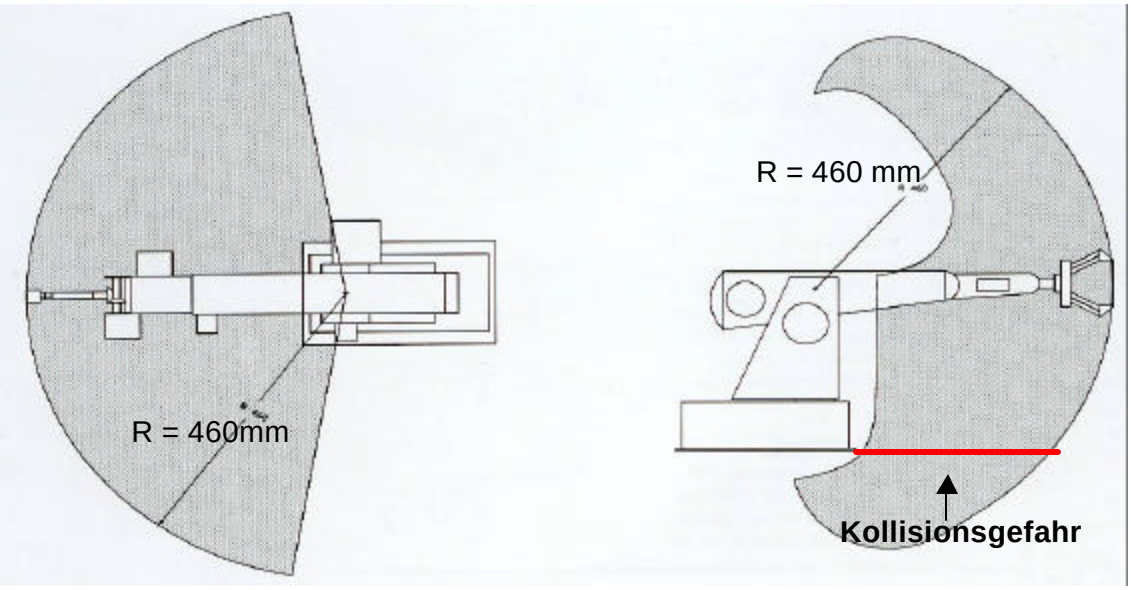


Abbildung 3.3 : Grundstellung des Trainingsroboters mit ROB 3

3.2 Arbeitsraum des Roboters

Der Arbeitsraum (engl. working range) eines Roboters ist ein wichtiges Kriterium beim Beschreiben der Leistung eines Roboters. Er beschreibt alle vom Endeffektor erreichbare Stellen im Raum. Die Stellung des Endeffektors wird auch **Tool Center Point** (**TCP**) genannt, wobei es sich dabei um das Zentrum des Endeffektors handelt. Hier wird das benötigte Werkzeug montiert. Je nach Anwendung kann es z. B. eine Spritzpistole oder ein Greifer zum Heben von Maschinenteilen sein. Je mehr Stellen ein Manipulator anfahren kann, desto hochwertiger ist er und kann effektiver eingesetzt werden.



Draufsicht Seitenansicht

Abbildung 3.4 : Der Arbeitsraum des Roboters ROB3 [8]

Die Abbildung 3.4 zeigt den Arbeitsraum des Trainingsroboters ROB 3 der Firma Eurobtec. Der mechanische Aufbau erlaubt es den Endeffektor des Roboterarmes auch in einen möglichen Kollisionsbereich (z. B. Tischunterkante) zu steuern, was die Beschädigung des Roboters zur Folge haben kann und durch entsprechende Programmierung vermieden werden muss.

3.3 Transformation in Welt- und Achsenkoordinatensysteme

Zur Positionierung und Orientierung von Werkstücken (TCP) durch einen Roboter in seinem Arbeitsraum braucht dieser eine exakte Position, die er mit seinem Endeffektor anfahren soll. Diese Position wird auch bei der Kommunikation des Roboters mit einer Maschine oder einem System (z. B. eine speicherprogrammierbare Steuerung) benötigt. Um zu vermeiden, dass die Systeme die Kinematik des Roboters kennen müssen, erfolgt die Übergabe der Position von TCP universell mit kartesischen Koordinaten $\underline{P}_{TCP} = (x, y, z)^T$.

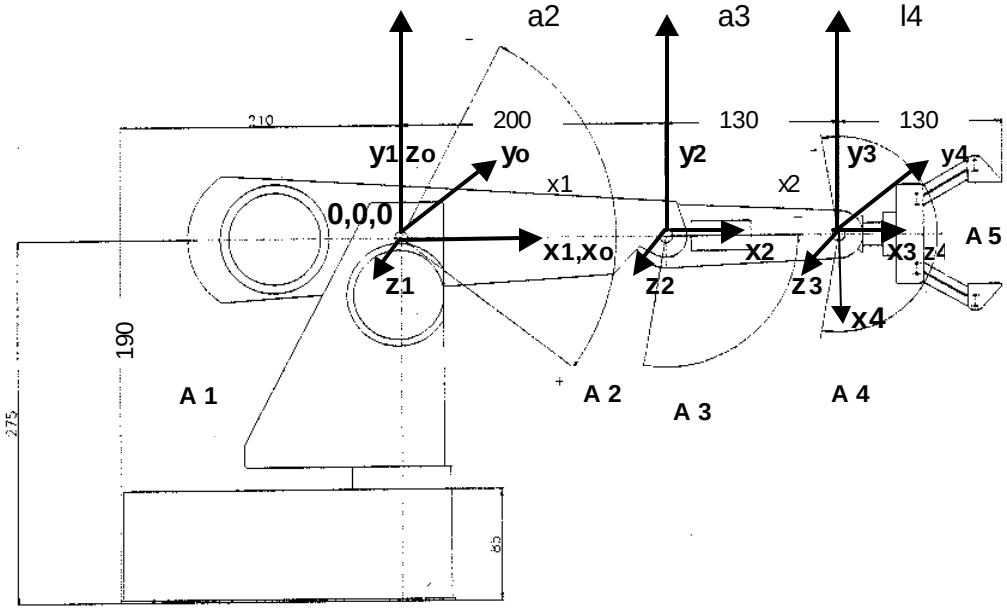


Abbildung 3.5 : Grundstellung des Roboterarmes mit Weltkoordinaten [7]

Der Roboterarm wird jedoch durch seine Gelenke bewegt und benötigt deshalb bei einer Drehbewegung achsenspezifische Sollwerte. Damit ergibt sich bei einem fünfachsigen Roboterarm der TCP als Funktion der Winkel der einzelnen Achsen:

$$\underline{P}_{TCP} = \underline{f}(\partial_1, \partial_2, \partial_3, \partial_4, \partial_5)^T. \qquad (3.1)$$

Mit den Winkeln hat man die Position des TCP, aber keine Orientierung des Greifers selbst, die Greiferlage(Abb. 3.6). Um die Greiferlage zu erfassen müssen die drei Orientierungswinkel y (xe, Gieren), J (ye, Nicken), j (ze, Rollen) vorgegeben werden.

$$\underline{P}_{TCP} = \underline{f}(\partial_{1\text{-}6})^T * Rot_X(y) * Rot_Y(J) * Rot_Z(j) \qquad (3.2)$$

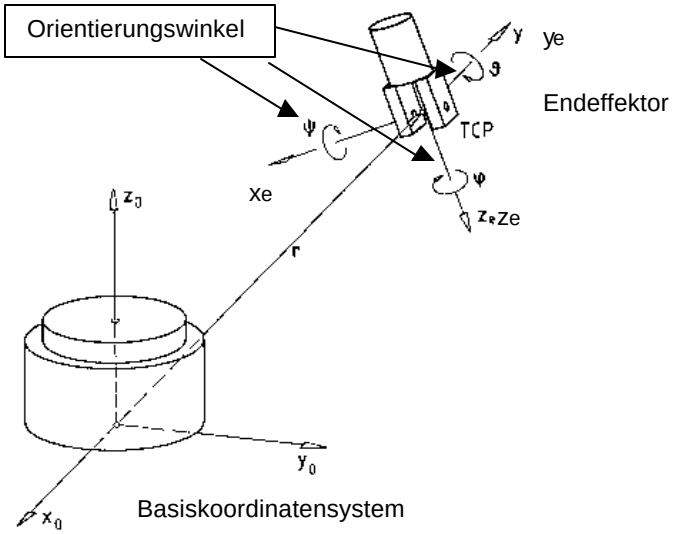


Abbildung 3.6 : Endeffektor mit Orientierungswinkel [5]

Hieraus ergibt sich die Forderung, die kartesischen Koordinaten in Roboterkoordinaten (Achskoordinatensystem) umzurechnen.

Man bezeichnet diese Berechnung als **Rückwärtstransformation** oder **inverse Transformation**, $\partial = f^{-1}(\underline{p})$

Umgekehrt liefert der Roboterarm die Istwerte der einzelnen Achsen als Rückmeldungen nach dem Erreichen der Positionen als Gelenkwerte, die wiederum in kartesische Koordinaten umgerechtet werden müssen. Dieser Vorgang wird **Vorwärtstransformation** oder **direkte Transformation** genannt $\underline{p} = f(\partial)$.

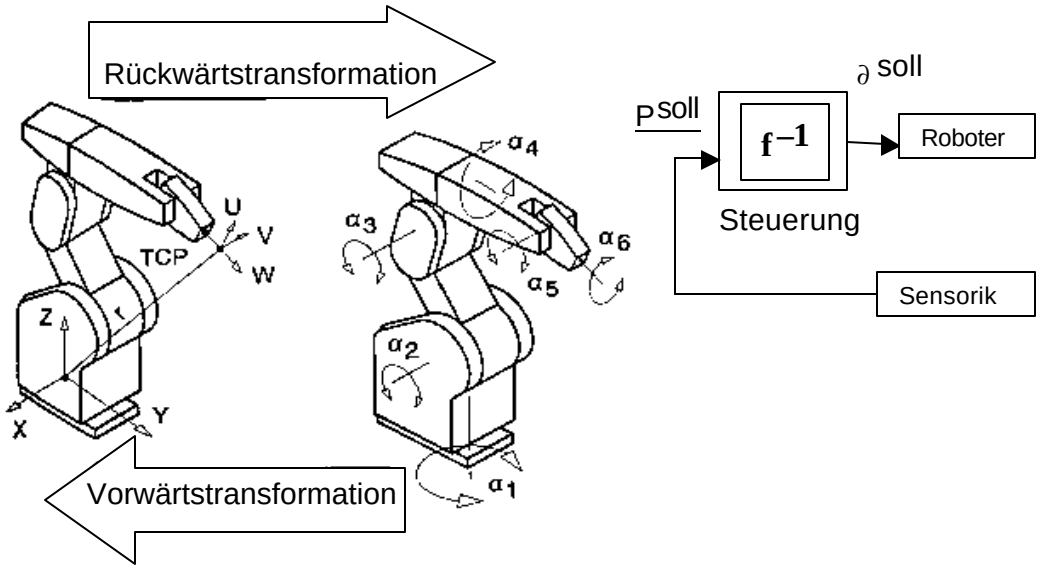


Abbildung 3.7 : Transformationen bei Robotersteuerung [1]

Wie schon am Anfang des Kapitels erwähnt, bilden die einzelnen Gelenke (hier Achsen) des Roboterarmes kinematische Bewegungsketten. Innerhalb der Bewegungsketten werden in Schritten die verwendeten Koordinatensysteme um die nächstfolgende Achse rotiert oder/und in eine beliebige Richtung verschoben.

Sind alle Vektoren (Achsen) so aneinandergereiht, kann die Position des TCP (des Endeffektors) bestimmt werden. Diese Umrechnung eines Koordinatensystems ins Andere wird mit Hilfe der homogenen Transformation realisiert. Dabei handelt es sich um eine 4x4 Matrix:

$$\textbf{Transformationsmatrix } \mathbf{T} = \begin{pmatrix} r11 & r12 & r13 & tx \\ r21 & r22 & r23 & ty \\ r31 & r32 & r33 & tz \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (3.3)$$

Die Verschiebung oder Modifikation des Koordinatensystems in die drei Koordinationsrichtungen x, y, z (**translatorische** Verschiebung) erfolgt mit **t**x, **t**y, **t**z, die rotatorische Verschiebung mit den r_{ij}, (nicht verwechseln mit px, py, pz des Endeffektors, den Koordinaten der Orientierungswinkel und der Formel 3.10, es ist nur ein Beispiel für homogene Transformation).

Bei einer rein translatorischen Verschiebung ergibt sich:

$$\textbf{T Trans (x, y, z)} = \begin{pmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (3.4)$$

Bei rotatorischen Verschiebung eines Koordinatensystems um die x, y, x Achsen:

$$\textbf{T Rot (x, } q \textbf{) =} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos q & -\sin q & 0 \\ 0 & \sin q & \cos q & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (3.5)$$

$$\textbf{T Rot (y, } q \textbf{) =} \quad \begin{pmatrix} \cos q & 0 & \sin q & 0 \\ 0 & 1 & 0 & 0 \\ -\sin q & 0 & \cos q & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (3.6)$$

$$\textbf{T Rot (z, } q \textbf{) =} \quad \begin{pmatrix} \cos q & -\sin q & 0 & 0 \\ \sin q & \cos q & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (3.7)$$

Die kinematischen Ketten können nun durch Multiplikation der oben genannten Transformationsmatrizen erfasst werden. Als Ergebnis der Multiplikation steht dann die gesamte Translation Tn (mit n als Anzahl der Gelenke und Multiplikationen) bezogen auf das ursprüngliche Koordinatensystem.
Innerhalb dieser Ketten wird in jedem Schritt das zuletzt verwendete Koordinatensystem (Gelenkkoordinatensystem) um die entsprechende Achse rotiert bzw. verschoben, um wieder auf das Basiskoordinatensystem zurückzukommen.

Für das Zurückrechnen der Orientierungswinkel y (Xe, Gieren), J (ye, Nicken), j (ze, Rollen) (siehe Abb. 3.6) des Greifers ergibt sich nach (3.5 – 3.7) und

mit (3.2) $T = Rot_X(y) * Rot_Y(q) * Rot_Z(j)$ [16]

nach der Multiplikation folgende Matrix :

$$\begin{pmatrix} \cos j \cos q & \cos j \sin q \sin y - \sin j \cos y & \cos j \sin q \cos y + \sin j \sin y & 0 \\ \sin j \cos q & \sin j \sin q \sin y + \cos j \cos y & \sin j \sin q \cos y - \cos j \sin y & 0 \\ \boxed{-\sin q} & \cos q \sin y & \cos q \cos y & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (3.9)$$

Mit den dazugehörigen Koordinatensystemen der einzelnen Orientierungswinkel und der Positionen x, y, z ergibt sich :

$$T(\underline{p}^{soll}) = \begin{pmatrix} x_{ex} & y_{ex} & z_{ex} & px \\ x_{ey} & y_{ey} & z_{ey} & py \\ \boxed{x_{ez}} & y_{ez} & z_{ez} & pz \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (3.10)$$

Durch das Einsetzen der vorgegeben Winkel y , q ,j bzw. x, y, z und Komponentenvergleich können die einzelnen Werte für die Rückwärts - bzw. Vorwärtstransformation zurückgewonnen werden (siehe dazu 3.3.2 und 3.3.3).
Bsp. 1.Spalte, 3.Zeile,
$-\sin q = x_{ez}$ -> $q = -\arcsin x_{ez}$ usw.

3.3.1 Denavit – Hartenberg – Methode

Eine Vereinfachung bei der Umrechnung eines Koordinatensystems ins Andere mit Hilfe der homogenen Transformation bietet die Denavit – Hartenberg – Methode. Die Denavit – Hartenberg – Methode beschreibt die relative Lage zwei benachbarten Gelenke als eine allgemeine Transformationsmatrix [5].
Diese universelle Transformationsmatrix muss je nach Aufbau des Gelenkes n mit Hilfe der Variabel $\mathbf{a_n}$, $\mathbf{l_n}$, und $\mathbf{?_n}$ modifiziert und konfiguriert werden und dann als die zu dem entsprechenden Gelenk n Transformationsmatrix A_n erstellt werden. Danach werden alle Transformationsmatrizen miteinander multipliziert, um die gesuchte $T_n(\partial)$ zu bekommen.
Zu erst werden alle Gelenke in aufsteigender Folge durchnummeriert. Es sind keine geschlossenen kinematischen Ketten erlaubt. Bei der Orientierung der Koordinatensysteme muss noch folgendes beachtet werden:

- die z_{n-1} – Achse liegt parallel zum nachfolgendem Glied des Roboterarmes (z.B. in der Abbildung 3.8 $\mathbf{z_{n-1}}$ zu $\mathbf{z_n}$)
- die x_n – Achse steht senkrecht zur z_{n-1} und zeigt von ihr weg [5].

Die Länge $\mathbf{a_n}$ des Gliedes wird als Länge der Linie definiert, die jeweils auf den Achsen zweier benachbarter Gelenke senkrecht steht (Abbildung 3.8).

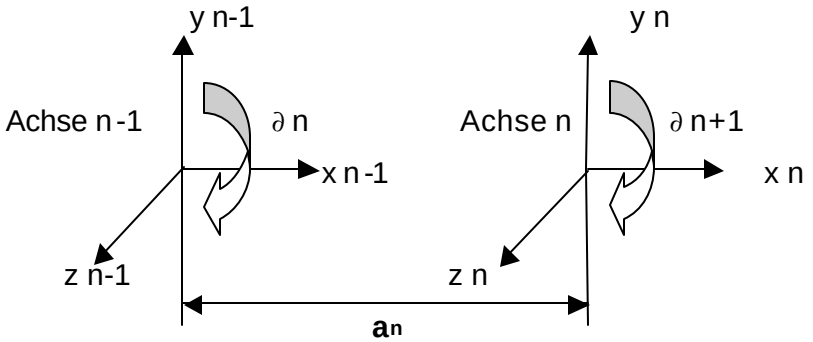


Abbildung 3.8 : Senkrechte Verschiebung zwischen benachbarten Gelenken

Die Entfernung l_n, ist die Entfernung zwischen den Lotrechten der benachbarten Gliedern (Abbildung 3.9). Diese Größe wird auch als Offset des Gelenks bezeichnet. Im Fall eines linearen Gelenks handelt es sich um die Variable des Gelenks.

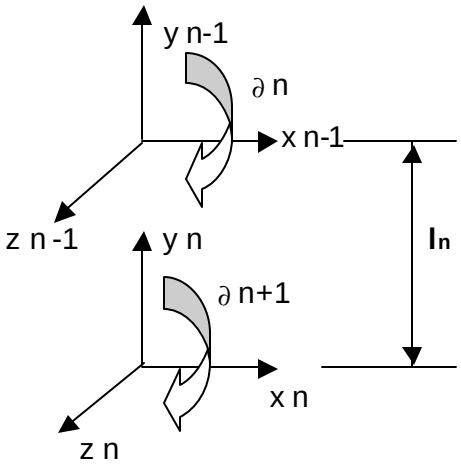


Abbildung 3.9 : Lotrechte Verschiebung zwischen zwei benachbarten Gelenken

Der Winkel $?_n$ besteht aus der Verdrillung zwischen den Achsen der Gelenke in einer zu a_n senkrechten Ebene (Beispiel Abb. 3.12 und 3.13).
Der Winkel ∂_n ist der Winkel zwischen den Gliedern, der als Winkel zwischen den Lotrechten a_n und a_{n-1} in der Ebene senkrecht zu den Achsen des Gelenks gemessen wird [3].
Eine allgemeine Transformation kann hergeleitet werden durch:

- eine Drehung um den Winkel ∂_n um die Achse z_{n-1},
  damit x_{n-1} parallel zu x_n wird
- eine Verschiebung um x_n die Entfernung l_n entlang z_{n-1}
- eine Verschiebung entlang x_n um eine Länge an und
- einer Drehung um x_n um den Torsionswinkel ?,
  damit die z_{n-1}-Achse mit z_n – Achse übereinstimmt [3].

Somit ergibt sich für jedes Gelenk durch die Multiplikation
Rot (z_1, ∂_2) * **Trans**(0, 0, l_2) * **Trans**(a_2, 0, 0) **Rot**(x_2, $?_2$)
und Benutzung der Formeln 3.4 bis 3.7 die **Denavit – Hartenberg – Matrix** :

$$\text{D – H – Matrix} = \begin{matrix} \cos\partial & -\sin\partial\cos l & \sin\partial\sin l & a\cos\partial \\ \sin\partial & \cos\partial\cos l & -\cos\partial\sin l & a\sin\partial \\ 0 & \sin l & \cos l & l \\ 0 & 0 & 0 & 1 \end{matrix} \qquad (3.11)$$

Als Beispiel für die Anwendung der Denavit – Hartenberg – Methode wurden die Transformationsmatrizen für den ROB3 aufgestellt.
Der Verfasser geht von der Grundstellung des Roboters wie in der Abb.3.5 aus.

Bei dem Verfahren ist es wichtig zuerst alle Parameter für die Gelenke zu definieren. Nach Denavit – Hartenberg – Methode ergibt sich für ROB3 folgende Tabelle:

Gelenk (Achse)	∂_n	l	a_n	l_n
1	∂_1	90°	0	0
2	∂_2	0°	$a_{2=200}$	0
3	∂_3	0°	$a_{3=130}$	0
4	∂_4	0°	0	0
5	∂_5	-90°	0	0
Greifer	0°	0°	0	$l_{5=130}$

Tabelle 3.2 : Variabel für die Denavit – Hartenberg - Matrix

Die sechste Zeile in der Tabelle 3.1 ist reine translatorische Verschiebung(nur Offset) durch den Greifer. Die aufeinander folgenden Gelenkachsen sind zueinander parallel angeordnet. In der Abbildung 3.11 sind die einzelnen Koordinatensysteme der entsprechenden Achsen gezeichnet, wobei das erste Koordinatensystem (Achse 1) mit dem Index 0 beginnt. Bei der räumlichen Darstellung der Weltkoordinatensysteme wird die „Rechte – Hand - Regel“ verwendet (Abb. 3.10).

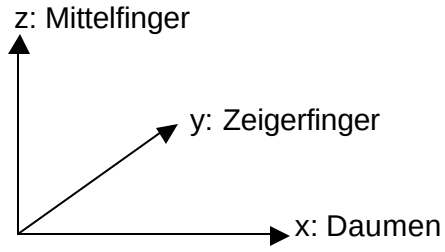


Abbildung 3.10 : „Rechte – Hand - Regel“

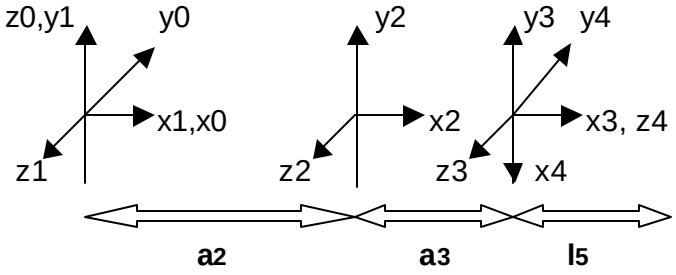


Abbildung 3.11 : Koordinatensystem der einzelnen Achsen

Der Winkel l beträgt bei dem ersten Gelenk + 90°.
Durch diese Drehung um die x1 - Achse werden die Drehachsen (z – Achsen) der beiden Gelenke zur Deckung gebracht (Abb.3.12 und 3.13).

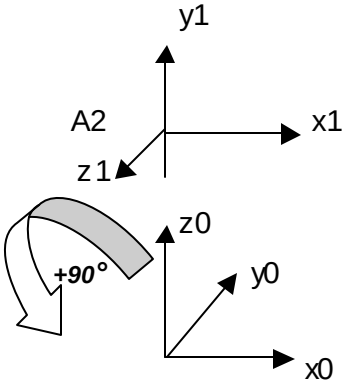


Abbildung 3.12 : Drehung um die x – Achse bei erstem Gelenk

Nach dem die beiden benachbarten x- Achsen in Deckung gebracht worden sind, ergibt sich beim letzten Drehgelenk A5 ein Winkel l von - 90° (Abb.3.13)

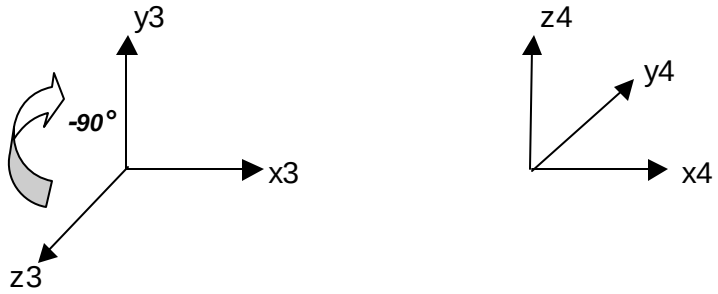


Abbildung 3.13 : Drehung um die x – Achse bei letztem Gelenk

Mit dieser Methode ergeben sich für den ROB 3 mit seinen 5 Achsen und dem Greifer (siehe Abbildung 3.5) nach Tabelle 3.2 folgende Matrizen:

$$A_1 = \begin{pmatrix} \cos\partial_1 & 0 & \sin\partial_1 & 0 \\ \sin\partial_1 & 0 & -\cos\partial_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (3.12)$$

$$A_2 = \begin{pmatrix} \cos\partial_2 & -\sin\partial_2 & 0 & a_2\cos\partial_2 \\ \sin\partial_2 & \cos\partial_2 & 0 & a_2\sin\partial_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (3.13)$$

mit a_2=200 mm

$$A_3 = \begin{pmatrix} \cos\partial_3 & -\sin\partial_3 & 0 & a_3\cos\partial_3 \\ \sin\partial_3 & \cos\partial_3 & 0 & a_3\sin\partial_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (3.14)$$

mit a3=130 mm

$$A_4 = \begin{pmatrix} \cos\partial_4 & -\sin\partial_4 & 0 & 0 \\ \sin\partial_4 & \cos\partial_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (3.15)$$

$$A_5 = \begin{pmatrix} \cos\partial_5 & 0 & -\sin\partial_5 & 0 \\ \sin\partial_5 & 0 & \cos\partial_5 & 0 \\ 0 & -1 & 0 & l_5 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (3.16)$$

Greifer => Trans (x) = $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l_5 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ (3.17) reine transl. Verschiebung nach (3.4)

mit l_5 = 130mm

Die Transformationsmatrix lautet somit:

$$T^6 = \prod_{i=1}^{6} Ai = (A1*A2*A3*A4*A5*A6) \qquad (3.18)$$

Durch die Multiplikation erhält man eine 4x4 Matrix aus der man durch einen Komponentenvergleich (3.9) und (3.10) direkt die Position (x, y, z) des TCP bestimmen kann (**p = f (**∂**)**). Die Orientierungswinkel y , q , j werden ebenfalls aus (3.10) und (3.9) gewonnen.

3.3.2 Rückwärtstransformation (inverse Transformation)

Mit Hilfe der Rückwärtstransformation, die auch inverse Transformation genannt wird, können die Gelenkstellungen bei gegebenen Position und Ausrichtung des Endeffektors gewonnen werden. Diese Achsenwerte (Winkel) werden direkt an die Steuerung weitergeben.

Rückwärtstransformation oder **inverse Transformation**

Weltkoordinatensystem	⇒	Achsenkoordinatensystem
Positionsvorgabe		**Gelenkstellungen**

Die Lösungsgleichungen für die Rückwärtstransformation der Koordinaten wurden nach einer Modifikation des im **[5]** als Beispiel benutzten HDS 06 (NOKIA)

Manipulators aufgestellt. Die hat zum Vorteil, dass die Lösungsgleichungen bereits vorhanden sind.

HDS 06 [5]	ROB3
q1	$\partial 1$
q2	$\partial 2$
q3	$\partial 3 + 90°$
q4	0°
q5	$\partial 4$
q6	$\partial 5$

Tabelle 3.3 Vergleich der Gelenke zwischen HDS 06 und ROB3

Im [5] ergeben sich für die 6 Winkel des HDS 06 folgende Gleichungen:

$$\partial 1 = \arctan 2\left[\frac{y}{x}\right] \tag{3.19}$$

$$\partial 3 = \arctan 2\left[\frac{\left((c1\times x)+(s1\times y)\right)^2 + z^2 - l4^2 - a2^2}{\pm\sqrt{\left(4a2^2 l4^2 - \left[\left((c1\times x)+(s1\times y)\right)^2 + z^2 - l4^2 - a2^2\right]^2\right)}}\right] \tag{3.20}$$

$$\partial 2 = \arctan 2\left[\frac{\left(a2+(l4\times s3)\right)\times z+\left((c1\times x)+(s1\times y)\right)\times(l4\times c3)}{\left((c1\times x)+(s1\times y)\right)\times\left(a2\times(l4\times s3)\right)-(z\times l4\times c3)}\right] \tag{3.21}$$

$$\partial 4 = \arctan 2\left[\frac{s1z_{ex} - c1z_{ey}}{c32\left(c1z_{ex}+s1z_{ey}\right)+s32z_{ez}}\right] \tag{3.22}$$

$$\partial 5 = \arctan 2\left[\frac{c4\left[c32\left(c1z_{ex}+s1z_{ey}\right)+s32z_{ez}\right]+s4\left(s1z_{ex}-c1z_{ey}\right)}{s32\left(c1z_{ex}+s1z_{ey}\right)-c32z_{ex}}\right] \qquad (3.23)$$

$$\partial 6 = \arctan 2\left[\frac{-c5\left\{c4\left[c32\left(c1y_{ex}+s1y_{ey}\right)+s32y_{ez}\right]+s4\left(s1y_{ex}-c1y_{ey}\right)\right\}+s5\left[s32\left(c1y_{ex}+s1y_{ey}\right)-c32y_{ez}\right]}{-s4\left[c32\left(c1y_{ex}+s1y_{ey}\right)+s32y_{ez}\right]+c4\left(s1y_{ex}-c1y_{ey}\right)}\right]$$

(3.24)

Mit:

s = sin – Funktion + Winkelnummer, die mit der Achsennummer übereinstimmt,

c= cos – Funktion , z.B. und **c = 32** = cos ($\partial 3$ + $\partial 2$)

Die Funktion **arctan2** berechnet den Arcustangens aus den Parametern x und y im Bereich $-p$; p . Dieser entspricht weitgehend dem Arcustangens aus x / y, allerdings kann hier das Vorzeichen beider Parameter ausgewertet und so der Quadrant des Ergebnisses bestimmt werden.

Die kartesischen Koordinaten der einzelnen Orientierungswinkel ergeben sich wieder durch das Gleichsetzen der beiden Formeln (3.9) und (3.10) (Orientierungswinkel gegeben).

z.B. $X_{ex} = \cos j \cos q$

Nach der Modifikation mit Tabelle 3.3 und

l4 (HDS 06) = a3 (ROB3) ergeben sich folgende Lösungsgleichungen für den ROB 3:

$$\partial 1 = \arctan 2\left[\frac{y}{x}\right] \qquad (3.25)$$

$$\partial 3 = \arctan 2\left[\frac{\left((c1\times x)+(s1\times y)\right)^2+z^2-a3^2-a2^2}{\pm\sqrt{\left(4a2^2a3^2-\left[\left((c1\times x)+(s1\times y)\right)^2+z^2-a3^2-a2^2\right]^2\right)}}\right] \qquad (3.26)$$

$$\partial 2 = \arctan 2\left[\frac{\left(a2+\left(a3\times s3\right)\right)\times z+\left(\left(c1\times x\right)+\left(s1\times y\right)\right)\times\left(a3\times c3\right)}{\left(\left(c1\times x\right)+\left(s1\times y\right)\right)\times\left(a2\times\left(a3\times s3\right)\right)-\left(z\times a3\times c3\right)}\right] \tag{3.27}$$

$$0 = \arctan 2\left[\frac{s1z_{ex}-c1z_{ey}}{c32\left(c1z_{ex}+s1z_{ey}\right)+s32z_{ez}}\right] \tag{3.28}$$

$$\partial 4 = \arctan 2\left[\frac{\left[c32\left(c1z_{ex}+s1z_{ey}\right)+s32z_{ez}\right]}{s32\left(c1z_{ex}+s1z_{ey}\right)-c32z_{ex}}\right] \tag{3.29}$$

$$\partial 5 = \arctan 2\left[\frac{-c4\left\{\left[c32\left(c1y_{ex}+s1y_{ey}\right)+s32y_{ez}\right]\right\}+s4\left[s32\left(c1y_{ex}+s1y_{ey}\right)-c32y_{ez}\right]}{\left(s1y_{ex}-c1y_{ey}\right)}\right] \tag{3.30}$$

Wobei der Winkel $\partial 3 = \partial 3 + 90°$ beträgt (Tabelle 3.3).

3.3.3 Vorwärtstransformation (direkte Transformation)

Umgekehrt der Rückwärtstransformation ist die Vorwärtstransformation. Hier werden die Koordinaten des Endeffektors in Weltkoordinaten aus gegebenen Gelenkstellungen bestimmt.

Vorwärtstransformation

Achsenkoordinatensystem ⟹ Weltkoordinatensystem

Nach [5] ergeben sich folgende Gleichungen für den HDS 06:

$$x_{ex} = c6[c5\left(c4c32c1+s4s1\right)-s5s32c1]+s6\left[c4s1-s4c32c1\right] \tag{3.31}$$

$$x_{ey} = c6[c5\left(c4c32s1-s4c1\right)-s5s32s1]-s6\left[c4c1+s4c32s1\right] \tag{3.32}$$

$$x_{ez} = c6[c5c4s32+s5c32]-c6s4s32 \tag{3.33}$$

$$y_{ex} = -s6[\,c5\,(c4c32\,c1 + s4s1) - s5s32c1] + c6\,[c\,4s1 - s\,4c32c1] \tag{3.34}$$

$$y_{ey} = -s6[\,c5(c4c32\,s1 - s4c1) - s5s32s1] - c6\,[c4c1 - s4c32s1] \tag{3.35}$$

$$y_{ez} = -s6[c5c4s\,32 + s5c32] - c6s4s32 \tag{3.36}$$

$$z_{ex} = s5\,(c4c32\,c1 + s\,4s1) + c5s32c1 \tag{3.37}$$

$$z_{ey} = s5(c\,4c32\,s1 - s4c1) + c5s32s1 \tag{3.38}$$

$$z_{ez} = s5c4s32 - c5c32 \tag{3.39}$$

$$\begin{aligned} x &= l4s32\,c1 + a\,2c2\,c1 \\ y &= l4s32\,s1 + a2c2\,s1 \\ z &= -l4c32 + a2s\,2 \end{aligned} \tag{3.40}$$

Mit:

s = sin – Funktion + Winkelnummer, die mit der Achsennummer übereinstimmt,

c= cos – Funktion , z.B. und **c = 32** = cos (∂_3 + ∂_2)

Für den ROB3 ergeben sich mit Tabelle 3.3:

$$x_{ex} = c5[c\,4c32c1 - s4s32c1] + s5s1 \tag{3.41}$$

$$x_{ey} = c5[c\,4c32c1 - s4s32c1] - s5s1 \tag{3.42}$$

$$x_{ez} = c5[c\,4s32 + s4c32] \tag{3.43}$$

$$y_{ex} = -s\,5[c\,4c32c1 - s4s32c1] + c5s1 \tag{3.44}$$

$$y_{ey} = -s5[c\,4c32\,c1 - s4s32c1] - c5s1 \tag{3.45}$$

$$y_{ez} = -s5[c\,4s32 + s4c32] \tag{3.46}$$

$$z_{ex} = s\,4c32\,s1 + c4s\,32s1 \tag{3.47}$$

$$z_{ey} = s4c32\,s1 + c4s32s1 \tag{3.48}$$

$$z_{ez} = s\,4s32 - c\,4c32 \tag{3.49}$$

Berechnung der Nick- , Gier- und -Rollwinkel mit (3.9) und (3.10) :

q (Rot_y) = arcsin (x_e_z) (3.50)

j (Rot_z) = arcsin ($x\_e_y/\cos q$) (3.51)

y (Rot_x) = arcsin ($y\_e_z/\cos q$) (3.52)

$$x = a3s32c1 + a2c2c1 \quad (3.53)$$

$$y = a3s3\,2s1 + a2c2\,s1 \quad (3.54)$$

$$z = -a3c32 + a2s2 \quad (3.55)$$

4 Programmierungsarten eines Manipulators

Die Programmierung eines Roboters kann je nach Anwendung auf verschiedene Art geschehen. Es gibt vier verschiedene Programmierverfahren:

- Teach-in-Programmierung
- Play-back-Programmierung
- Off-Line-Programmierung und
- Sensorunterstützte Programmierung

4.1 Teach-in-Programmierung

Diese Programmierungsart wird bei Programmierung von Roboter am meisten angewendet. Die Programmierung erfolgt mit Hilfe eines Programmierhandgerätes auch **Teachbox** genannt vor Ort unter Sichtkontrolle des Programmierers.
Im Zeitalter des Laptops kann die Stellung mit Hilfe eines PCs und dafür entwickelter Software erreicht und übernommen werden. Die gewünschte Robotersteuerung wird nach jedem Schritt in den Programmspeicher übernommen. Da der Zeitaufwand sehr groß ist, wird das Verfahren überwiegend in Serienfertigung angewendet.

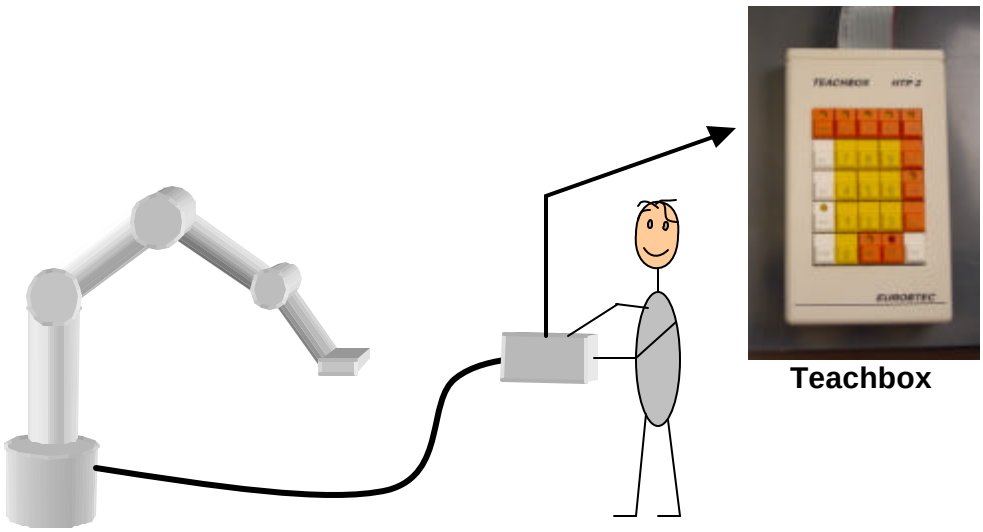


Abbildung 4.1 : Teach-in-Programmierung mit Teachbox des ROB 3

Der Trainingsroboter ROB3 lässt sich mit Hilfe der mitgelieferten Teachbox (Abb.4.1) ohne Computer programmieren.

4.2 Play-Back-Programmierung

Ähnlich wie Teach-in-Programmierung erfolgt bei dieser Programmierungsart die Programmierung vor Ort unter Sichtkontrolle des Programmierers. Bei Play-Back-Programmierung wird der Roboter bei abgeschalteten Antrieben mit der Hand an die gewünschte Position geführt. Die Steuerung speichert die Bewegung und führt sie dann im Betrieb. Die Play-Back-Programmierung wird fast nur beim Lackieren kleiner Teile eingesetzt.

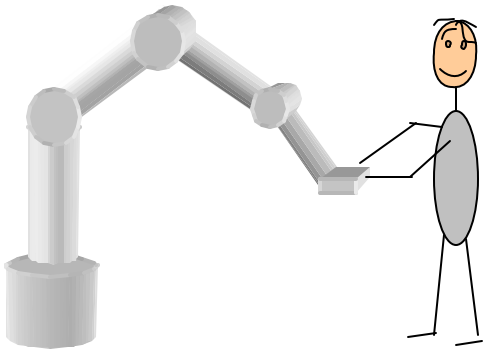


Abbildung 4.2 : Play-Back-Programmierung

4.3 Off-Line-Programmierung

Die Off-Line-Programmierung der Roboterbewegung erfolgt fern vom Roboter an einem Computer. Bei komplexeren Arbeitsaufgaben wie z. B. Karosseriebau, müssen off-line erstellte Programme durch die prozessnahe Teach-in nachprogrammiert werden.

4.4 Sensorunterstützte Programmierung

Wie der Name schon sagt, erfolgt diese Programmierung mit Hilfe der Sensoren. Mit Videokameras und Berührungssensoren werden, meist in Verbindung mit der Off-Line Programmierung, die Roboterbewegungen erzeugt. Der Roboter bewegt sich z.B. entlang am Boden befestigten Magnetstreifen oder geklebten Farbstreifen, die eine am Roboter befestigte Kamera analysiert. Die Flexibilität der Bewegungen ist bei dieser Methode sehr gering. Der Roboter kann nur entlang der Streifen fahren. Dazu kommt ein hoher Aufwand bei den Änderungen der Wege (neue Streifen verlegen, alte beseitigen).

5 Programmieren des Roboters ROB 3

Der Roboter ROB 3 der Firma Eurobtec lässt sich über das TBPS (Teach Box Programmier System, lauffähig nur unter Windows 98), die Teachbox (Handprogrammiergerät) oder PSI (Programmier System für Industrieroboter, nur optional) direkt programmieren.

Über das nachträglich mitgelieferte Low – Level – Protokoll kann eine zusätzliche Schnittstelle für eine Programmierumgebung für den Roboter selbst erstellt werden. Als die neue Programmierumgebung wird die Software - SPS ProSys der Firma Deltalogic benutzt.

Die oben genannten Programmiermöglichkeiten sowie das Low – Level-Protokoll benutzen als Schnittstelle die serielle Schnittstelle RS 232.

Die Programmierumgebung, die durch den Verfasser in der Programmiersprache Visual Basic 6.0 erstellt wurde, erlaubt als Betriebssystem Windows NT und hat als Verbindung zur SPS die DDE -Schnittstelle.

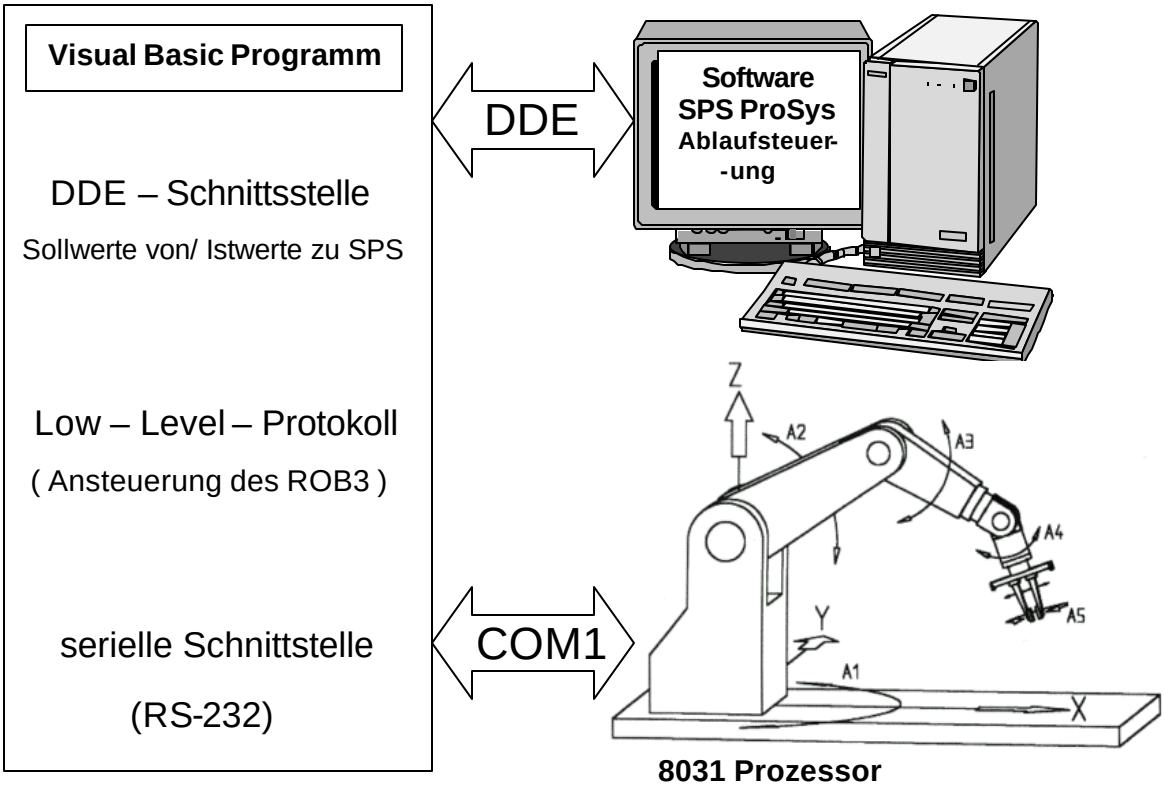


Abbildung 5.1. Prinzip der Programmierung und Ansteuerung des ROB3

5.1 Low – Level – Protokoll

Das Low – Level – Protokoll der Firma Eurobtec ist ein nicht standardisiertes Protokoll zur Nachrichtenübertragung und dient der Ansteuerung des ROB 3 über die serielle Schnittstelle RS 232. Der Antriebscontroller (8031 Prozessor) empfängt die Kommandowörter und steuert die Servomotoren der Achsen und des Greifers.
Die Kommunikation arbeitet ohne Hardware Handshake, aber bei den meisten (nicht allen) Kommandos gibt der Roboter eine Rückmeldung an die Steuereinheit ab. Bei den Rückmeldungen schickt der Roboter immer das Kommandobyte mit.

5.1.1 Kommunikationsablauf

Die Schnittstelle des Roboters stellt nach vorangegangenem RESET am Roboter (Rückseite des Sockels) die Baudrate automatisch ein.

Als aller erstes Byte muss von der Steuereinheit 20 Hex (SPACE) an den Roboter übertragen werden. Als Rückantwort für einen fehlerfreien Start der Kommunikation schickt der Roboter je nach Initialisierung 15, F1, F2, $F3_{Hex}$ an die Steuereinheit zurück. Andere Antworten des Roboters bedeuten fehlerhafte Kommunikation. Nach einmaligem SPACE kommen die eigentlichen Kommandodatensätze, die immer mit 03_{Hex} (ETX) abgeschlossen werden müssen.

Funktion	Kommando	Bemerkung
Kommunikation starten	20 Hex (SPACE)	einmalige Initialisierung am Anfang

↓

Antwort des Roboters auf Anfang der fehlerfreien Kommunikation
-> 15 Hex Initialisierung
-> F1, F2, F3 Hex Mehrfachinitialisierung des Roboters

↓

Kommandodatensatz	siehe Kapitel 6.1.2	eigentliche Kommandos
Kommandodatensatz abschließen	03 Hex (ETX)	immer am Ende jedes Kommandos

Abbildung 5.2 : Übersicht über den Kommunikationsablauf

5.1.2 Kommandos zur Ansteuerung des Roboters

Die eigentlichen Kommandos werden für das bessere Verständnis in binärer Form byteweise dargestellt und dann in die Hexform umgewandelt. Das Kommandoschlüsselwort, das auch als Rückmeldung bei den Positionierkommandos zurückgeliefert wird, ist das erste Kommandobyte ohne den Sollwert bzw. Zeitfaktor.

Positionier – Kommandos:

eine Achse	| 0 | 0 | 0 | 0 | R | a | a | a |	00 – 0D Hex

+ 1 Byte Sollwert (0-255)

mit R = 1, Rückmeldung nach dem Erreichen der Sollposition

mit a = Achsennummer (0 – 5, 5 entspricht dem Greifer)

alle Achsen	| 0 | 0 | 0 | 0 | R | 1 | 1 | 1 |	0F Hex

+ 7 Byte Sollwert, (0-255)

Die Rückmeldung des Roboters ist das Kommandoschlüsselwort und ETX.

Positionier – Kommandos mit Geschwindigkeit:

eine Achse	| 0 | 1 | 1 | 1 | R | a | a | a |	70 – 7D Hex

+ 1 Byte Sollwert (0-255)

+ 1 Byte Zeitfaktor T, sinnvolle Werte: 0 für max. Geschwindigkeit bis 7 = sehr langsam (bedeutet eine weitere Änderung der Achsenposition nach 70msec)

mit R = 1, Rückmeldung nach dem Erreichen der Sollposition

mit a = Achsennummer (0 – 5, 5 entspricht dem Greifer)

Die Rückmeldung des Roboters ist das Kommandoschlüsselwort und ETX.

alle Achsen	| 0 | 1 | 1 | 1 | R | 1 | 1 | 1 |	7F Hex

+ 6 Byte Sollwerte (0-255)

+ 6 Byte Zeitfaktoren T

Mit der Übernahme des Kommandos wird die vorhergehende Sollposition jeweils nach Ablauf der Zeit T um einen Schritt erhöht bzw. erniedrigt bis die Endposition erreicht ist. Die Zeit T errechnet sich aus Zeitfaktor *10 msec.
Bei Zeitfaktor 0 wird der Roboter mit seiner maximalen Geschwindigkeit angefahren.

Regel – Kontroll – Kommandos:

60 Hex = Motor – Regelung ausschalten
61 Hex = Motor – Regelung einschalten
62 Hex = Positionier – Abschaltung (Software Not – Aus)
momentane Ist –Position bleibt erhalten

Seriennummer – Abfrage:

63 Hex = Seriennummer Abfrage

Die Rückmeldung des Roboters ist das Kommandoschlüsselwort, S0, S1; S2 und ETX.

Positions-Abfrage-Kommandos

eine Achse	| 0 | 1 | 0 | 0 | 0 | a | a | a |	40 – 45 Hex

mit a = Achsennummer (0 – 5)
Rückmeldung:
- Kommando, 1 Byte Istwert (0 -255), ETX

alle Achsen	| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |	4F Hex

Rückmeldung:
- Kommando, 7 Byte Istwert (0 -255), ETX

Kommandos für die Ansteuerung der I/O Testbox:

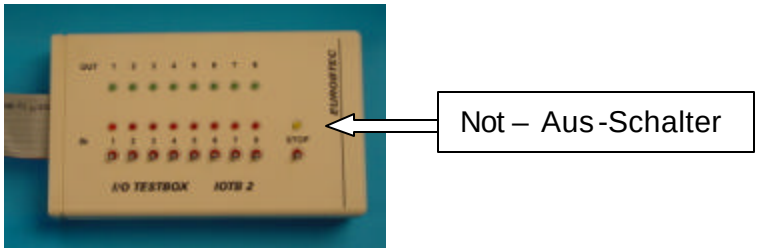


Abbildung 5.3 : Die I/O Testbox der Firme EUROBTEC

Die I/O Testbox besitzt 8 digitale Ein- und Ausgänge und wird direkt an dem Roboter angeschlossen.
Zusätzlich gibt es an der Testbox einen Schalter, der eine Not – Aus –Funktion hat.

Abfrage der digitalen Eingänge (Schalter) :

digitale Eingänge	| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |	54 Hex

Rückmeldung:
- Kommando, 1 Byte Wert der digitalen Eingänge, ETX

Ansteuerung der digitalen Ausgänge (LED's) :

digitale Ausgänge	| 0 | 0 | 0 | 1 | 0 | 0 | u | o |	10-13 Hex

mit u und o:

u	o	Funktion
0	0	Maske direkt ausgeben
0	1	Maske odern (setzen)
1	0	Maske unden(löschen)
1	1	Maske odern(invertieren)

digitale Ausgänge	| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |	10-13 Hex

Beispiel: Ansteuerung Achse 1 mit Geschwindigkeit, Position 50 mit Rückmeldung

	binär	dezimal	hexadezimal
->	0 1 1 1 1 0 0 1	9	A -> Kommandoschlüsselwort
->	0 0 1 1 0 0 1 0	50	32 -> Position
->	0 0 0 0 0 0 0 1	1	1 -> Geschwindigkeit
->	0 0 0 0 0 0 1 1	3	3 -> ETX, Abschluss des Kommandos

Die Rückmeldung nach dem Erreichen der Position ist das Kommandoschlüsselwort und ETX.

6 SPS – Programmierung mit ProSys

Die eigentliche Vergabe der Positionen und Geschwindigkeiten des Roboterarmes geschieht über die Software – SPS. Die Firma Deltalogic bietet mit ProSys eine Entwicklungsumgebung, die sowohl als reine Programmierung für S5 und S7- Steuerungen, als auch Software – SPS (Simulation der Hardware) benutzt werden kann. Als Verbindung zu einer Windows – Applikation bietet ProSys die DDE – Schnittstelle (siehe dazu Kapitel 7.4.2)

Neben aus der SPS – Technik bekannten AWL/FUP/KOP-Editoren gibt es noch zwei zusätzliche Editoren:

- den **Strukturierten Text** (**ST**) - eignet sich besonders gut für Programmieren von mathematischen Formeln, Schleifen und bedingten Verzweigungen.
- eine **Ablaufsteuerung** (**AS**) - dient zum Erstellen von Ablaufketten, bei denen die Schritte sequentiell abgearbeitet werden.

Die Vergabe der nacheinender folgenden Achsen- und Greiferpositionen wurde in der Ablaufsteuerung programmiert. Zwei in Strukturiertem Text erstellte Funktionsbausteine berechnen die dazu erforderlichen Vor- bzw. Rücktransformationen.

6.1 Ablaufssprachen(AS) – Editor

In der Ablaufsteuerung werden in einzelnen Schritten die Instanzen der Funktionsbausteine aufgerufen, die Vor- und Rückwärtstransformation ausführen.

Der Ablaufsprachen – Editor (AS) besteht aus zwei Elementen (Abbildung 6.1):

- **Transitionen** und
- **Aktionen**.

Erst die Erfüllung der Transition (hier Schalter1 betätigt) führt zur Ausführung einer Aktion (Step1). In einer Aktion kann eine Reihe von Anweisungen in allen Programmiermethoden eingeführt werden. Auch eine Ablaufsteuerung, die dann als eine innen verschachtelte Ablaufsteuerung wirkt, kann als eine Aktion der Ablaufsteuerung eingeführt werden.

Nach den letzten Transition (hier Schalter 3) wird wieder die Ablaufkette ab Aktion „Step 1“ abgearbeitet.

Die erste Aktion ist in diesem Fall ein Schritt, der nur beim Start der Ablaufkette abgearbeitet wird, also eine Initialisierung.

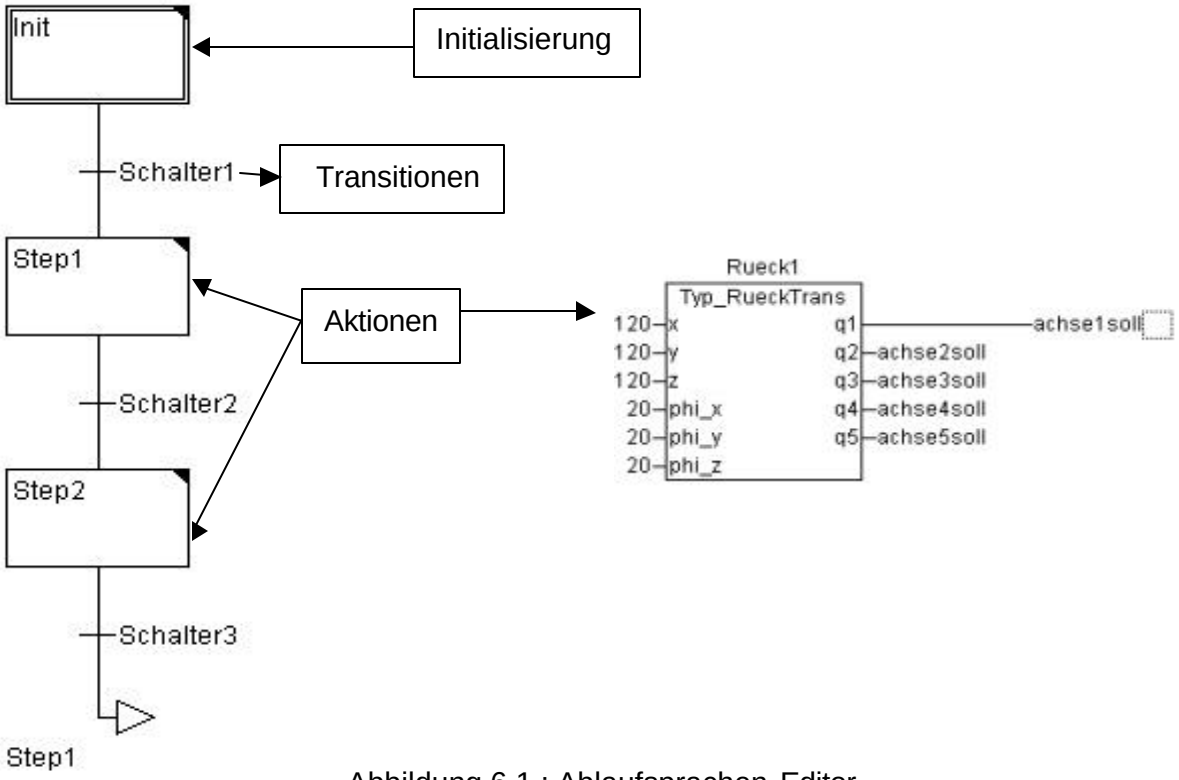


Abbildung 6.1 : Ablaufsprachen-Editor

6.2 Strukturierter Text (ST) - Editor

Die komplexe mathematische Funktionen (Vor- und Rückwärtstransformation), die, die beiden Funktionsbausteine ausführen (Kapitel 3.3.2), können in der SPS – Programmiertechnik nur im Strukturierten Text programmiert werden.
In keinem anderen Editor können z.B. trigonometrische Funktionen in ein SPS - Programm eingegeben werden.
ProSys rechnet die Winkelfunktionen im Bogenmaß. Die Achsen des Roboters werden aber in Grad programmiert. Deshalb müssen die ausgerechneten Achsenwerte im ST noch in Grad umgerechnet werden:

```
q1:= ATAN2(y/x);
q1:=q1*57.29578; (* RAD -> GRAD Umrechnung in ProSys - AS*)
```

Der Funktionsbaustein „Rueck1“ (Abbildung 6.1) sieht nach dem Aufruf in ST folgendermaßen aus:

1. Aufruf des Funktionsbausteins Typ „Rueck_Trans“

```
Rueck1(x:=20 , y:=400 , z:=20 , phi_x:=20 , phi_y:=20 , phi_z:=20 );
```

2. Zugriff auf die Ausgangsvariabel des Funktionsbausteins

```
achse1soll:=REAL_TO_INT( Rueck1.q1);
achse2soll:=REAL_TO_INT( Rueck1.q2);
achse3soll:=REAL_TO_INT( Rueck1.q3);
achse4soll:=REAL_TO_INT( Rueck1.q4);
achse5soll:=REAL_TO_INT( Rueck1.q5);
```

Erklärung :
Die Berechnung im Baustein geschieht mit Variabel von Typ REAL, also Gleitkommazahlen. Die Werte(achse1-5soll), die an die Steuerung weitergegeben werden, sind als Integerzahlen deklariert. Deswegen muss eine Typwandlung von Gleitkomma- in Ganzzahlen stattfinden (REAL_TO_INT).

Die für die Rückwärtstransformation benötigte ATAN2 (Kapitel 3.3.2, Seite 26) wird ebenfalls in ST eingegeben:

```
(*ATAN2 Berechnung*)
IF (x=0) THEN

 IF y<0 THEN
 q1:=-180/2 ;
ELSE
 q1:=180/2;
 END_IF;

ELSE

 IF x<0 THEN
```

```
 IF y<0 THEN
q1:= (ATAN(y/x)*57.29578) -180;
ELSE
q1:=(ATAN(y/x)*57.29578)+180;
END_IF;

ELSE
q1:=ATAN(y/x)*57.29578;
END_IF;

END_IF;
```

7 Visual Basic (VB)

7.1 Allgemein

Was ist Visual Basic?

Visual Basic (VB) ist ein flexibles, leistungsfähiges und erweiterbares Entwicklungssystem für Windows 3.x, 95 und NT. VB ist sowohl für professionelle Softwareentwicklung, wie Datenbanken und Visualisierungen als auch für Hobbyprogrammierung geeignet. Mit Visual Basic (VB) lassen sich in kürzester Zeit Windows - Applikationen erstellen.

Für die Gestaltung der Bedienoberfläche (der Visualisierung) ist keine Programmierung erforderlich. Das Zauberwort in VB - Programmierung heißt Werkzeugsammlung mit Steuerelementen.

Steuerelemente dienen meistens der Gestaltung von Bedienoberfläche und Durchführung der Ein- und Ausgaben. Sie werden aus der schon vorhandenen Werkzeugsammlung mit Hilfe der Maus auf das gewünschte Formular gezogen und entsprechend konfiguriert. Erst an dieser Stelle kommt das eigentliche Programmieren.

VB ist im Vergleich zur C und C++ eine Ereignisgesteuerte Programmiersprache d. h. es gibt keinen eindeutigen, voraussagbaren Programmverlauf. Ein weiterer Unterschied zu C und C++ ist, dass in VB keine Zeiger gibt., d.h. es können keine Adressen für die Variable vergeben werden.

7.2 Die Entwicklungsumgebung (IDE)

Nach dem Öffnen eines neuen Projektes erscheint dem Programmierer die integrierte Entwicklungsumgebung (**IDE I**ntegreted **D**evelopment **E**nvironment) von VB (Abb.7.1).

Die IDE besteht beim Starten von neuem Projekt grundsätzlich aus der von vielen Windowsprogrammen bekannten Menüleiste, Werkzeugsammlung (Toolbox) mit Steuerelementen, dem Eigenschaftsfenster, Projektexplorer und dem Formularfenster.

Man kann zusätzliche Fenster aktivieren wie z. B. ein Überwachungsfenster (Debugger), die oben genannten Fenster geben jedoch die wichtigsten Informationen bei der Erstellung eines VB– Programms.

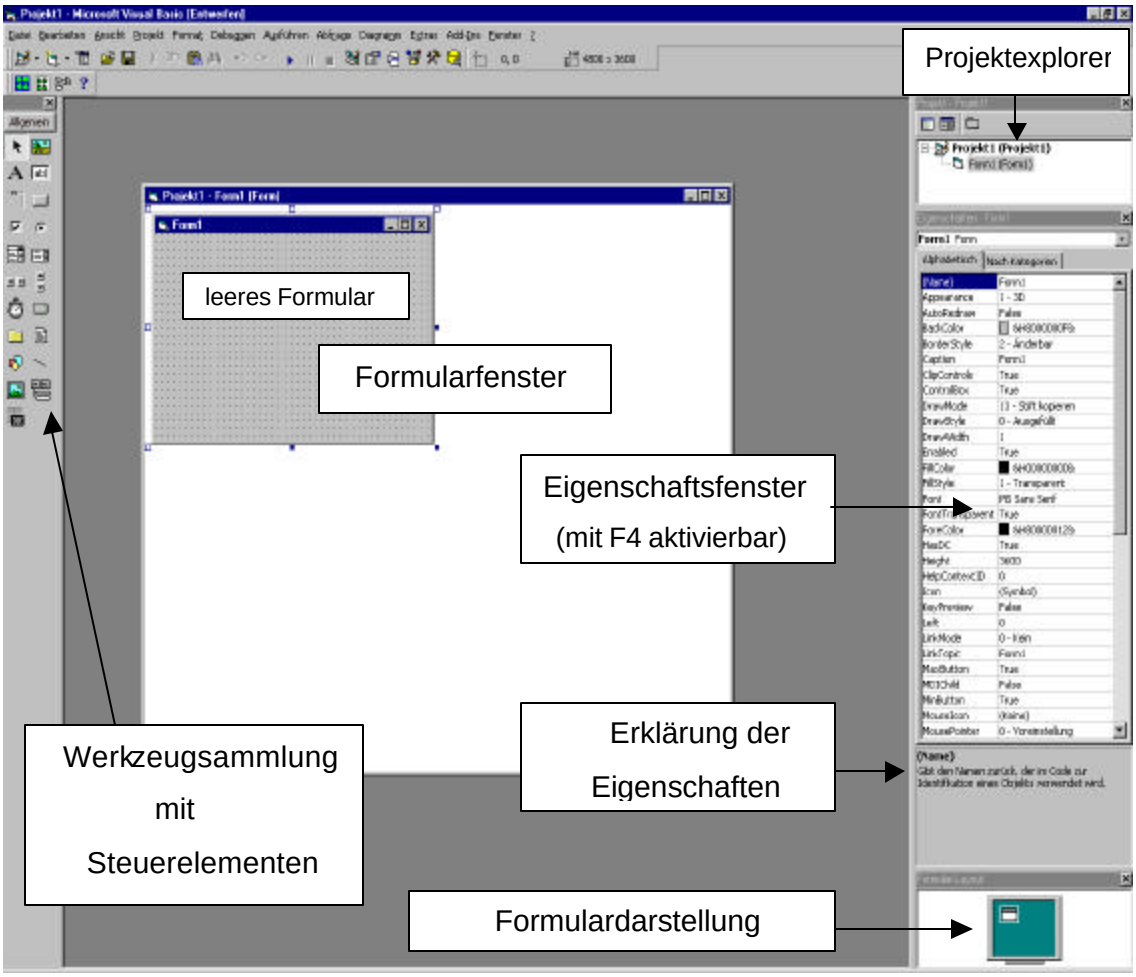


Abbildung 7.1 Visual - Basic - Entwicklungsumgebung

Jedes VB – Programm besteht aus mindestens einem Formular, auf dem die graphischen Elemente platziert sind. Es sind meistens die Steuerelemente aus der Werkzeugsammlung, die dann auf die Ereignisse wie z.B. ein Mausklick auf einen Button reagieren. Die Formulardarstellung zeigt die aktuelle Größe und die Position des gerade aktivierten Formulars. Hier wird die Position des Formulars geändert, die Änderung der Größe eines Formulars geschieht über das Formularfenster.

Das Eigenschaftsfenster mit der Erklärung der Funktionen der einzelnen Eigenschaften kann je nach Wunsch entweder alphabetisch oder nach Kategorien angezeigt werden.

Wie in der Abbildung 7.1, ist auf der IDE im ersten Augeblick kein Editor zu sehen, in dem die eigentliche Programmierung stattfindet. In VB unterscheidet man zwischen Code und Objekt.

Objekt ist die Bedienoberfläche, in der Entwicklungsphase das Formularfenster, wo die Steuerelemente platziert werden. Durch einen Doppelklick auf das zugefügte Steuerelement gelangt man in den Code – Editor von VB, wo die Verarbeitung der Ereignisse stattfindet.

Die zweite Möglichkeit bietet der Projektexplorer, der die zwei Buttons zum umschaltet zwischen Code und Objekt besitzt (Abb. 7.2). Selbstverständlich bietet IDE auch die Möglichkeit mehrere Code- und/oder Objektfenster gleichzeitig offen zu halten.

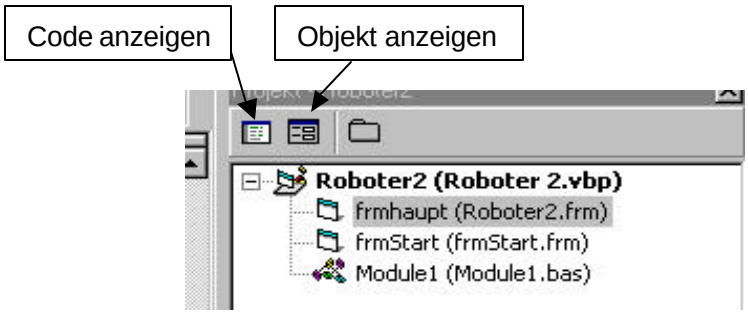


Abbildung 7.2 Projekt Explorer

7.3 Das Prinzip der VB - Programmierung

Der größte Vorteil bei der VB – Programmierung ist das schnelle Gestalten der Bedienoberfläche. Nach der Überlegung, wie die Visualisierung zu gestalten ist, werden die notwendigen Steuerelemente mit der Maus auf das Formular gezogen, um dann mit Hilfe der Eigenschaften zu parametrieren. Gehören die notwendigen Steuerelemente nicht zu den Standartsteuerelementen, können weitere Steuerelemente aus den Bibliotheken in die Werkzeugsammlung eingefügt oder selbst erstellt werden.

Der Verfasser hat in seiner Arbeit unter anderen für die Ansteuerung der seriellen Schnittstelle das Steuerelement „MSComm“ gebraucht.

Über die Menüleiste -> „Projekt“ -> „Komponente“, gelangt der Programmierer zu dem gesuchten Steuerelement („Microsoft Comm Controll 6.0“).

Das Zusatzsteuerelement „MSComm“ erscheint nach dem Einfügen in der Werkzeugsammlung und kann im Programm sofort eingebunden werden.

Das Programmieren selbst geschieht meist über die Eigenschaften der Steuerelemente und damit verbundene Ereignisse. Hier kommt die größte Anforderung an dem Programmierer. Sie lautet: „Was kann das Steuerelement und wie kann ich seine Eigenschaften sinnvoll nutzen?“. Als sehr hilfsreich erweist sich die Tatsache, dass das VB das Eigenschaftsfenster besitzt, das die vielen Eigenschaften der Steuerelemente Stichpunktartig erklärt (Abb. 7.1).

„MSComm“ besitzt stolze 27 spezifische Eigenschaften, die je nach Anwendung eingesetzt werden können. Wie am Anfang schon erwähnt, ist VB eine ereignisgesteuerte Programmiersprache. Auch die Kommunikation zwischen dem VB – Programm und der seriellen Schnittstelle kann durch entsprechende Parametrierung der Eigenschaften ereignisgesteuert gemacht werden.

Mit Hilfe der Eigenschaft „RThreshold“ kann die Anzahl der Zeichen festgesetzt werden, auf die das entsprechende „CommEvent“ reagiert. Ist „RThreshold“ z.B. mit 1 belegt, so kommt nach jedem Zeichen von der seriellen Schnittstelle, das „CommEvent“ - Ereignis. Jetzt kann der Programmierer in dem Ereignis „CommEvent“ die entsprechende Programmroutine programmieren oder aber auch eine Eigenschaft eines Steuerelements ansprechen wie z. B. Text- oder Farbewechsel.

Der zweite große Vorteil von VB sind seine Assistenten. Nach dem Start von VB fragt der erste Assistent nach Art des zu erstellendem Projekts. Auch bei Ansprechung der Eigenschaften der Steuerelemente sind die Assistenten sehr hilfsreich (Abb. 7.5). Selbst die von dem Programmierer Selbsterstellten Funktionen lassen sich in die IDE einbinden. Wird die Funktion, nach dem sie deklariert wurde, im Programm aufgerufen, so erscheinen bei der Referenzübergabe die Übergabeparameter wie in der Abbildung 7.3 automatisch im Programm. Die Add-Ins erleichtern dadurch die Entwicklungsarbeit.

Beispiel 7.1 :

```
Public Sub sendecom1(sollpos1, achse, speed)   'Unterprogrammdeklaration
……..
End Sub
```

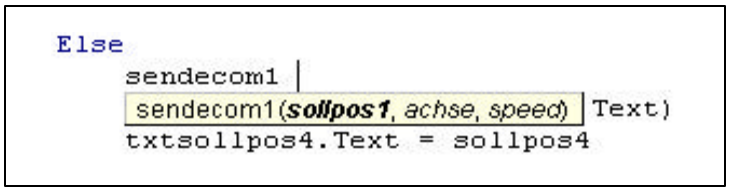


Abbildung 7.3 Aufruf der Funktion aus Bsp.7.1

Das Programmieren der Ereignisse wird sehr hilfsreich von VB unterstützt.
Ein Doppelkick im Formularfenster auf ein CommandButton mit der Bezeichnung „cmdAchse3minus“ führt zum Aufruf folgender Funktionsvorlage:

```
Private Sub cmdAchse3minus_Click(Index As Integer)
.
.       ' hier kommt die Programmierroutine
.
End Sub
```

Der Programmierer muss an dieser Stelle die Funktion „cmdAchse3minus_Click“ ausprogrammieren. Die Erweiterung des Funktionsnamen (hier „_Click“) hilft, neben dem vom Programmieren eingefügtem Kommentar, beim Erkennen um was für ein Ereignis sich bei der Funktion handelt.

7.4 Die Standardsteuerelemente

Das Erstellen eines neuen VB – Programms beginnt mit dem Gestalten des Aussehens eines Formulars. Die Visualisierung geschieht meistens (man kann auch Bilder etc. einfügen) mit Hilfe der Steuerelemente. Man unterscheidet zwischen den Standartsteuerelementen und Zusatzsteuerelementen.
Die 21 Standartsteuerelemente (Abb. 7.4) stehen dem Programmierer beim Starten eines neuen Programms sofort zu Verfügung und reichen für gewöhnliche Aufgaben durch die Vielzahl von Eigenschaften vollkommen aus.

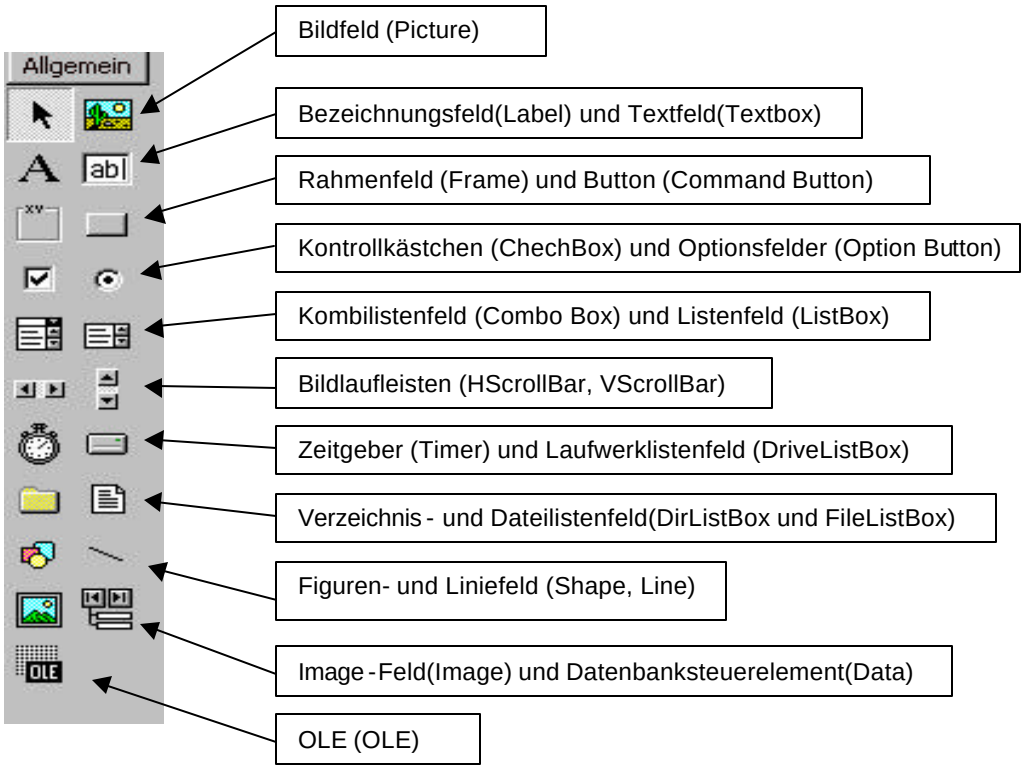


Abbildung 7.4 Werkzeugliste mit Steuerelementen

TIPP: Nach dem ein Steuerelement in ein Formular eingefügt wurde, ist eine einheitliche und aussagekräftige Namenskonvention sehr vom Nutzen. Das VB schlägt selbst einen Namen vor, der für das Verständnis der Prozeduren nicht gerade hilfsreich ist (z.B. Command17). So hat es sich eingebürgert alle CommandButtons mit dem Kürzel cmd + Funktionsbezeichnung umzubenennen. Soll z.B. ein Button eine Multiplikation zweier Zahlen hervorrufen, so wäre eine der möglichen logischen Bezeichnungen eine cmdMultiplikation.

einige Namenskonventionen der Steuerelemente in VB:

command() ->cmd Funktionsbezeichnung

text() -> txt Funktionsbezeichnung

label() -> lbl Funktionsbezeichnung

picturr() ->pic Funktionsbezeichnung

Ähnlich auch die logische Bezeichnung für ein Formular z.B. das Formular Start bekommt eine frmStart – Bezeichnung.

Der Zugriff auf die Eigenschaften der Steuerelemente wird auch von dem Assistenten begleitet. Nach dem der Programmierer den Namen des Steuerelementes mit einen zusätzlichen Punkt. im Programm eingegeben hat, werden alle Eigenschaften des entsprechenden Steuerelementes im einem Scrollbalken angezeigt (Abb.7.5 lblswitch8). Jetzt kann man die gewünschte Eigenschaft auswählen und parametrieren.

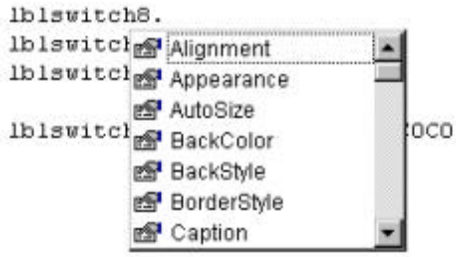


Abbildung 7.5 Zugriff auf die Eigensc haften eines Steuerelementes

7.5 Die Schnittstellen – Programmierung mit Visual Basic

7.5.1 Zugriff auf die serielle Schnittstelle RS 232 (COM1)

Visual Basic bietet eine sehr komfortable Methode auf die serielle Schnittstelle zuzugreifen. Über das Steuerelement „MSComm“ wird die serielle Schnittstelle parametriert und angesprochen. Das Steuerelement gehört nicht zu den Standartsteuerelementen und muss deshalb vom Programmierer in die Werkzeugliste manuell eingefügt werden (Kap. 7.3). Die graphische Darstellung des „MSComm“ ist auf dem Formular ist auf Abb. 7.6 zu sehen.

Abbildung 7.6 Das Steuerelement „MSComm“

Die schon im Kapitel 7.3 angesprochene „RThreshold“ – Eigenschaft von dem Steuerelement kann die Kommunikation mit Hilfe von Feststellung der Anzahl der eintreffenden Zeichen an der seriellen Schnittstelle steuern.

Die Grundkonfiguration der seriellen Schnittstelle wird üblicherweise beim Laden des entsprechenden Formulars (siehe dazu Kapitel 8.2.1) durchgeführt.

Wie im Kapitel 7.3 erklärt, kann ein VB – Programm mit Hilfe der Eigenschaft **„RThreshold“** mit der seriellen Schnittstelle ereignisgesteuert kommunizieren.
Beim Starten des Formulars wird „Form_Load“ - Ereignis aufgerufen. Hier kann die serielle Schnittstelle initialisiert werden.

Beispiel für die Konfiguration der seriellen Schnittstelle RS 232 (COM 1):
Beispiel 7.2 :

```
MSComm1.CommPort = 1
      ‘Port Nummer 1 für COM1
MSComm1.Settings = "9600,N,8,1"
      ‘9600 =Baudrate,
      ’n = none für keine Parität, even = gerade, odd= ungerade Parität
      ’8 = Anzahl der Datenbits
      ’1 = Anzahl der Stoppbits
MSComm1.InputMode = comInputModeText
      'Datentyp von den am Input liegenden Daten
MSComm1.InputLen = 0
       ’ Gibt die Anzahl der Zeichen, die bei jedem Lesezugriff gelesen werden
       ’ 0 = alle Zeichen lesen
MSComm1.PortOpen = True
      'serielle Schnittstelle öffnen
MSComm1. RThreshold = 1
```

Mit jedem ankommenden Zeichen wird das „**OnComm**“ – Ereignis aufgerufen
Hier werden im VB - Ansteuerungsprogramm des Roboters die Zeichen ausgewertet. (siehe das nachfolgende Beispiel)

Beispiel 7.2

```
Private Sub MSComm1_OnComm()                'Serielle Schnittstelle
Dim zahl As Variant                         ‘Deklaration der Variabel
Dim inputzähler As Integer                  'Anzahl von COM1 (MSComm)
                                            ’zurückgegebenen Zeichen
```

```
Select Case MSComm1.CommEvent
  Case comBreak                                   'Break-Zeichen eingetroffen
   Case comerrxparity                             'Paritätsfehler
    MsgBox „Paritätsfehler“
  Case comEvReceive                               'Zeichen angekommen

     rückgabe = MSComm1.Input      'Ist-Position Übergabe vom Roboter
     inputzähler = Len(rückgabe) 'Anzahl von MSComm zurückgegebenen
                                  ' Zeichen
    'Label2.Caption = inputzähler
     flag = False

If inputzähler = 8 Then                           'Rückgabe der Istwerte
   Istwerte_vom_Rob (rückgabe)
'Unterprogramm: Verarbeitung und Weitergabe der Istwerte
End If

If (inputzähler = 3 And flagtimeIstAbfrage = False) Then      'Rückgabe der I/O
                                                              'Testbox
   digout = rückgabe
   Digout_vom_Rob (digout)
End If

If inputzähler = 2 Then                           'Rückgabe Sollposition erreicht
   flag_Position_erreicht = True
   Beep
   lblPos_erreicht.Caption = 1
  ' lblPos_erreicht.LinkPoke                      'Rückgabe an die SPS
   Soll_Pos_erreicht                             'Soll Position erreicht
Else
    lblPos_erreicht.Caption = 0
   'lblPos_erreicht.LinkPoke
    End If
```

```
    If inputzähler = 5 Then                    'Rückgabe der Versionsnummer
      cmdversion.Caption = "Versionsnummer"

    End If
End Select

End Sub
```

Erklärung: Mit Select Case wird abhängig von dem MSComm1.**CommEvent** – Ausdruck die Programmierroutine ausgeführt. Wird ein Zeichen vom ROB3 an die serielle Schnittstelle gesendet (z.B. Istwerte der Roboterachsen) und an der COM1 – Schnittstelle empfangen wird die Programmierroutine nach **Case comEvReceive** ausgeführt.

Mit „rückgabe = MSComm1.Input“ werden die ankommenden Zeichen an die Variable „rückgabe“ übergeben und gezählt. Mit Anzahl der Zeichen kann bestimmt werden was ROB3 gesendet hat. Bei acht Zeichen handelt es sich um die Rückgabe der Rückwerte, bei drei um die Rückgabe der Werte aus der I/O Box usw.

7.5.2 DDE - Schnittstelle

Mit der Einführung des Betriebssystems Windows stellt Microsoft die Methode des Dynamic Data Exchange (DDE) vor. Diese Methode arbeitet nach dem Client – Server – Prinzip. Dabei stellt der Server dem Client die Daten zur Verfügung. Die Kommunikation kann zwischen mehreren Konversationspartnern gleichzeitig bestehen (z. B. SPS – Software, Word oder Excel Dokument). Die Adressierung besteht aus:

- der Anwendung (Applikation, z. B. SPS Projekt >> „PROSYS|D:\DDE.pro“), Datum (Item, z. B. ein Datenelement als globale Variable >> „**.**Schalter1“) und dem Typ der Verbindung(LinkMode, z.B. 1 = vbLinkAutomatic, Verbindung automatisch bei Änderung der Daten)

Unter Visual Basic 6.0 kann eine DDE – Verbindung zwischen einem Visual – Basic – Programm und Software –SPS (ProSys) folgendermaßen aussehen:

```
txtspeed.LinkTopic = ProjektName
txtspeed.LinkItem = ".Schalter1"
```

txtspeed.**Link**Mode = vbLinkAutomatic,

wobei in einem Visual – Basic – Programm direkt nur drei Steuerelemente an einer DDE – Konversation teilnehmen können:

- Bezeichnungsfelder (Label)
- Textfelder (TextBox) und
- Bildfelder (PictureBox)

Wie auf dem Beispiel auf der anderen Seite zusehen ist, hat die Vorsilbe **Link** immer etwas mit einer DDE – Verbindung zu tun.

7.5.2.1 Beispiel einer DDE – Verbindung mit Fehlerbehandlung

Das folgende Beispiel zeigt eine DDE – Verbindung zum SPS – Programm. Die Kommunikation von SPS erfolgt über drei Datenelemente, eing1, eing2, ausg1, die als globale Variable im SPS – Programm definiert sind.

Bei Parameterübergabe >> Text1.LinkItem = "**.**eing1" bitte den Punkt vor Variabelnamen beachten Das Visual – Basic – Programm benutzt als Steuerelement die Textfelder (Abb.7.7).

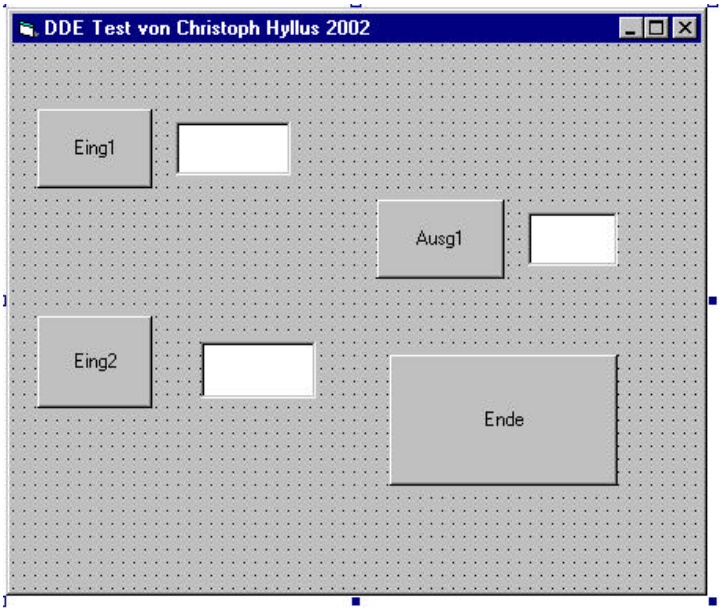


Abbildung 7.7 Formular zur DDE – Testansteuerung

Beim Schließen eines DDE – Kanals („LinkClose“ Ereignis) erscheint sofort eine Fehlermeldung (MessageBox Abb. 7.8) auf dem Bildschirm, die den Anwender auffordert das SPS – Programm zu starten oder das VB – Programm zu beenden.

Die gleiche Abfrage erscheint beim Laden des Formulars.

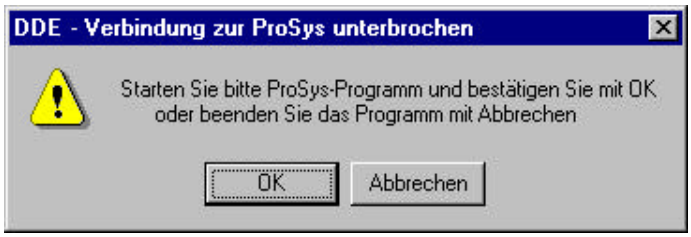


Abbildung 7.8 Fehlermeldung beim LinkClose Ereignis

Quellcode zur DDE – Ansteuerung:

```
'Test Verbindung zur ProSys SPS - Software
'DDE Fehler Behandlung

Option Explicit
Dim ProjektName As String       ' Variablen Deklaratio

Private Sub Form_Load()

   DDE_Verbindung               ' >> DDE– Routine + Fehlerbehandlung

End Sub

Private Sub DDE_Verbindung()     'Unterprogramm DDE Verbindung herstellen + Fehler
Behandlung
Dim Mldg, Stil, Titel, Antwort    'Parameter für Message Box
   ProjektName = "PROSYS|D:\DDE.pro"

On Error GoTo DDE_Fehler          ' DDE - Kommunikationsfehler >> Sprung zur Marke DDE_Fehler

  txteing1.LinkTopic = ProjektName                ' "PROSYS|D:\Rob1.pro" D :\1test\dde.pro"
  txteing1.LinkItem = ".eing1"                     '
  txteing1.LinkMode = vbLinkAutomatic

  txteing2.LinkTopic = ProjektName
  txteing2.LinkItem = ".eing2"                     '
  txteing2.LinkMode = vbLinkAutomatic

  txtausg1.LinkTopic = ProjektName
  txtausg1.LinkItem = ".ausg1"                     '
```

```
    txtausg1.LinkMode = vbLinkAutomatic
```

```
 Exit Sub
DDE_Fehler:                    ' DDE - Fehler
Mldg = "Starten Sie bitte ProSys-Programm und bestätigen Sie mit OK " & Chr$(10) & "      oder
beenden Sie das Programm mit Abbrechen"   ' Meldung definieren.
Stil = vbOKCancel + vbExclamation + vbDefaultButton1
     'Schaltflächen definieren.
Titel = "DDE - Verbindung zur ProSys unterbrochen"   ' Titel definieren.
Antwort = MsgBox(Mldg, Stil, Titel) ' Hilfe, Ktxt)   ' Meldung anzeigen.
If Antwort = vbOK Then   'Benutzer hat "OK" gewählt.
    Resume          'Fortsetzung bei der Anweisung >> Fehler
Else 'Benutzer hat "Abbrechen" (geht auch über vbCancel)
     ' oder Abbruch(geht auch über vbAbort)gewählt
   End         'Programm beenden

End If

End Sub

Private Sub cmdEing1_Click()
txteing1.Text = 1                'zur SPS
    txteing1.LinkPoke
End Sub

Private Sub cmdEing2_Click()
txteing2.Text = 1
    txteing2.LinkPoke             'zur SPS
End Sub

Private Sub cmdAusg_Click()
txtausg1.Text = 1                'zur SPS
    txtausg1.LinkPoke
End Sub

Private Sub txteing1_LinkClose()
    DDE_Verbindung
End Sub

Private Sub txteing2_LinkClose()
    DDE_Verbindung
```

```
End Sub

Private Sub txtausg1_LinkClose()    ‘DDE - Verbindung unterbrochen
    DDE_Verbindung
End Sub

Private Sub cmdEnde_Click()
txteing1.LinkMode = None              'beendet DDE - Verbindung
txteing2.LinkMode = None              'beendet DDE - Verbindung
txtausg1.LinkMode = None              'beendet DDE - Verbindung
  End                                       ’Programm Ende
End Sub
```

Erklärung:

Die Rückgabe zu SPS geschieht über „**LinkPoke**“ – Methode.

```
Private Sub cmdEing1_Click()
  txteing1.Text = 1
  txteing1.LinkPoke          'Rückgabe des Inhaltes von txteing1 zur SPS - > eing1
End Sub
```

8 Programmbeschreibung

8.1 Allgemeiner Programmablauf

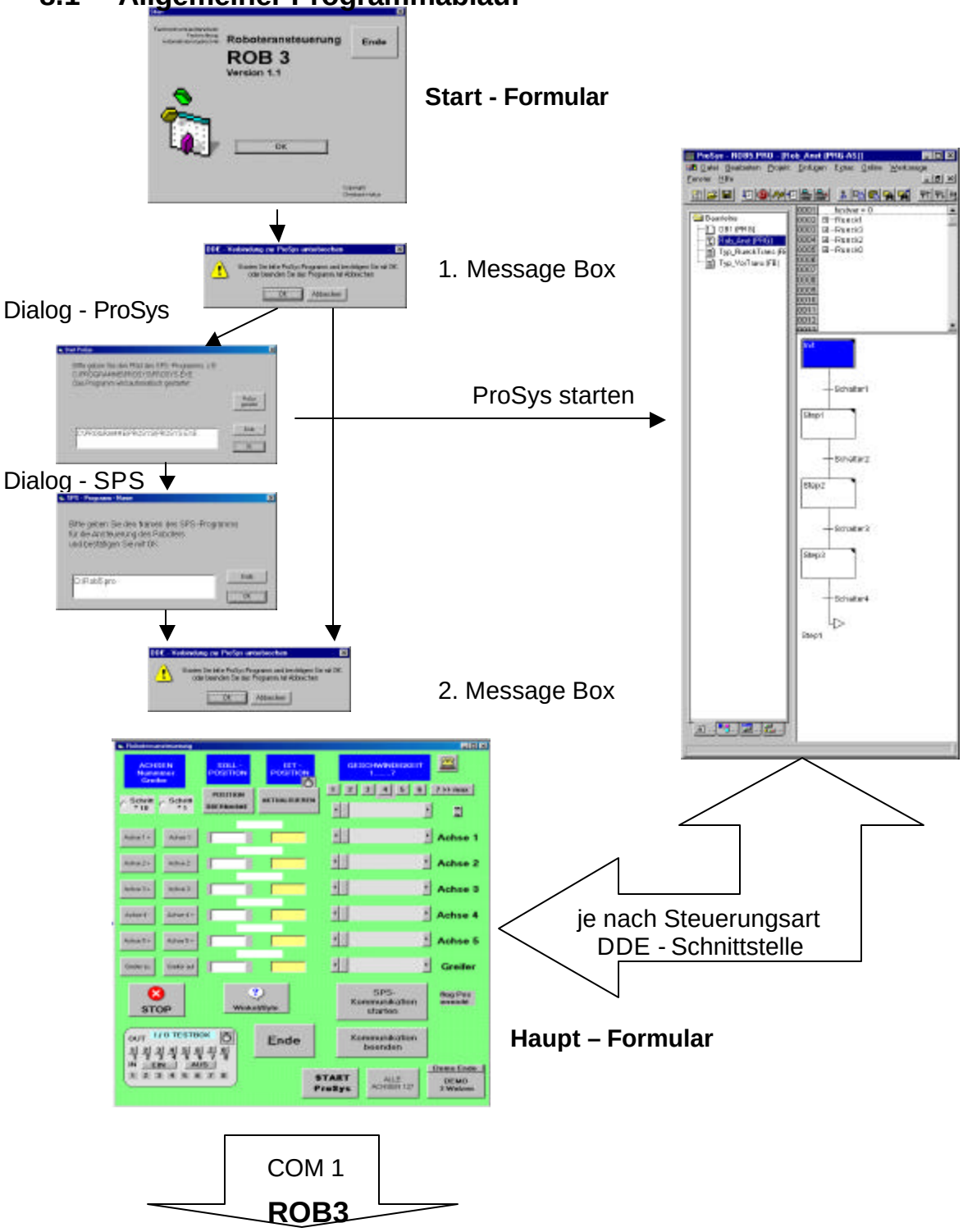


Abbildung 8.1 Programmablauf in der Ansteuerung des ROB3

8.2 Visual Basic Programm

Das VB – Programm zur Ansteuerung des ROB3 besitzt vier Formulare:

- Start-Formular,
- Dialog – ProSys – Formular,
- Dialog – SPS – Formular und
- Haupt – Formular.

Die Abbildung 8.1 zeigt die Struktur des VB - Programmablaufs.

Mit Hilfe der ersten Message - Box kann der Anwender die Art (Sollwerte von der SPS oder B&B – Oberfläche) der Ansteuerung auswählen. Die zweite Message - Box fragt den Anwender nach dem Format der Sollwerte (Winkel oder Byte 0..255).

8.2.1 Das Start – Formular

Das Start – Formular wird als Informationsformular benutzt. Nach dem Bestätigen durch „OK“ - Button wird in der Message - Box die Ansteuerungsart gefragt. Je nach Auswahl der Steuerungsart (SPS oder B&B) wird im Programm **flag_SPS** gesetzt bzw. rückgesetzt. Nachdem der Anwender keine SPS – Ansteuerung gewählt hat, wird sofort das Hauptformular gestartet.

Abbildung 8.2 Startformular

8.2.2 Das Dialog - ProSys - Formular

Nach der Auswahl der Steuerungsart - SPS (!) wird das zweite Formular gestartet und das Start – Formular beendet (entladet). Das Formular hat als Funktion das Programm „ProSys“ zu starten. Der Anwender kann in der Text - Box den Pfad

von „ProSys“ angeben. Der Pfad wird nach der Bestätigung mit „OK“ – Button in die Ablaufroutine übernommen. Mit „ProSys gestartet“ – Button kann die ProSys - Startroutine des Programms übersprungen werden und das Dialog - SPS - Formular gestartet werden.

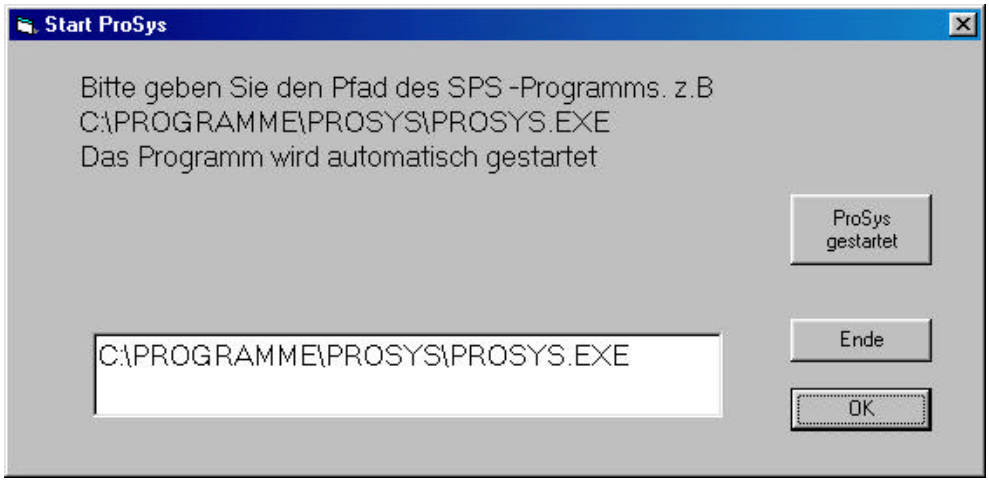


Abbildung 8.3 Dialog – ProSys – Formular

8.2.3 Das Dialog - SPS - Formular

In der Steuerungsart SPS wird nach dem ProSys – Dialog – Formular das Dialog – SPS – Formular gestartet (Abb. 8.4). In dem Formular werden der Pfad und der Name des in ProSys erstellten SPS – Programms zur Ansteuerung des Roboters gefragt. Es *müssen mindestens alle in der Tabelle 8.1* (Kapitel 8.3) SPS – Variabel im angegebenen SPS – Programm (hier „Rob5“) als *globale* Variabel deklariert sein.

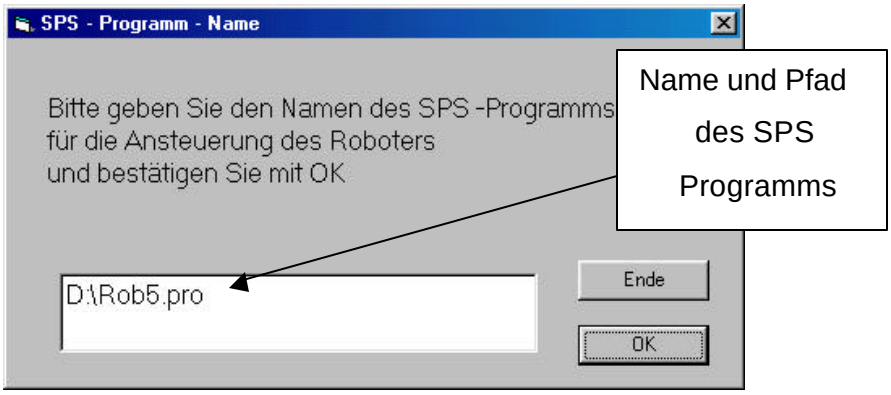


Abbildung 8.4 SPS - Programm – Name – Formular

8.2.4 Das Hauptformular

Beim Starten des Hauptformulars wird der Anwender zuerst über Start – Button aufgefordert den Roboter ROB3 einzuschalten und zurückzusetzen (Reset – Taster an der Rückseite).

Wird die Steuerungsart SPS gewählt, versucht das VB – Programm im Haupt – formular die DDE – Verbindung aufzubauen (Unterprogramm DDE_Verbindung, siehe dazu Kapitel 8.2.4.1). Der Anwender kann die VB – Oberfläche nur als Meldeoberfläche benutzen, d.h. es besteht keine Möglichkeit die Roboterpositionen von dieser Oberfläche anzusteuern. Bei erfolgreichem Aufbau der DDE – Verbindung wird der Anwender nach dem Format der Sollwerte gefragt. Je nach Auswahl wird dadurch im Programm **flag_Winkel** gesetzt oder rückgesetzt.

Bei Ansteuerungsart nur über B&B wird sofort das Format der Sollwerte abgefragt. Dabei besteht keine DDE - Verbindung zum SPS – Programm.

Abhängig von der Steuerungsart gibt es im Hauptformular zwei ähnliche, aber unterschiedliche Oberflächen (Abb.8.5):

- eine für Ansteuerung des Roboterarmes über SPS und
- eine für Ansteuerung nur über VB – Oberflächen.

Beide Oberflächen bekommen die Ist – Zustände der Roboterachsen zahlenmäßig (Bezeichnungsfeld, Label) und graphisch (Scrollbalken) angezeigt.

Bei der Vorgabe der Sollwerte bei der Ansteuerungsart ohne SPS – Verbindung kann der Anwender die Achsensollwerte als Zahlenwert direkt in die Text – Box übergeben und mit „Position – Übernahme“ – Button bestätigen oder über „Schritt“ – Buttons die Achsen und der Greifer einzeln ansteuern. Die Schritten können je nach Auswahl in 1-er oder 10-er Schritten erfolgen.

Mit dem „Aktualisieren“ – Button werden die Istwerte auch in die Text – Box der Sollwerte übertragen.

Zusätzlich werden die Zustände der I/O Testbox graphisch dargestellt. Die I/O Box kann von beiden Steuerungsarten benutzt werden, vorrausgesetzt man hat sie eingeschaltet (Einschaltbutton auf der B&B).

Der Anwender kann in B&B die Kommunikation zum SPS – Programm starten bzw. beenden. Die Formate der Sollwerte für die Achsen können auch umgeschaltet werden. Über den „Stop“ – Button wird der Roboter gestoppt (Übergabe der Ist – Werte als neue Sollwerte). Die Geschwindigkeit des Roboters kann über Buttons eingestellt werden.

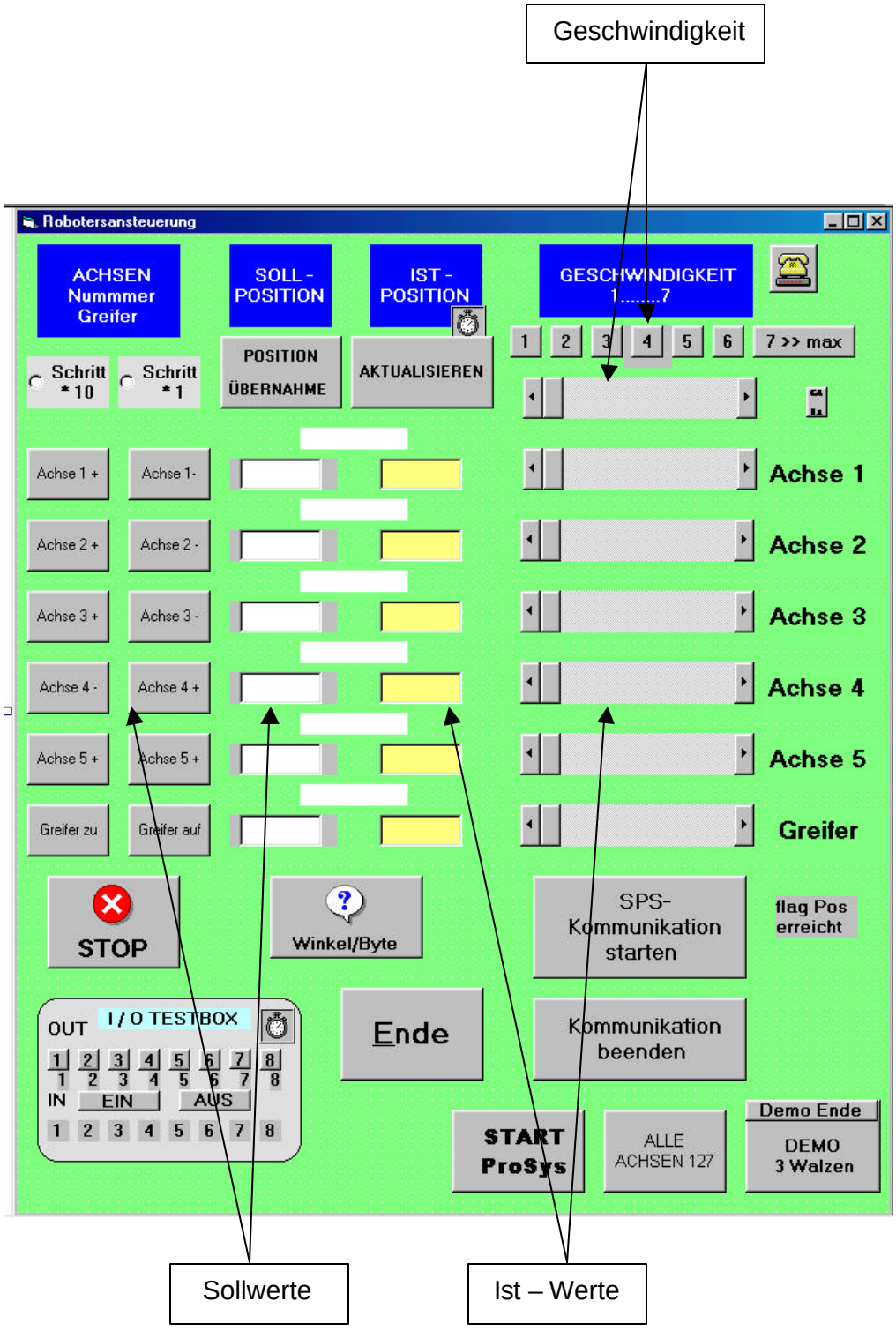


Abbildung 8.5 Hauptformular

8.2.4.1 Ablauf im Unterprogramm DDE_Verbindung

<table>
<tr><td colspan="4">Hauptformular</td></tr>
<tr><td colspan="4">Load – Ereignis :</td></tr>
<tr><td colspan="3">flag_SPS = True
Pfadübergabe + Projektname (ProSys)
Unterprogramm DDE_Verbindung
(versucht die DDE_Verbindung aufzubauen)</td><td rowspan="6">flag_SPS = False
No_DDE_Oberfläche laden</td></tr>
<tr><td colspan="3">® Verbindung aufgebaut (SPS - Projekt wurde gestartet)</td></tr>
<tr><td>Ja
Verbindung aufgebaut (SPS - Projekt ist vorhanden und gestartet)</td><td colspan="2">Nein -> DDE_Fehler

SPS - Projekt oder DDE - Kanal nicht vorhanden
oder
SPS – Projekt wurde nicht gestartet</td></tr>
<tr><td rowspan="3">DDE_Oberfläche starten</td><td colspan="2">Message Box
„DDE – Verbindung zu ProSys unterbrochen“</td></tr>
<tr><td>„Ja“
Unterprogramm
DDE_Verbindung</td><td rowspan="2">„Nein“
flag_SPS =False

Unterprogramm
No_DDE_
Oberfläche
laden</td></tr>
<tr><td>⇧
siehe
oben ®</td></tr>
</table>

Abbildung 8.6 Programmablauf im Hauptformular

8.3 Die DDE – Schnittstelle von/zu SPS

Visual Basic	Format	**SPS**	Format	Funktion
lblsollpos0_DDE	Global	achse1soll	Global	Sollwertvorgabe von SPS
lblsollpos1_DDE	Global	achse2soll	Global	Sollwertvorgabe von SPS
lblsollpos2_DDE	Global	achse3soll	Global	Sollwertvorgabe von SPS
lblsollpos3_DDE	Global	achse4soll	Global	Sollwertvorgabe von SPS
lblsollpos4_DDE	Global	achse5soll	Global	Sollwertvorgabe von SPS
lblsollpos0_DDE	Global	Greifersoll	Global	Sollwertvorgabe von SPS
lblspeed_DDE	Global	speed	Global	Sollwertvorgabe von SPS
lblAchse1ist	Global	achse1ist	Global	Istwertvorgabe zu SPS
lblAchse2ist	Global	achse2ist	Global	Istwertvorgabe zu SPS
lblAchse3ist	Global	achse3ist	Global	Istwertvorgabe zu SPS
lblAchse4ist	Global	achse4ist	Global	Istwertvorgabe zu SPS
lblAchse5ist	Global	achse5ist	Global	Istwertvorgabe zu SPS
lblGreiferist	Global	Greiferist	Global	Istwertvorgabe zu SPS
lblPos_erreicht	Global	.flag_Pos_erreicht	Global	Rob Position erreicht
I/O Box				
lblswitch1	Global	Schalter1	Global	Schalterzustand1 zu SPS
lblswitch2	Global	Schalter2	Global	Schalterzustand2 zu SPS
lblswitch3	Global	Schalter3	Global	Schalterzustand3 zu SPS
lblswitch4	Global	Schalter4	Global	Schalterzustand4 zu SPS
lblswitch5	Global	Schalter5	Global	Schalterzustand5 zu SPS
lblswitch6	Global	Schalter6	Global	Schalterzustand6 zu SPS
lblswitch7	Global	Schalter7	Global	Schalterzustand7 zu SPS
lblswitch8	Global	Schalter8	Global	Schalterzustand8 zu SPS
lblLED1	Global	OUTLED1	Global	LED Ansteuerung von SPS
lblLED2	Global	OUTLED2	Global	LED Ansteuerung von SPS
lblLED3	Global	OUTLED3	Global	LED Ansteuerung von SPS
lblLED4	Global	OUTLED4	Global	LED Ansteuerung von SPS
lblLED5	Global	OUTLED5	Global	LED Ansteuerung von SPS
lblLED6	Global	OUTLED6	Global	LED Ansteuerung von SPS

Visual Basic	Format	**SPS**	Format	Funktion
I/O Box				
lblLED7	Global	OUTLED7	Global	LED Ansteuerung von SPS
lblLED8	Global	OUTLED8	Global	LED Ansteuerung von SPS

Tabelle 8.1 Übergabe - Parameter von/zu SPS

Beispiel.

```
lblsollpos0_DDE.LinkTopic = ProjektName                    '
lblsollpos0_DDE.LinkItem = ".achse1soll"                   '
lblsollpos0_DDE.LinkMode = vbLinkAutomatic'
                                   -> aktiviert Verbindung automatisch
```

Über die Stringvariabel „ProjektName“ wird der Pfad (Position des Programms) und der Name des SPS - Programms (ProSys) sowie des Projektes (ROB5.pro) übergeben.

Es müssen alle in der Tabelle angegebenen SPS – Variabel im Programm deklariert sein. Beim einem Verbindungsfehler meldet das VB – Programm, dass keine DDE – Verbindung aufgebaut werden konnte bzw. unterbrochen wurde (Abb. 8.6).

9 Schlussfolgerung

9.1 Zusammenfassung der Ergebnisse

Die Zusammenfassung in der Diplomarbeit erreichten Ergebnisse kann in drei Unterkapitel, die im Laufe der Diplomarbeit zu Schwerpunkten wurden, unterteilt werden:

- Die Schnittstellen- und B&B - Programmierung
- Vorwärts- und Rückwärtstransformation und
- SPS – Programmierung

9.1.1 Die Schnittstellenprogrammierung

Bei der Schnittstellenprogrammierung musste der Verfasser zuerst die geeignete Programmiersprache wählen. Die Wahl fiel auf Visual Basic. Diese Programmiersprache bietet die erforderlichen Möglichkeiten, eine Verbindung zu einer anderen Windows-Applikation herzustellen und kann zugleich die serielle Schnittstelle ansprechen.

Visual Basic bietet auch die Möglichkeit, sehr schnell eine graphische Oberfläche zu gestalten. Die Bedienoberfläche ist in diesem Fall für die Ansteuerung des Roboters und Visualisierung der Ist-Zustände der einzelnen Achsen und Schalter der I/O – Testbox zuständig.

Die Programmierung der Schnittstelle, die den Roboter anzusteuern orientiert sich nach dem Low – Level - Protokoll. Die Abfrage der Ist- Positionen der Achsen des Roboters geschieht über einen Timer, der ein entsprechendes Kommando in fest vorgegebenen Abständen an den Antriebskontroller schickt.

Über einen zweiten Timer werden die Schalterzustände der I/O Box abgefragt.

Die DDE – Schnittstelle zwischen dem VB- und SPS Programm reagiert auf die Änderung der für die DDE – Verbindung benutzten Steuerelemente. Die Auswahl erlaubt die Vergabe der neuen Achsenpositionen entweder von der SPS oder direkt von der Bedienoberfläche. Beim Abbruch der Verbindung wird eine entsprechende Fehlerroutine gestartet (siehe Kapitel 7.5.2.1).

9.1.2 Vorwärts - und Rückwärtstransformation

Das Problem, die Werte von Weltkoordinatensystem auf das Achsen -spezifische umzurechnen erwies sich als eine komplexe mathematische Aufgabe. Die Lösungsgleichungen für die Achsenwinkel wurden aus Gleichungen aus [5] bestimmt, nachdem sie für den Spezialfall ROB3 modifiziert worden sind.
Der Verfasser hat die Denavit – Hartenberg - Methode für die Aufstellung der Lösungsgleichungen zur den einzelnen Achsen ausführlich beschrieben.
Die Denavit – Hartenberg – Methode wurde anhand des Beispiels am ROB3 erklärt.

9.1.3 SPS – Programmierung

Der Schwerpunkt der SPS – Programmierung lag in der Erstellung der beiden Funktionsbausteine, die Vor- und Rückwärtstransformationen durchführen.
Be der Umsetzung der mathematischen Formeln mussten noch ATAN2 - Funktion, RAD - GRAD – und REAL – INTEGER Umrechnung programmiert werden.
Für die Vorgabe der Sollpositionen wurde eine Ablaufsteuerung programmiert.

9.2 Bewertung der Ergebnisse

Das Hauptziel der Arbeit (Punkt 2 im Kapitel 1.2.1), den Roboterarm auf die von der SPS vorgegebene Position zu fahren, wurde vollständig erreicht. Je nach Einstellung kann der Anwender die Position als Winkel oder Integer – Zahl vorgeben.
Der Punkt 1 im Kap. 1.2.1 gesetzten Ziele wurden vollständig erreicht. Die Entwicklung und die Notwendigkeit der Roboter wurden anhand Beispiele erläutert.
Um das Hauptziel zu erreichen wurde die Schnittstelle zum Antriebscontroller und Bedien- und Beobachtungsoberfläche programmiert.
Die Vorgabe der Sollwerte kann entweder von der erstellten Bedien- und Beobachtungsoberfläche (B&B) oder durch ein SPS – Programm erfolgen.
Damit wurden Unterpunkte 2.2 und 2.3 der Aufgabestellung vollständig erreicht.
Der letzte Unterpunkt der Aufgabenstellung, die Sollwerte als Weltkoordinate zu vorgeben, konnte nicht vollständig erreicht werden. Der Verfasser hat die Lösungsansätze für die Transformationen und die Denavit – Hartenberg – Methode in seiner Arbeit beschrieben. Die Gleichungssysteme konnten durch Ihre Komplexität nicht gelöst werden. Auch die Modifizierung der Lösungen aus der Literatur führten nicht zu befriedigenden Ergebnissen.

9.3 Ausblick auf weiterführende Arbeiten

Ein sehr wichtiges Gebiet bei der Ansteuerung von Roboter sind die Sensoren. Mit Hilfe von Sensoren kann die Programmierung erweitert und sensibler gemacht werden. Mit einer am Endeffektor montierten Kamera könnte die Umgebung des Roboters analysiert werden. Die von der Kamera zurückgelieferten Signale könnten durch entsprechende Verarbeitung und Programmierung eine mögliche Kollisionsgefahr mit einem Hindernis vermeiden. Mit den am Greifer angebrachten Drucksensoren kann durch die Peripheriesignale die Greiferposition geregelt werden.

Als Ausblick auf weiterführende Arbeiten mit dem Roboterarm ROB3 bietet sich auch die Möglichkeit eine entsprechende Visualisierung über die Roboterposition und seine Umgebung zu gestalten. Die Ansteuerung des Roboters über diese Oberfläche und die Anbindung ans Internet könnten das Ansteuern des Roboterarmes von verschienen am Internet angebundene PC’s flexibler und komfortabler machen.

10 Abkürzungen, Fremdwörter und Erläuterungen

Alphabetisch angeordnet

Add-Ins - *Add-Ins* sind Assistenten, die in einem Programm unter Verwendung von Objekten und Selbstdeklarierten Funktionen die Visual Basic-Umgebung anpassen und erweitern. Die Add-Ins lassen sich in die IDE von VB einbinden.

AS - **A**blau**s**prache

AWL - **A**n**w**eisungs**l**iste

BASIC – Beginners All purpose Symbolic Instruction Code

B&B – Bedien- und Beobachtungsoberfläche

CP-Steuerung – CP = continuos point, Bahnsteuerung der Roboterbewegung

DDE – **D**ynamic **D**ata **E**xchange, unter Windows festgelegtes Kommunikationsprotokoll. Ermöglicht eine Kommunikation mit Windows Programmen beliebiger Hersteller.

Endeffektor – am Handgelenk des Roboters fest montierte Einrichtung um bestimmte Aufgaben zu erledigen (z.B. Greifer oder Spritzpistole)

Formular – Fenster des VB – Programms, z.B. Formular - Start

FUP - Funktionsplan

Handshake – Ein Kommunikationsprotokoll, das den Datenfluß über die serielle *->Schnittstelle*, etwa zwischen *->Modem* und PC, kontrolliert. Man unterscheidet dabei zwischen Hardware- und Software-Handshake. Hardware-Handshake ist schneller als Software-Handshake, da dafür keine Zeichen übertragen werden müssen.
[http://www.tigerweb.de/internet/glossar/lex8.htm]

IDE – **I**ntegreted **D**evelopment **E**nvironment, integrierte Enwicklungsumgebung (VB)

KOP – **Ko**ntakt**p**lan

Low – Level – Protokoll - ein nicht standardisiertes Protokoll zur Nachrichtenübertragung und dient der Ansteuerung des ROB 3 über die serielle Schnittstelle RS 232.

Pick and Place – Aufnehmen und Ablegen

ProSys – SPS – Entwicklungsumgebung für S5- und S7- Steuerungen

PtP – Steuerung – PtP = point to point, Punkt-zu-Punkt-Steuerung der Roboterbewegung

SPS – **s**peicher**p**rogrammierbare **S**teuerungen

ST – **S**trukturierte **T**ext

TBPS - **T**each **B**ox **P**rogrammier **S**ystem, eine menügeführte Programmiersoftware zur Ansteuerung des Roboters ROB 3

TCP - **T**ool **C**enter **P**oint

Teachbox – Handprogrammiergerät, mit Hilfe der Teachbox kann der Roboter ohne Verwendung eines Computers programmiert werden

VB **V**isual **B**asic

11 Verzeichnisse

11.1 Literaturverzeichnis

[1] Stefan Hesse, Günter Seitz
Robotik Grundwissen für die Berufliche Bildung
Viewegs Fachbücher der Technik ISBN 3-528-04951-0

[2] Lorbeer/Werner
Wie funktionieren Roboter
B.G. Teubner Stuttgart ISBN 3-519-12531-5

[3] M.K.Groover, M.Weiss, R.N.Nagel, N.G.Odrey
Robotik umfassend
McGraw-Hill-Texte ISBN3-89028-087-0

[4] Prof. Dietel Script zur Vorlesung Robotik
Elektrotechnik/Nachrichtentechnik

[5] E.J.Kreuzer, J.-B.Lugtenburg, H.-G.Meißner, A.Truckenbrodt
Industrieroboter ISBN 3-540-54630-8

[6] Harald Tillmann Diplomarbeit
http://www.tillmann-group.de/khm/kap5.htm

[7] TBPS für Windows 95 der FIRMA EUROBTEC

[8] Broschüre der FIRMA EUROBTEC zum Knickarmroboter ROB 3

[9] Harald Tillmann – Diplomarbeit – Kapitel V
http://www.tillmann-group/khm/kap5.htm

[10] Gerd Bittner
Workshop der Automatisierungstechnik

[11] A.Morecki, J.Knapczyk
Basics of robotics
Springer-Wien-NewYork

[12] Peter Monadjemi
Visual Basic 4 Das Kompendium
Markt&Technik

[13] Peter Monadjemi
Visual Basic 5 Das Kompendium
Markt&Technik

[14] Michael Kofler
Visual Basic 6 Programmiertechniken, Datenbanken, Internet
Addison - Wesley

[15] Prof. Dr. Dipl.-Ing. H. Tolle
Seminar – Robotik und künstlic he Intelligenz

[16] Florian Rötzer
Die Roboter kommen
www.heise.de/tp/deutsch/special/robo/8965/1.html

[17] http://www.uni-essen.de/lfm/forsch/arbgeb/entkop_sfb/skywash.html

[18] http://www.fraunhofer.de/german/publications/df/df1998/398-06.htm

11.2 Abbildungsverzeichnis

11.3 Tabellenverzeichnis

Anhang

A1 Quellecode - Module (Deklaration von globalen Variabel)

```
'Globale Variabel

Global rückkommando As Variant
Global achse As Variant

Global pos0 As Variant       'Achsen Ist - Positionen
Global pos1 As Variant
Global pos2 As Variant
Global pos3 As Variant
Global pos4 As Variant
Global pos5 As Variant

Global sollpos0 As Integer   'Soll - Position 5 Achsen  + Greifer
Global sollpos1 As Integer
Global sollpos2 As Integer
Global sollpos3 As Integer
Global sollpos4 As Integer
Global sollpos5 As Integer

Global switch1 As Boolean    'Sc halter I/O Box betätigt
Global switch2 As Boolean
Global switch3 As Boolean
Global switch4 As Boolean
Global switch5 As Boolean
Global switch6 As Boolean
Global switch7 As Boolean
Global switch8 As Boolean

Global klick10 As Variant     'Hilfsvariabel 10-er Schritt betätigt
Global klick1 As Variant      'Hilfsvariabel 1-er Schritt betätigt

Global speed As Variant       'Eingabe der Geschwindigkeit
Global stopp As Variant

Global sollpos0help1, sollpos0help2 As Boolean  'Hilfsvariabel Eingabe der Position 0
Global sollpos1help1, sollpos1help2 As Boolean  'Hilfsvariabel Eingabe der Position 1
Global sollpos2help1, sollpos2help2 As Boolean  'Hilfsvariabel Eingabe der Position 2
Global sollpos3help1, sollpos3help2 As Boolean  'Hilfsvariabel Eingabe der Position 3
Global sollpos4help1, sollpos4help2 As Boolean  'Hilfsvariabel Eingabe der Position 4
```

```
Global sollpos5help1, sollpos5help2 As Boolean  'Hilfsvariabel Eingabe der Position 5

Global hilfs4 As Variant, hilfs5 As Variant, hilfs6 As Variant       'Hilfsvar iabel Eingabe der Geschwindigkeit

Global inputzähler As Integer    'Zähler der Zeichen, die über COM1 gekommen
Global speedh1, speedh2
Global kommando As Variant       'Befehl (Kommando)
Global OrakelObjekt As Object
Global Startbutton As Boolean    'Star tbutton gedrückt Abfrage

Global flag_Position_erreicht As Boolean  'Soll Position erreicht >> Positionier Kommando mit Rückmeldung
Global flag_move As Boolean               'Achsen in Bewegung
Global flag_box_ein As Boolean            'I/O Box eingeschaltet
Global demoend_gedrückt As Boolean        'Demo 3 Walzen beendet

Global ProgName As String              'Pfad von ProSys
Global ProjektName As String           'Name des SPS - Projektes
Global NameSPS As String               'Name des SPS - Projektes für die Message - Box

Global flag_SPS_Wahl As Boolean        'Ansteuerungsart SPS
Global flag_Winkel As Boolean          'Übergabe der Werte als Winkel

Global rückgabe As Variant             'Rückgabe - Wert der COM1 - Schnittstelle
Global digout As Variant               'Rückgabe - Switch I/O TestBox, keine DDE - Variabel
Global flagcmdtext                     'Flag Ist - Text Übergabe

Global b1 As Integer   'Hilfsvariabel bit gesetzt -> rückgesetzt
Global b2 As Integer
Global b3 As Integer
Global b4 As Integer
Global b5 As Integer
Global b6 As Integer
Global b7 As Integer
Global b8 As Integer

Global pos0real  'Hilfsvariabel für die Umrechnung Bytes -> Winkel
Global pos1real
Global pos2real
Global pos3real
Global pos4real
```

A2 Quellcode – Start Formular

```
'Christoph Hyllus Roboteransteuerung 2002 Version 1.0
'Roboter ROB3 Ansteuerung über serielle Schnittstelle mit Low-Level-Protokoll
'Die Bauderate am Roboter wird automatisch nach einem vorangegangenem Reset ( am Roboter, Rückseite des  Gehäuses )
'eingestellt

Option Explicit

Private Sub Form_Load()

Exit Sub

'alt >> ProSys starten falls noch nicht gestartet

Dim ProgName As String
  On Error Resume Next
  AppActivate "PROSYS"
If Not Err = 0 Then
     ProgName = "D:\PROGRAMME\PROSYS\PROSYS.EXE D:\1test\dde.pro" ' nur Programmstart, kein Projekt
     Shell ProgName, 1
End If

End Sub

Private Sub cmdOK_Click()       'Abfrage der  Verbindungsart >> Message Box

Dim Mldg As String
Dim Stil As Variant
Dim Titel As Variant
Dim Antwort As Variant

frmStart.Visible = False        'Verstecke Startformular

Mldg = "Um eine Verbindung zur SPS herzustellen bestätigen Sie mit >>OK " & Chr$(10) & "wählen Sie >> Nein um den
Roboter ohne SPS  - Verbindung anzusteuern "
Stil = vbYesNo + vbQuestion + vbDefaultButton1
    'Schaltflächen definieren.
Titel = "Wie wollen Sie den Roboter ansteuern ?"   ' Titel definieren.
Antwort = MsgBox(Mldg, Stil, Titel) ' Hilfe, Ktxt)   ' Meldung anzeigen.

If Antwort = vbYes Then   'Benutzer hat "Ja" gewählt >> Ansteuerung über SPS .

   flag_SPS_Wahl = True
   DialogProSys.Show

ElseIf Antwort = vbNo Then 'Benutzer hat "Nein" gewählt >> Ansteuerung nur über Visual Basics

   flag_SPS_Wahl = False
   frmhaupt.Show
```

```
End If

    Unload frmStart 'entlade Startformular

End Sub

Private Sub cmdEnde_Click()     'Programm beenden
    End
    Unload frmStart
End Sub
```

A3 Quellcode – Dialog – ProSys – Formular

```
'Christoph Hyllus Roboteransteuerung 2002 Version 1.0
'Roboter ROB3 Ansteuerung über serielle Schnittstelle mit Low-Level-Protokoll
'Die Bauderate am Roboter wird automatisch nach einem vorangegangenem Reset ( am Roboter, Rückseite des Gehäuses )
'eingestellt

'DialogProSys - Formular >> Funktion Automatische Start von ProSys

Option Explicit

Private Sub cmdEnde_Click()
End
End Sub

Private Sub cmdProSysgestartet_Click()

DialogSPS.Show
Unload DialogProSys

End Sub

Private Sub Form_Load()

lblInfo.Caption = "Bitte geben Sie den Pfad des SPS -Programms. z.B C: \PROGRAMME\PROSYS\PROSYS.EXE
Das Programm wird automatisch gestartet"

'Vorbelegung des Textfeldes

 txtProgName = "D: \PROGRAMME\PROSYS\PROSYS.EXE"  'Pfad FH
'txtProgName = "C: \PROGRAMME\PROSYS\PROSYS.EXE"  'Pfad home

End Sub

Private Sub cmdOK_Click()

   ProgName = txtProg Name.Text            'Übergabe des Textes
   On Error Resume Next                    'wiederhole solange kein Fehler
   AppActivate "PROSYS"
   If Not Err = 0 Then                     'angegebene Pfad korrekt und ProSys gestartet
```

```
    'ProgName = "C:\PROGRAMME\PROSYS\PROSYS.EXE"
    'ProgName = "PROSYS|D:\DDE.pro" ' nur Programmstart, kein Projekt"
    Shell ProgName, 1
    lblInfo.Caption = "ProSys ist gestartet"
    Unload DialogProSys
    DialogSPS.Show
  End If

    DialogSPS.Show
    Unload DialogProSys

'lblInfo.Caption = "Bitte geben Sie den Pfad des SPS -Programms.   Das Programm wird automatisch gestartet"

End Sub
```

A4 Quellcode – Dialog – SPS - Formular

```
'Christoph Hyllus Roboteransteuerung 2002 Version 1.0
'Roboter ROB3 Ansteuerung über serielle Schnittstelle mit Low-Level-Protokoll
'Die Bauderate am Roboter wird automatisch nach einem vorangegangenem Reset ( am Roboter, Rückseite des Gehäuses )
'eingestellt

'DialogSPS - Formular >> Funktion Übergabe des Namen SPS - Programms an DDE - Routine

Option Explicit

Private Sub cmdEnde_Click()   'Programm beenden

  End

End Sub

Private Sub Form_Load() 'Vorbelegung für den Namen des SPS - Programms

lblSPSProgramm.Caption = "Bitte geben Sie den Namen des SPS -Programms        für die Ansteuerung des Roboters
und bestätigen Sie mit OK"
txtSPSProgName.Text = "D: \tagscha.pro"
'txtSPSProgName.Text = "D: \ROB5.pro"

End Sub

Private Sub cmdOK_Click()   'Übergabe des Namen SPS - Programms

NameSPS = txtSPSProgName.Text
  Unload DialogSPS          'entlade Formular
  frmhaupt.Show             'gehe zur Hauptformular

End Sub
```

A5 Quellcode – Hauptformular

```
'Christoph Hyllus Roboteransteuerung 2002 Version 1.0
'Roboter ROB3 Ansteuerung über serielle Schnittstelle mit Low-Level-Protokoll
'Die Bauderate am Roboter wird automatisch nach einem vorangegangenem Reset ( am Roboter, Rückseite des Gehäuses )
'eingestellt

'Haupt-Formular >> Funktion Ansteuerung & Darstellung der Roboterpositionen

        'Const >> Konstanten (dürfen im Programm nicht mehr verändert werden)
Const ETX = &H3                 'Kommandodatensatz abschließen
Const SPACE = &H20              'Anfang des Kommunikationsablaufs


Private Declare Sub Sleep Lib "kernel32" (ByVal _
    dwMilliSeconds As Long)     'Timer Deklaration >> http://www.activevb.de/vb/index.html

Private Sub cmdbit1_Click()     'Bit setzen bzw. rücksetzen

  b1 = b1 + 1
  MSComm1.Output = Chr$(&H13)   'Komandowort zum Setzen eines Bits am digitalen Port (LED)
  MSComm1.Output = Chr$(1)      ' >> Bit Nummer binär kodiert
  MSComm1.Output = Chr$(ETX)
If b1 = 2 Then
  cmdbit1.BackColor = &H80FF80  'bit gesetzt
  b1 = 0
End If

If b1 = 1 Then
  cmdbit1.BackColor = &HC0C0C0  'bit zurückgesetzt
End If

End Sub

Private Sub cmdbit2_Click()     'Bit setzen bzw. rücksetzen
  b2 = b2 + 1
MSComm1.Output = Chr$(&H13)     'Komandowort zum Setzen eines Bits am digitalen Port (LED)
MSComm1.Output = Chr$(2)        ' >> Bit Nummer binär kodiert
MSComm1.Output = Chr$(ETX)
If b2 = 2 Then
  cmdbit2.BackColor = &H80FF80  'bit gesetzt
  b2 = 0
End If

If b2 = 1 Then
  cmdbit2.BackColor = &HC0C0C0  'bit zurückgesetzt
End If

End Sub

Private Sub cmdbit3_Click()     'Bit setzen bzw. rücksetzen
```

```
   b3 = b3 + 1
MSComm1.Output = Chr$(&H13)        'Komandowort zum Setzen eines Bits am digitalen Port (LED)
MSComm1.Output = Chr$(4)           ' >> Bit Nummer binär kodiert
MSComm1.Output = Chr$(ETX)
If b3 = 2 Then
   cmdbit3.BackColor = &H80FF80       'bit gesetzt
   b3 = 0
End If

If b3 = 1 Then
   cmdbit3.BackColor = &HC0C0C0       'bit zurückgesetzt
End If

End Sub

Private Sub cmdbit4_Click()        'Bit setzen bzw. rücksetzen
   b4 = b4 + 1
MSComm1.Output = Chr$(&H13)        'Komandowort zum Setzen eines Bits am digitalen Port (LED)
MSComm1.Output = Chr$(8)           ' >> Bit Nummer binär kodiert
MSComm1.Output = Chr$(ETX)
If b4 = 2 Then
   cmdbit4.BackColor = &H80FF80       'bit gesetzt
   b4 = 0
End If

If b4 = 1 Then
   cmdbit4.BackColor = &HC0C0C0       'bit zurückgesetzt
End If

End Sub

Private Sub cmdbit5_Click()        'Bit setzen bzw. rücksetzen
   b5 = b5 + 1
MSComm1.Output = Chr$(&H13)        'Komandowort zum Setzen eines Bits am digitalen Port (LED)
MSComm1.Output = Chr$(16)          ' >> Bit Nummer binär kodiert
MSComm1.Output = Chr$(ETX)
If b5 = 2 Then
   cmdbit5.BackColor = &H80FF80       'bit gesetzt
   b5 = 0
End If

If b5 = 1 Then
   cmdbit5.BackColor = &HC0C0C0       'bit zurückgesetzt
End If

End Sub

Private Sub cmdbit6_Click()        'Bit setzen bzw. rücksetzen
   b6 = b6 + 1
MSComm1.Output = Chr$(&H13)        'Komandowort zum Setzen eines Bits am digitalen Port (LED)
MSComm1.Output = Chr$(32)          ' >> Bit Nummer binär kodiert
MSComm1.Output = Chr$(ETX)
```

```
If b6 = 2 Then
   cmdbit6.BackColor = &H80FF80        'bit gesetzt
   b6 = 0
End If

If b6 = 1 Then
   cmdbit6.BackColor = &HC0C0C0        'bit zurückgesetzt
End If

End Sub

Private Sub cmdbit7_Click()         'Bit setzen bzw. rücksetzen
   b7 = b7 + 1
MSComm1.Output = Chr$(&H13)         'Komandowort zum Setzen eines Bits am digitalen Port (LED)
MSComm1.Output = Chr$(64)           ' >> Bit Nummer binär kodiert
MSComm1.Output = Chr$(ETX)
If b7 = 2 Then
   cmdbit7.BackColor = &H80FF80        'bit gesetzt
   b7 = 0
End If

If b7 = 1 Then
   cmdbit7.BackColor = &HC0C0C0        'bit zurückgesetzt
End If

End Sub

Private Sub cmdbit8_Click()         'Bit setzen bzw. rücksetzen
   b8 = b8 + 1
MSComm1.Output = Chr$(&H13)         'Komandowort zum Setzen eines Bits am digitalen Port (LED)
MSComm1.Output = Chr$(128)          ' >> Bit Nummer
MSComm1.Output = Chr$(ETX)
If b8 = 2 Then
   cmdbit8.BackColor = &H80FF80        'bit gesetzt
   b8 = 0
End If

If b8 = 1 Then
   cmdbit8.BackColor = &HC0C0C0        'bit zurückgesetzt
End If

End Sub

Private Sub cmdboxaus_Click()          'I/O Box ausschalten

flag_box_ein = False    'rücksetze Timer Abfrage - I/O Box

MSComm1.Output = Chr$(&H10) 'Kommandowort zum Setzen bzw. Rücksetzen der Bits am digitalem Port
MSComm1.Output = Chr$(255)  'hier alle Bits rücksetzen(255 -> 0)
MSComm1.Output = Chr$(ETX)

'I/O Box auf B&B zurücksetzen
```

```
cmdbit1.BackColor = &HC0C0C0        'bit zurückgesetzt
cmdbit2.BackColor = &HC0C0C0        'bit zurückgesetzt
cmdbit3.BackColor = &HC0C0C0        'bit zurückgesetzt
cmdbit4.BackColor = &HC0C0C0        'bit zurückgesetzt
cmdbit5.BackColor = &HC0C0C0        'bit zurückgesetzt
cmdbit6.BackColor = &HC0C0C0        'bit zurückgesetzt
cmdbit7.BackColor = &HC0C0C0        'bit zurückgesetzt
cmdbit8.BackColor = &HC0C0C0        'bit zurückgesetzt

lblswitch1.BackColor = &HC0C0C0     'Farbe ändern >> grau
lblswitch2.BackColor = &HC0C0C0     'Farbe ändern >> grau
lblswitch3.BackColor = &HC0C0C0     'Farbe ändern >> grau
lblswitch4.BackColor = &HC0C0C0     'Farbe ändern >> grau
lblswitch5.BackColor = &HC0C0C0     'Farbe ändern >> grau
lblswitch6.BackColor = &HC0C0C0     'Farbe ändern >> grau
lblswitch7.BackColor= &HC0C0C0      'Farbe ändern >> grau
lblswitch8.BackColor = &HC0C0C0     'Farbe ändern >> grau

End Sub

Private Sub cmdboxein_Click()          'I/O Box einschalten

flag_box_ein = True

End Sub

Private Sub cmdDDE_beenden_Click()    'DDE - Verbindung beenden
flag_SPS_Wahl = False

   DDE_Schließen
   No_DDE_Oberfläche

cmdDDE_beenden.BackColor = &HFF&             'Button aktiv >> rot
cmdDDE_Verbindung.BackColor = &HE0E0E0      'Button inaktiv >> grau

End Sub

Private Sub cmdDDE_Verbindung_Click() 'DDE - Verbindung starten
flag_SPS_Wahl = True

   DDE_Verbindung

End Sub

Private Sub cmdDemo3Walzen_Click()  'Demo 3 Walzen starten

Dim n As Integer   'Hilfsvariabel für For - Schleife
demoend_gedrückt = False

'Positionen und Geschw. für 3 Walzen Demo
```

```
pos1 = Array(4, 127, 127, 127, 127, 127, 127, 0)     'Vorgabe der Werte in Felder, keine Zeiger in VB
pos2 = Array(4, 144, 159, 115, 107, 236, 0)
pos3 = Array(4, 144, 210, 135, 65, 236, 0)
'Walzen greifen

pos4 = Array(4, 144, 210, 135, 65, 236, 200)
pos5 = Array(4, 89, 210, 105, 77, 236, 200)
pos6 = Array(4, 89, 210, 85, 88, 236, 200)
pos7 = Array(4, 86, 237, 98, 69, 236, 200)
'Walze ablegen

pos8 = Array(4, 86, 237, 98, 69, 236, 0)
pos9 = Array(4, 176, 155, 69, 233, 0)
pos10 = Array(4, 144, 159, 115, 107, 236, 0)

pos11 = Array(4, 142, 231, 109, 73, 236, 200)
pos12 = Array(4, 142, 202, 126, 73, 236, 200)
pos13 = Array(4, 84, 176, 159, 65, 236, 200)
pos14 = Array(4, 88, 191, 145, 97, 236, 200)
pos15 = Array(4, 88, 191, 145, 97, 236, 0)

pos16 = Array(5, 253, 210, 127, 45, 242, 140)
pos17 = Array(5, 252, 104, 216, 42, 242, 140)
pos18 = Array(5, 151, 104, 216, 42, 242, 140)
pos19 = Array(3, 151, 200, 159, 33, 242, 138)
pos20 = Array(5, 151, 200, 159, 33, 242, 8)
pos21 = Array(5, 151, 200, 159, 33, 242, 8)
pos22 = Array(5, 151, 120, 198, 42, 242, 0)
pos23 = Array(5, 251, 120, 198, 42, 242, 0)
pos24 = Array(5, 253, 212, 122, 44, 242, 0)
pos25 = Array(5, 253, 248, 71, 54, 242, 0)
pos26 = Array(5, 253, 248, 71, 54, 242, 140)
pos27 = Array(5, 252, 104, 216, 42, 242, 140)
pos28 = Array(5, 151, 104, 216, 42, 242, 140)
pos29 = Array(3, 152, 170, 184, 36, 242, 140)
pos30 = Array(3, 152, 172, 185, 35, 242, 140)
pos31 = Array(5, 152, 172, 185, 35, 242, 0)
pos32 = Array(5, 152, 53, 239, 64, 242, 0)

   MSComm1.Output = Chr$(&H61)                    'Kommando Motor - Regelung einschalten
   MSComm1.Output = Chr$(ETX)

If demoend_gedrückt = True Then                'wenn EndeButton gedrückt
Exit Sub

Else
End If

For n = 0 To 5
Move_Rob                                       ' Achsen in Bewegung
   MSComm1.Output = Chr$(120 + n)              '120 + Achsennummer, hier
   MSComm1.Output = Chr$(pos1(n + 1))          'Position
```

```
   MSComm1.Output = Chr$(pos1(0))              'Geschwindigkeit
   MSComm1.Output = Chr$(ETX)
Next n

Sleep (1000)

'While flag_Position_erreicht = False

'Wend

If demoend_gedrückt = True Then              'wenn EndeButton gedrückt
   Exit Sub
Else
End If

For n = 0 To 5
Move_Rob                                ' Achsen in Bewegung
   MSComm1.Output = Chr$(120 + n)           '120 + Achsennummer, hier
   MSComm1.Output = Chr$(pos2(n + 1))         'Position
   MSComm1.Output = Chr$(pos2(0))           'Geschwindigkeit
   MSComm1.Output = Chr$(ETX)
Next n

Sleep (1000)

'While flag_Position_erreicht = False

'Wend

If demoend_gedrückt = True Then              'wenn EndeButton gedrückt
   Exit Sub
Else
End If

For n = 0 To 5
Move_Rob                                ' Achsen in Bewegung
   MSComm1.Output = Chr$(120 + n)           '120 + Achsennummer, hier
   MSComm1.Output = Chr$(pos3(n + 1))         'Position
   MSComm1.Output = Chr$(pos3(0))           'Geschwindigkeit
   MSComm1.Output = Chr$(ETX)
Next n

Sleep (1000)

'While flag_Position_erreicht = False

'Wend

If demoend_gedrückt = True Then              'wenn EndeButton gedrückt
   Exit Sub
Else
End If
```

```
For n = 0 To 5
Move_Rob                                ' Achsen in Bewegung
   MSComm1.Output = Chr$(120 + n)              '120 + Achsennummer, hier
   MSComm1.Output = Chr$(pos4(n + 1))          'Position
   MSComm1.Output = Chr$(pos4(0))              'Geschwindigkeit
   MSComm1.Output = Chr$(ETX)
Next n

Sleep (1000)

'While flag_Position_erreicht = False

'Wend

If demoend_gedrückt = True Then              'wenn EndeButton gedrückt
   Exit Sub
Else
End If

For n = 0 To 5
Move_Rob                                ' Achsen in Bewegung
   MSComm1.Output = Chr$(120 + n)              '120 + Achsennummer, hier
   MSComm1.Output = Chr$(pos5(n + 1))          'Position
   MSComm1.Output = Chr$(pos5(0))              'Geschwindigkeit
   MSComm1.Output = Chr$(ETX)
Next n

Sleep (1000)

'While flag_Position_erreicht = False

'Wend

If demoend_gedrückt = True Then              'wenn EndeButton gedrückt
   Exit Sub
Else
End If

For n = 0 To 5
Move_Rob                                ' Achsen in Bewegung
   MSComm1.Output = Chr$(120 + n)              '120 + Achsennummer, hier
   MSComm1.Output = Chr$(pos6(n + 1))          'Position
   MSComm1.Output = Chr$(pos6(0))              'Geschwindigkeit
   MSComm1.Output = Chr$(ETX)
Next n

Sleep (1000)

'While flag_Position_erreicht = False

'Wend
```

```
For n = 0 To 5
Move_Rob                                ' Achsen in Bewegung
   MSComm1.Output = Chr$(120 + n)           '120 + Achsennummer, hier
   MSComm1.Output = Chr$(pos7(n + 1))         'Position
   MSComm1.Output = Chr$(pos7(0))           'Geschwindigkeit
   MSComm1.Output = Chr$(ETX)
Next n

Sleep (1000)

'While flag_Position_erreicht = False

'Wend

If demoend_gedrückt = True Then             'wenn EndeButton gedrückt
   Exit Sub
Else
End If

For n = 0 To 5
Move_Rob                                ' Achsen in Bewegung
   MSComm1.Output = Chr$(120 + n)           '120 + Achsennummer, hier
   MSComm1.Output = Chr$(pos8(n + 1))         'Position
   MSComm1.Output = Chr$(pos8(0))           'Geschwindigkeit
   MSComm1.Output = Chr$(ETX)
Next n

Sleep (1000)

'While flag_Position_erreicht = False

'Wend

If demoend_gedrückt = True Then             'wenn EndeButton gedrückt

Else
End If

For n = 0 To 5
Move_Rob                                ' Achsen in Bewegung
   MSComm1.Output = Chr$(120 + n)           '120 + Achsennummer, hier
   MSComm1.Output = Chr$(pos9(n + 1))         'Position
   MSComm1.Output = Chr$(pos9(0))           'Geschwindigkeit
   MSComm1.Output = Chr$(ETX)
Next n

Sleep (1000)

'While flag_Position_erreicht = False

'Wend
```

```
For n = 0 To 5
Move_Rob                                ' Achsen in Bewegung
   MSComm1.Output = Chr$(120 + n)           '120 + Achsennummer, hier
   MSComm1.Output = Chr$(pos10(n + 1))        'Position
   MSComm1.Output = Chr$(pos10(0))            'Geschwindigkeit
   MSComm1.Output = Chr$(ETX)
Next n

Sleep (1000)

'While flag_Position_erreicht = False

'Wend

If demoend_gedrückt = True Then              'wenn EndeButton gedrückt
   Exit Sub
Else
End If

For n = 0 To 5
Move_Rob                                ' Achsen in Bewegung
   MSComm1.Output = Chr$(120 + n)           '120 + Achsennummer, hier
   MSComm1.Output = Chr$(pos11(n + 1))        'Position
   MSComm1.Output = Chr$(pos11(0))            'Geschwindigkeit
   MSComm1.Output = Chr$(ETX)
Next n

Sleep (1000)

'While flag_Position_erreicht = False

'Wend

For n = 0 To 5
Move_Rob                                ' Achsen in Bewegung
   MSComm1.Output = Chr$(120 + n)           '120 + Achsennummer, hier
   MSComm1.Output = Chr$(pos12(n + 1))        'Position
   MSComm1.Output = Chr$(pos12(0))            'Geschwindigkeit
   MSComm1.Output = Chr$(ETX)
Next n

Sleep (1000)

'While flag_Position_erreicht = False

'Wend

If demoend_gedrückt = True Then              'wenn EndeButton gedrückt
   Exit Sub
Else
End If
```

```
For n = 0 To 5
Move_Rob                                ' Achsen in Bewegung
   MSComm1.Output = Chr$(120 + n)              '120 + Achsennummer, hier
   MSComm1.Output = Chr$(pos13(n + 1))         'Position
   MSComm1.Output = Chr$(pos13(0))             'Geschwindigkeit
   MSComm1.Output = Chr$(ETX)
Next n

Sleep (1000)

'While flag_Position_erreicht = False

'Wend

For n = 0 To 5
Move_Rob                                ' Achsen in Bewegung
   MSComm1.Output = Chr$(120 + n)              '120 + Achsennummer, hier
   MSComm1.Output = Chr$(pos14(n + 1))         'Position
   MSComm1.Output = Chr$(pos14(0))             'Geschwindigkeit
   MSComm1.Output = Chr$(ETX)
Next n
Exit Sub
Sleep (1000)

'While flag_Position_erreicht = False

'Wend

If demoend_gedrückt = True Then              'wenn EndeButton gedrückt
   Exit Sub
Else
End If

For n = 0 To 5
Move_Rob                                ' Achsen in Bewegung
   MSComm1.Output = Chr$(120 + n)              '120 + Achsennummer, hier
   MSComm1.Output = Chr$(pos15(n + 1))         'Position
   MSComm1.Output = Chr$(pos15(0))             'Geschwindigkeit
   MSComm1.Output = Chr$(ETX)
Next n

Sleep (1000)

'While flag_Position_erreicht = False

'Wend

If demoend_gedrückt = True Then              'wenn EndeButton gedrückt
   Exit Sub
Else
End If
```

```
For n = 0 To 5
Move_Rob                                ' Achsen in Bewegung
   MSComm1.Output = Chr$(120 + n)              '120 + Achsennummer, hier
   MSComm1.Output = Chr$(pos16(n + 1))         'Position
   MSComm1.Output = Chr$(pos16(0))             'Geschwindigkeit
   MSComm1.Output = Chr$(ETX)
Next n

Sleep (1000)

'While flag_Position_erreicht = False

'Wend

For n = 0 To 5
Move_Rob                                ' Achsen in Bewegung
   MSComm1.Output = Chr$(120 + n)              '120 + Achsennummer, hier
   MSComm1.Output = Chr$(pos17(n + 1))         'Position
   MSComm1.Output = Chr$(pos17(0))             'Geschwindigkeit
   MSComm1.Output = Chr$(ETX)
Next n

Sleep (1000)

'While flag_Position_erreicht = False

'Wend

If demoend_gedrückt = True Then              'wenn EndeButton gedrückt
   Exit Sub
Else
End If

For n = 0 To 5
Move_Rob                                ' Achsen in Bewegung
   MSComm1.Output = Chr$(120 + n)              '120 + Achsennummer, hier
   MSComm1.Output = Chr$(pos18(n + 1))         'Position
   MSComm1.Output = Chr$(pos18(0))             'Geschwindigkeit
   MSComm1.Output = Chr$(ETX)
Next n

Sleep (1000)

'While flag_Position_erreicht = False

'Wend

If demoend_gedrückt = True Then              'wenn EndeButton gedrückt
   Exit Sub
Else
End If
```

```
For n = 0 To 5
Move_Rob                                ' Achsen in Bewegung
   MSComm1.Output = Chr$(120 + n)           '120 + Achsennummer, hier
   MSComm1.Output = Chr$(pos19(n + 1))        'Position
   MSComm1.Output = Chr$(pos19(0))            'Geschwindigkeit
   MSComm1.Output = Chr$(ETX)
Next n

Sleep (1000)

'While flag_Position_erreicht = False

'Wend

If demoend_gedrückt = True Then             'wenn EndeButton gedrückt
   Exit Sub
Else
End If

For n = 0 To 5
Move_Rob                                ' Achsen in Bewegung
   MSComm1.Output = Chr$(120 + n)           '120 + Achsennummer, hier
   MSComm1.Output = Chr$(pos20(n + 1))        'Position
   MSComm1.Output = Chr$(pos20(0))            'Geschwindigkeit
   MSComm1.Output = Chr$(ETX)
Next n

Sleep (1000)

'While flag_Position_erreicht = False

'Wend

If demoend_gedrückt = True Then             'wenn EndeButton gedrückt
   Exit Sub
Else
End If

For n = 0 To 5
Move_Rob                                ' Achsen in Bewegung
   MSComm1.Output = Chr$(120 + n)           '120 + Achsennummer, hier
   MSComm1.Output = Chr$(pos21(n + 1))        'Position
   MSComm1.Output = Chr$(pos21(0))            'Geschwindigkeit
   MSComm1.Output = Chr$(ETX)
Next n

Sleep (1000)

'While flag_Position_erreicht = False

'Wend
```

```
If demoend_gedrückt = True Then              'wenn EndeButton gedrückt
   Exit Sub
Else
End If

For n = 0 To 5
Move_Rob                                ' Achsen in Bewegung
   MSComm1.Output = Chr$(120 + n)            '120 + Achsennummer, hier
   MSComm1.Output = Chr$(pos22(n + 1))         'Position
   MSComm1.Output = Chr$(pos22(0))             'Geschwindigkeit
   MSComm1.Output = Chr$(ETX)
Next n

Sleep (1000)

'While flag_Position_erreicht = False

'Wend

If demoend_gedrückt = True Then              'wenn EndeButton gedrückt
   Exit Sub
Else
End If

For n = 0 To 5
Move_Rob                                ' Achsen in Bewegung
   MSComm1.Output = Chr$(120 + n)            '120 + Achsennummer, hier
   MSComm1.Output = Chr$(pos23(n + 1))         'Position
   MSComm1.Output = Chr$(pos23(0))             'Geschwindigkeit
   MSComm1.Output = Chr$(ETX)
Next n

Sleep (1000)

'While flag_Position_erreicht = False

'Wend

If demoend_gedrückt = True Then              'wenn EndeButton gedrückt
   Exit Sub
Else
End If

For n = 0 To 5
Move_Rob                                ' Achsen in Bewegung
   MSComm1.Output = Chr$(120 + n)            '120 + Achsennummer, hier
   MSComm1.Output = Chr$(pos24(n + 1))         'Position
   MSComm1.Output = Chr$(pos24(0))             'Geschwindigkeit
   MSComm1.Output = Chr$(ETX)
Next n
```

```
Sleep (1000)

'While flag_Position_erreicht = False

'Wend

If demoend_gedrückt = True Then              'wenn EndeButton gedrückt
   Exit Sub
Else
End If

For n = 0 To 5
Move_Rob                                ' Achsen in Bewegung
   MSComm1.Output = Chr$(120 + n)            '120 + Achsennummer, hier
   MSComm1.Output = Chr$(pos25(n + 1))          'Position
   MSComm1.Output = Chr$(pos25(0))              'Geschwindigkeit
   MSComm1.Output = Chr$(ETX)
Next n

Sleep (1000)

'While flag_Position_erreicht = False

'Wend

If demoend_gedrückt = True Then              'wenn EndeButton gedrückt
   Exit Sub
Else
End If

For n = 0 To 5
Move_Rob                                ' Achsen in Bewegung
   MSComm1.Output = Chr$(120 + n)            '120 + Achsennummer, hier
   MSComm1.Output = Chr$(pos26(n + 1))          'Position
   MSComm1.Output = Chr$(pos26(0))              'Geschwindigkeit
   MSComm1.Output = Chr$(ETX)
Next n

Sleep (1000)

'While flag_Position_erreicht = False

'Wend

If demoend_gedrückt = True Then              'wenn EndeButton gedrückt
   Exit Sub
Else
End If

For n = 0 To 5
Move_Rob                                ' Achsen in Bewegung
   MSComm1.Output = Chr$(120 + n)            '120 + Achsennummer, hier
```

```
    MSComm1.Output = Chr$(pos27(n + 1))          'Position
    MSComm1.Output = Chr$(pos27(0))              'Geschwindigkeit
    MSComm1.Output = Chr$(ETX)
Next n

Sleep (1000)

'While flag_Position_erreicht = False

'Wend

If demoend_gedrückt = True Then              'wenn EndeButton gedrückt
    Exit Sub
Else
End If

For n = 0 To 5
Move_Rob                                  ' Achsen in Bewegung
    MSComm1.Output = Chr$(120 + n)           '120 + Achsennummer, hier
    MSComm1.Output = Chr$(pos28(n + 1))          'Position
    MSComm1.Output = Chr$(pos28(0))              'Geschwindigkeit
    MSComm1.Output = Chr$(ETX)
Next n

Sleep (1000)

'While flag_Position_erreicht = False

'Wend

If demoend_gedrückt = True Then              'wenn EndeButton gedrückt
    Exit Sub
Else
End If

For n = 0 To 5
Move_Rob                                  ' Achsen in Bewegung
    MSComm1.Output = Chr$(120 + n)           '120 + Achsennummer, hier
    MSComm1.Output = Chr$(pos29(n + 1))          'Position
    MSComm1.Output = Chr$(pos29(0))              'Geschwindigkeit
    MSComm1.Output = Chr$(ETX)
Next n

Sleep (1000)

'While flag_Position_erreicht = False

'Wend

If demoend_gedrückt = True Then              'wenn EndeButton gedrückt
    Exit Sub
Else
```

```
End If

For n = 0 To 5
Move_Rob                                ' Achsen in Bewegung
   MSComm1.Output = Chr$(120 + n)              '120 + Achsennummer, hier
   MSComm1.Output = Chr$(pos30(n + 1))         'Position
   MSComm1.Output = Chr$(pos30(0))             'Geschwindigkeit
   MSComm1.Output = Chr$(ETX)
Next n

Sleep (1000)

End Sub

Private Sub cmddemoende_Click()     'Demo 3 Walzen beenden

demoend_gedrückt = True

End Sub

Private Sub cmdspeed7_Click()       'Übergabe der Geschwindigkeit mit den Buttons, umgekehrter Reihenfolge beachten !
speed = 0                           '7 = max. Geschw.
End Sub

Private Sub cmdspeed6_Click()       'Übergabe der Geschwindigkeit mit den Buttons
speed = 1
End Sub

Private Sub cmdspeed5_Click()       'Übergabe der Geschwindigkeit mit den Buttons
speed = 2
End Sub

Private Sub cmdspeed4_Click()       'Übergabe der Geschwindigkeit mit den Buttons
speed = 3
End Sub

Private Sub cmdspeed3_Click()       'Übergabe der Geschwindigkeit mit den Buttons
speed = 4
End Sub

Private Sub cmdspeed2_Click()       'Übergabe der Geschwindigkeit mit den Buttons
speed = 5
End Sub

Private Sub cmdspeed1_Click()       'Übergabe der Geschwindigkeit mit den Buttons
speed = 6
End Sub

Private Sub cmdübergabe_ist_txt_Click()        'STOP Übergabe der Ist - Werte an TextSollPosition

   txtsollpos0.Text = pos0
   txtsollpos1.Text = pos1
```

```
    txtsollpos2.Text = pos2
    txtsollpos3.Text = pos3
    txtsollpos4.Text = pos4
    txtsollpos5.Text = pos5
End Sub

Private Sub cmdWinkel_Click()
        'Abfrage nach Art der Sollwerte >> Grad oder Byte 0..255

Mldg = "Wollen die Sollwerte der Achsen als Grad angeben dann bestätigen Sie mit >>OK " & Chr$(10) & "Wollen Sie die Werte
Byte(0..255) angeben, dann wählen Sie >> Nein "
Stil = vbYesNo + vbQuestion + vbDefaultButton1
    'Schaltflächen definieren.
Titel = "Wie wollen Sie den Roboter ansteuern ?"   ' Titel definieren.
Antwort = MsgBox(Mldg, Stil, Titel) ' Hilfe, Ktxt)   ' Meldung anzeigen.

If Antwort = vbYes Then   'Benutzer hat "Ja" gewählt >> Sollwerte als Grad


    flag_Winkel = True      'Übergabe der Sollwerte als Grad

scrollpos1.Min = -80    'scrollbars für Winkel initialisieren
scrollpos1.Max = 80
scrollpos1.Value = pos0

scrollpos2.Min = -30
scrollpos2.Max = 70
scrollpos2.Value = pos1

scrollpos3.Min = -100
scrollpos3.Max = 0
scrollpos3.Value = pos2

scrollpos4.Min = -100
scrollpos4.Max = 100
scrollpos4.Value = pos3

scrollpos5.Min = -100
scrollpos5.Max = 100
scrollpos5.Value = pos4

scrollgreifer.Min = 0
scrollgreifer.Max = 255
scrollgreifer.Value = pos5

    lblBereich1.Caption = "80     0     -80"
    lblBereich2.Caption = "70     0       "
    lblBereich3.Caption = "       0   -100"
    lblBereich4.Caption = "100    0   -100"
    lblBereich5.Caption = "100    0   -100"
    lblBereich6.Caption = "0.........255"
    cmdWinkel.Caption = "Winkel"
```

```
ElseIf Antwort = vbNo Then 'Benutzer hat "Nein" gewählt >> Sollwerte als Byte

    flag_Winkel = False     'Übergabe der Sollwerte als Byte 0...255

    'Istwerte_vom_Rob (rückgabe)
    'Sleep (1000)

scrollpos1.Min = 0  'scrollbars für Byte initialisieren
scrollpos1.Max = 255
scrollpos1.Value = pos0

scrollpos2.Min = 0
scrollpos2.Max = 255
scrollpos2.Value = pos1

scrollpos3.Min = 0
scrollpos3.Max = 255
scrollpos3.Value = pos2

scrollpos4.Min = 0
scrollpos4.Max = 255
scrollpos4.Value = pos3

scrollpos5.Min = 0
scrollpos5.Max = 255
scrollpos5.Value = pos4

scrollgreifer.Min = 0
scrollgreifer.Max = 255
scrollgreifer.Value = pos5

    lblBereich1.Caption = " 0.....255"
    lblBereich2.Caption = " 0.....255"
    lblBereich3.Caption = " 0.....255"
    lblBereich4.Caption = " 0.....255"
    lblBereich5.Caption = " 0.....255"
    lblBereich6.Caption = " 0.....255"
    cmdWinkel.Caption = "Byte"

    cmdDDE_beenden.BackColor = &HFF&            'Button aktiv >> rot
    cmdDDE_Verbindung.BackColor = &HE0E0E0      'Button inaktiv >> grau

End If

End Sub

Private Sub Form_Initialize()        'Wird vor Load ausgeführt

cmdstart.Caption = "Bitte Roboter einschalten und zurücksetzen" & Chr$(10) & " RESET- Taster an der Rückseite des Sockels"

timeIstAbfrage.Interval = 100          'Abtastzeit für Abfrage der Ist - Positionen vorbelegen
```

```
timeInOut.Interval = 350              'Abtastzeit für Abfrage der I/O - Box vorbelegen
flag_Position_erreicht = False
flag_box_ein = False                  'I/O Box beim starten ausgeschaltet

b1 = 1              'Vorbelegung, für Setzen/Rücksetzen der Farben der Buttons (LED-OUT)
b2 = 1
b3 = 1
b4 = 1
b5 = 1
b6 = 1
b7 = 1
b8 = 1

txtsollpos0.MaxLength = 3             'maximale Anzahl der Zeichen
txtsollpos1.MaxLength = 3
txtsollpos2.MaxLength = 4
txtsollpos3.MaxLength = 4
txtsollpos4.MaxLength = 4
txtsollpos5.MaxLength = 3

lblswitch1.BackColor = &HC0C0C0     'Vorbelegung der Farbe der Schalter und LED's >> grau
lblswitch2.BackColor = &HC0C0C0
lblswitch3.BackColor = &HC0C0C0
lblswitch4.BackColor = &HC0C0C0
lblswitch5.BackColor = &HC0C0C0
lblswitch6.BackColor = &HC0C0C0
lblswitch7.BackColor = &HC0C0C0
lblswitch8.BackColor = &HC0C0C0

lblAchse1ist.BackColor = &HFFFFFF   'Vorbelegung der Farbe der Ist - Position - Labels >> weiss
lblAchse2ist.BackColor = &HFFFFFF
lblAchse3ist.BackColor = &HFFFFFF
lblAchse4ist.BackColor = &HFFFFFF
lblAchse5ist.BackColor = &HFFFFFF
lblGreiferist.BackColor = &HFFFFFF

speed = 3                           'Grundgeschw. vorgeben

scrollspeed.Min = 6
scrollspeed.Max = 0
scrollspeed.Value = speed

klick1 = True                       'Vorgabe 1-er Schritt
klick10 = False
Schritt1.BackColor = &HFF&              '1-Schritt BackColor = rot
   MSComm1.CommPort = 1             'Initialisierung der seriellen Schnittstelle (MSComm)
   MSComm1.Settings = "9600,N,8,1"
   MSComm1.InputMode = comInputModeText  'Textdaten in einem Wert von Typ Text -> Binary
   MSComm1.InputLen = 0
   MSComm1.PortOpen = True          'serielle Schnittstelle öffnen

   MSComm1.Output = Chr$(SPACE)             'Kommmando START der Kommunikation-> SPACE = 20 hex
```

```
                                    'Anfang der Kommunikation kein ETX - Zeichen schicken !!!

    startabrage = True

lblPos_erreicht.Visible = False ' Hilfsbezeichnungsfeld unsichtbar machen

lblLED1.Visible = False        ' Hilfslabel für DDE >> LED1-8 I/O Box unsichtbar machen
lblLED2.Visible = False
lblLED3.Visible = False
lblLED4.Visible = False
lblLED5.Visible = False
lblLED6.Visible = False
lblLED7.Visible = False
lblLED8.Visible = False

End Sub

Private Sub Form_Load()                'Initialisierung

Dim Mldg As String  'Variabel - Deklaration für MessageBox
Dim Stil As Variant
Dim Titel As Variant
Dim Antwort As Variant


If flag_SPS_Wahl = True Then  'SPS  - Ansteuerung

   DDE_Verbindung  ' DDE - Verbindung aufbauen & Fehlerbehandlung
   Exit Sub        ' Sub Ende

Else              'keine SPS  - Ansteuerung

End If
         'Abfrage nach Art der Sollwerte >> Grad oder Byte 0..255

    'Meldung definieren.
Mldg = "Wollen die Sollwerte der Achsen als Grad angeben dann bestätigen Sie mit >> OK " & Chr$(10) & "Wollen Sie die
Werte Byte(0..255) angeben, dann wählen Sie >> Nein "
Stil = vbYesNo + vbQuestion + vbDefaultButton1
    'Schaltflächen definieren.
Titel = "Wie wollen Sie den Roboter ansteuern ?"   ' Titel definieren.
Antwort = MsgBox(Mldg, Stil, Titel) ' Hilfe, Ktxt)   ' Meldung anzeigen.

If Antwort = vbYes Then   'Benutzer hat "Ja" gewählt >> Sollwerte als Grad

   flag_Winkel = True      'Übergabe der Sollwerte als Grad

scrollpos1.Min = -80
scrollpos1.Max = 80
scrollpos1.Value = pos0

scrollpos2.Min = -30
```

```
scrollpos2.Max = 70
scrollpos2.Value = pos1


scrollpos3.Min = -100
scrollpos3.Max = 0
scrollpos3.Value = pos2

scrollpos4.Min = -100
scrollpos4.Max = 100
scrollpos4.Value = pos3

scrollpos5.Min = -100
scrollpos5.Max = 100
scrollpos5.Value = pos4

scrollgreifer.Min = 0
scrollgreifer.Max = 255
scrollgreifer.Value = pos5


ElseIf Antwort = vbNo Then 'Benutzer hat "Nein" gewählt >> Sollwerte als Grad

    flag_Winkel = False     'Übergabe der Sollwerte als Byte 0...255

scrollpos1.Min = 0
scrollpos1.Max = 255
scrollpos1.Value = pos0

scrollpos2.Min = 0
scrollpos2.Max = 255
scrollpos2.Value = pos1

scrollpos3.Min = 0
scrollpos3.Max = 255
scrollpos3.Value = pos2

scrollpos4.Min = 0
scrollpos4.Max = 255
scrollpos4.Value = pos3

scrollpos5.Min = 0
scrollpos5.Max = 255
scrollpos5.Value = pos4

scrollgreifer.Min = 0
scrollgreifer.Max = 255
scrollgreifer.Value = pos5

    cmdDDE_beenden.BackColor = &HFF&            'Button aktiv >> rot
    cmdDDE_Verbindung.BackColor = &HE0E0E0      'Button inaktiv >> grau
```

```
End If

  No_DDE_Oberfläche
  cmdDDE_beenden.BackColor = &HFF&            'Button aktiv >> rot
  cmdDDE_Verbindung.BackColor = &HE0E0E0      'Button inaktiv >> grau

End Sub

Private Sub cmdStart_Click()              'Roboter Starten

   Startbutton = True

MSComm1.Output = Chr$(&H10) 'Kommandowort zum Setzen bzw. Rücksetzen der Bits am digitalem Port
MSComm1.Output = Chr$(255)  'hier alle Bits rücksetzen(255 -> 0)
MSComm1.Output = Chr$(ETX)

txtsollpos0.Text = pos0     'Übergabe der Ist - Werte an Text
txtsollpos1.Text = pos1
txtsollpos2.Text = pos2
txtsollpos3.Text = pos3
txtsollpos4.Text = pos4
txtsollpos5.Text = pos5

   cmdstart.Visible = False        ' Commandschaltfläche Start nach dem Anklicken unsichtbar machen

End Sub

Private Sub cmdstop_Click() 'Roboter STOP >> Übergabe der momentanen Istposition als Sollpos.

   txtsollpos0.Text = pos0
   txtsollpos1.Text = pos1
   txtsollpos2.Text = pos2
   txtsollpos3.Text = pos3
   txtsollpos4.Text = pos4
   txtsollpos5.Text = pos5

   sollpos0 = pos0     'Sollwert - Vorgabe des Roboters = Ist -Wert >> STOP
   sollpos1 = pos1
   sollpos2 = pos2
   sollpos3 = pos3
   sollpos4 = pos4
   sollpos5 = pos5

   sendecom0 sollpos0, 0, speed
   sendecom1 sollpos1, 1, speed
   sendecom2 sollpos2, 2, speed
   sendecom3 sollpos3, 3, speed
   sendecom4 sollpos4, 4, speed
   sendecom5 sollpos5, 5, speed

End Sub
```

```
Private Sub lblAchse1ist_Change()    'Rückgabe der Ist  - Positionen an die SPS

If flag_SPS_Wahl = False Then            'keine DDE - Verbindung
   Exit Sub
Else
   On Error GoTo DDE_Fehler
   lblAchse1ist.LinkPoke              'DDE - Verbindung Wertrückgabe
End If

Exit Sub

DDE_Fehler: DDE_Verbindung                'Verbindung aufbauen

End Sub

Private Sub lblAchse1ist_LinkClose()

If flag_SPS_Wahl = False Then            'keine DDE - Verbindung

   Exit Sub    'Verbindung absichtlich abgebaut >> keine Reaktion auf Ereignis

Else
   DDE_Verbindung                'Verbindung aufbauen
End If

End Sub

Private Sub lblAchse2ist_Change()    'Rückgabe der Ist  - Positionen an die SPS

If flag_SPS_Wahl = False Then            'keine DDE - Verbindung

   Exit Sub

Else
   On Error GoTo DDE_Fehler
   lblAchse2ist.LinkPoke              'DDE - Verbindung Wertrückgabe

End If

Exit Sub

DDE_Fehler: DDE_Verbindung                'Verbindung aufbauen

End Sub

Private Sub lblAchse2ist_LinkClose()

If flag_SPS_Wahl = False Then            'keine DDE - Verbindung

   Exit Sub    'Verbindung absichtlich abgebaut >> keine Reaktion auf Ereignis

Else
```

```
    DDE_Verbindung              'Verbindung aufbauen
End If

End Sub

Private Sub lblAchse3ist_Change()    'Rückgabe der Ist  - Positionen an die SPS

 If flag_SPS_Wahl = False Then          'keine DDE - Verbindung

    Exit Sub
Else
    On Error GoTo DDE_Fehler
    lblAchse3ist.LinkPoke            'DDE - Verbindung Wertrückgabe
End If

Exit Sub

DDE_Fehler: DDE_Verbindung              'Verbindung aufbauen

End Sub

Private Sub lblAchse3ist_LinkClose()

If flag_SPS_Wahl = False Then          'keine DDE - Verbindung

    Exit Sub    'Verbindung absichtlich abgebaut >> keine Reaktion auf Ereignis

Else
    DDE_Verbindung              'Verbindung aufbauen
End If

End Sub

Private Sub lblAchse4ist_Change()    'Rückgabe der Ist  - Positionen an die SPS

If flag_SPS_Wahl = False Then          'keine DDE - Verbindung

    Exit Sub
Else
    On Error GoTo DDE_Fehler
    lblAchse4ist.LinkPoke            'DDE - Verbindung Wertrückgabe
End If

Exit Sub

DDE_Fehler: DDE_Verbindung              'Verbindung aufbauen

End Sub

Private Sub lblAchse4ist_LinkClose()

If flag_SPS_Wahl = False Then          'keine DDE - Verbindung
```

```
Exit Sub    'Verbindung absichtlich abgebaut >> keine Reaktion auf Ereignis

Else
   DDE_Verbindung                'Verbindung aufbauen
End If

End Sub

Private Sub lblAchse5ist_Change()    'Rückgabe der Ist  - Positionen an die SPS

 If flag_SPS_Wahl = False Then           'keine DDE - Verbindung

   Exit Sub
Else
   On Error GoTo DDE_Fehler
   lblAchse5ist.LinkPoke              'DDE - Verbindung Wertrückgabe
End If

Exit Sub

DDE_Fehler: DDE_Verbindung                'Verbindung aufbauen

End Sub

Private Sub lblAchse5ist_LinkClose()

If flag_SPS_Wahl = False Then           'keine DDE - Verbindung

   Exit Sub    'Verbindung absi chtlich abgebaut >> keine Reaktion auf Ereignis

Else
   DDE_Verbindung                'Verbindung aufbauen
End If

End Sub

Private Sub lblGreiferist_Change()    'Rückgabe der Ist - Positionen an die SPS

 If flag_SPS_Wahl = False Then           'keine DDE - Verbindung

  Exit Sub
Else
   On Error GoTo DDE_Fehler
   lblGreiferist.LinkPoke              'DDE - Verbindung Wertrückgabe
End If

Exit Sub

DDE_Fehler: DDE_Verbindung                'Verbindung aufbauen

End Sub
```

```
Private Sub lblGreiferist_LinkClose()

If flag_SPS_Wahl = False Then            'keine DDE - Verbindung

   Exit Sub    'Verbindung absichtlich abgebaut >> keine Reaktion auf Ereignis

Else
   DDE_Verbindung                'Verbindung aufbauen
End If

End Sub

Private Sub lblLED1_Change()        ' I/O Box LED Übergabe

If lblLED1.Caption = 1 Then
   MSComm1.Output = Chr$(&H13)        'Komandowort zum Setzen eines Bits am digitalen Port (LED)
   MSComm1.Output = Chr$(1)           ' >> Bit Nummer binär kodiert
   MSComm1.Output = Chr$(ETX)
      cmdbit1.BackColor = &H80FF80        'bit gesetzt
Else
   MSComm1.Output = Chr$(&H12)        'Komandowort zum Rücksetzen eines Bits am digitalen Port (LED)
   MSComm1.Output = Chr$(255 - 1)          ' >> Bit Nummer binär kodiert
   MSComm1.Output = Chr$(ETX)
      cmdbit1.BackColor = &HC0C0C0        'bit zurückgesetzt
End If

End Sub

Private Sub lblLED1_LinkClose()    'DDE Kanal geschlossen

If flag_SPS_Wahl = False Then            'keine DDE - Verbindung

   Exit Sub    'Verbindung absichtlich abgebaut >> keine Reaktion auf Ereignis

Else
   DDE_Verbindung  'Verbindung aufbauen
End If

End Sub

Private Sub lblLED2_Change()        ' I/O Box LED Übergabe

If lblLED2.Caption = 1 Then
   MSComm1.Output = Chr$(&H13)        'Komandowort zum Setzen eines Bits am digitalen Port (LED)
   MSComm1.Output = Chr$(2)           ' >> Bit Nummer binär kodiert
   MSComm1.Output = Chr$(ETX)
      cmdbit2.BackColor = &H80FF80        'bit gesetzt
Else
   MSComm1.Output = Chr$(&H12)        'Komandowort zum Rücksetzen eines Bits am digitalen Port (LED)
   MSComm1.Output = Chr$(255 - 2)          ' >> Bit Nummer binär kodiert
   MSComm1.Output = Chr$(ETX)
```

```
    cmdbit2.BackColor = &HC0C0C0        'bit zurückgesetzt
End If

End Sub

Private Sub lblLED2_LinkClose()

If flag_SPS_Wahl = False Then           'keine DDE - Verbindung

  Exit Sub       'Verbindung absichtlich abgebaut >> keine Reaktion auf Ereignis

Else
  DDE_Verbindung                'Verbindung aufbauen
End If

End Sub
Private Sub lblLED3_Change()       ' I/O Box LED Übergabe

If lblLED3.Caption = 1 Then
  MSComm1.Output = Chr$(&H13)       'Komandowort zum Setzen eines Bits am digitalen Port (LED)
  MSComm1.Output = Chr$(4)          ' >> Bit Nummer binär kodiert
  MSComm1.Output = Chr$(ETX)
    cmdbit2.BackColor = &H80FF80       'bit gesetzt
Else
  MSComm1.Output = Chr$(&H12)       'Komandowort zum Rücksetzen eines Bits am digitalen Port (LED)
  MSComm1.Output = Chr$(255 - 4)         ' >> Bit Nummer binär kodiert
  MSComm1.Output = Chr$(ETX)
    cmdbit3.BackColor = &HC0C0C0       'bit zurückgesetzt
End If

End Sub

Private Sub lblLED3_LinkClose()

If flag_SPS_Wahl = False Then           'keine DDE - Verbindung

  Exit Sub    'Verbindung absichtlich abgebaut >> keine Reaktion auf Ereignis

Else
  DDE_Verbindung                'Verbindung aufbauen
End If

End Sub

Private Sub lblLED4_Change()       ' I/O Box LED Übergabe

If lblLED4.Caption = 1 Then
  MSComm1.Output = Chr$(&H13)       'Komandowort zum Setzen eines Bits am digitalen Port (LED)
  MSComm1.Output = Chr$(8)          ' >> Bit Nummer binär kodiert
  MSComm1.Output = Chr$(ETX)
    cmdbit4.BackColor = &H80FF80       'bit gesetzt
Else
```

```
   MSComm1.Output = Chr$(&H12)        'Komandowort zum Rücksetzen eines Bits am digitalen Port (LED)
   MSComm1.Output = Chr$(255 - 8)       ' >> Bit Nummer binär kodiert
   MSComm1.Output = Chr$(ETX)
      cmdbit4.BackColor = &HC0C0C0       'bit zurückgesetzt
End If

End Sub

Private Sub lblLED4_LinkClose()

If flag_SPS_Wahl = False Then          'keine DDE - Verbindung

   Exit Sub    'Verbindung absichtlich abgebaut >> keine Reaktion auf Ereignis

Else
   DDE_Verbindung                'Verbindung aufbauen
End If

End Sub

Private Sub lblLED5_Change()       ' I/O Box LED Übergabe

If lblLED5.Caption = 1 Then
   MSComm1.Output = Chr$(&H13)        'Komandowort zum Setzen eines Bits am digitalen Port (LED)
   MSComm1.Output = Chr$(16)          ' >> Bit Nummer binär kodiert
   MSComm1.Output = Chr$(ETX)
      cmdbit5.BackColor = &H80FF80       'bit gesetzt
Else
   MSComm1.Output = Chr$(&H12)        'Komandowort zum Rücksetzen eines Bits am digitalen Port (LED)
   MSComm1.Output = Chr$(255 - 16)       ' >> Bit Nummer binär kodiert
   MSComm1.Output = Chr$(ETX)
      cmdbit5.BackColor = &HC0C0C0       'bit zurückgesetzt
End If

End Sub

Private Sub lblLED5_LinkClose()

If flag_SPS_Wahl = False Then          'keine DDE - Verbindung

   Exit Sub    'Verbindung absichtlich abgebaut >> keine Reaktion auf Ereignis

Else
   DDE_Verbindung                'Verbindung aufbauen
End If

End Sub

Private Sub lblLED6_Change()       ' I/O Box LED Übergabe

If lblLED6.Caption = 1 Then
   MSComm1.Output = Chr$(&H13)        'Komandowort zum Setzen eines Bits am digitalen Port (LED)
```

```
   MSComm1.Output = Chr$(32)          ' >> Bit Nummer binär kodiert
   MSComm1.Output = Chr$(ETX)
      cmdbit6.BackColor = &H80FF80       'bit gesetzt
Else
   MSComm1.Output = Chr$(&H12)        'Komandowort zum Rücksetzen eines Bits am digitalen Port (LED)
   MSComm1.Output = Chr$(255 - 32)         ' >> Bit Nummer binär kodiert
   MSComm1.Output = Chr$(ETX)
      cmdbit6.BackColor = &HC0C0C0       'bit zurückgesetzt
End If

End Sub

Private Sub lblLED6_LinkClose()

If flag_SPS_Wahl = False Then           'keine DDE - Verbindung

   Exit Sub    'Verbindung absichtlich abgebaut >> keine Reaktion auf Ereignis

Else
   DDE_Verbindung                'Verbindung aufbauen
End If

End Sub

Private Sub lblLED7_Change()       ' I/O Box LED Übergabe

If lblLED7.Caption = 1 Then
   MSComm1.Output = Chr$(&H13)        'Komandowort zum Setzen eines Bits am digitalen Port (LED)
   MSComm 1.Output = Chr$(64)          ' >> Bit Nummer binär kodiert
   MSComm1.Output = Chr$(ETX)
      cmdbit7.BackColor = &H80FF80       'bit gesetzt
Else
   MSComm1.Output = Chr$(&H12)        'Komandowort zum Rücksetzen eines Bits am digitalen Port (LED)
   MSComm1.Output = Chr$(255 - 64)         ' >> Bit Nummer binär kodiert
   MSComm1.Output = Chr$(ETX)
      cmdbit7.BackColor = &HC0C0C0       'bit zurückgesetzt
End If

End Sub

Private Sub lblLED7_LinkClose()

If flag_SPS_Wahl = False Then           'keine DDE - Verbindung

   Exit Sub    'Verbindung absichtlich abgebaut >> keine Reaktion auf Ereignis

Else
   DDE_Verbindung                'Verbindung aufbauen
End If

End Sub
```

```
Private Sub lblLED8_Change()        ' I/O Box LED Übergabe

If lblLED8.Caption = 1 Then
   MSComm1.Output = Chr$(&H13)        'Komandowort zum Setzen eines Bits am digitalen Port (LED)
   MSComm1.Output = Chr$(128)         ' >> Bit Nummer binär kodiert
   MSComm1.Output = Chr$(ETX)
      cmdbit8.BackColor = &H80FF80        'bit gesetzt
Else
   MSComm1.Output = Chr$(&H12)        'Komandowort zum Rücksetzen eines Bits am digitalen Port (LED)
   MSComm1.Output = Chr$(255 - 128)         ' >> Bit Nummer binär kodiert
   MSComm1.Output = Chr$(ETX)
      cmdbit8.BackColor = &HC0C0C0        'bit zurückgesetzt
End If

End Sub

Private Sub lblLED8_LinkClose()

If flag_SPS_Wahl = False Then           'keine DDE - Verbindung

   Exit Sub    'Verbindung absichtlich abgebaut >> keine Reaktion auf Ereignis

Else
   DDE_Verbindung                'Verbindung aufbauen
End If

End Sub

Private Sub lblswitch1_Change()       'Rückgabe der Schalter - Positionen

If flag_SPS_Wahl = False Then           'keine DDE - Verbindung

   Exit Sub
Else
   On Error GoTo DDE_Fehler
   lblswitch1.LinkPoke              'DDE - Verbindung Wertrückgabe

End If

Exit Sub

DDE_Fehler:     DDE_Verbindung                'Verbindung aufbauen

End Sub

Private Sub lblswitch1_LinkClose()

If flag_SPS_Wahl = False Then           'keine DDE - Verbindung

   Exit Sub    'Verbindung absichtlich abgebaut >> keine Reaktion auf Ereignis

Else
```

```
    DDE_Verbindung              'Verbindung aufbauen
End If

End Sub

Private Sub lblswitch2_Change()       'Rückgabe der Schalter - Positionen

If flag_SPS_Wahl = False Then          'keine DDE - Verbindung

    Exit Sub
Else
    On Error GoTo DDE_Fehler
    lblswitch2.LinkPoke             'DDE - Verbindung Wertrückgabe
End If

Exit Sub

DDE_Fehler: DDE_Verbindung              'Verbindung aufbauen

End Sub
Private Sub lblswitch2_LinkClose()

If flag_SPS_Wahl = False Then          'keine DDE - Verbindung

    Exit Sub    'Verbindung absichtlich abgebaut >> keine Reaktion auf Ereignis

Else
    DDE_Verbindung              'Verbindung aufbauen
End If

End Sub

Private Sub lblswitch3_Change()       'Rückgabe der Schalter - Positionen

If flag_SPS_Wahl = False Then          'keine DDE - Verbindung

    Exit Sub
Else
    On Error GoTo DDE_Fehler
    lblswitch3.LinkPoke             'DDE - Verbindung Wertrückgabe
End If

Exit Sub

DDE_Fehler: DDE_Verbindung              'Verbindung aufbauen

End Sub

Private Sub lblswitch3_LinkClose()

If flag_SPS_Wahl = False Then          'keine DDE - Verbindung
```

```
    Exit Sub    'Verbindung absichtlich abgebaut >> keine Reaktion auf Ereignis

Else
    DDE_Verbindung              'Verbindung aufbauen
End If

End Sub

Private Sub lblswitch4_Change()     'Rückgabe der Schalter - Positionen

If flag_SPS_Wahl = False Then       'keine DDE - Verbindung

    Exit Sub
Else
    On Error GoTo DDE_Fehler
    lblswitch4.LinkPoke             'DDE - Verbindung Wertrückgabe
End If

Exit Sub

DDE_Fehler: DDE_Verbindung              'Verbindung aufbauen

End Sub

Private Sub lblswitch4_LinkClose()

If flag_SPS_Wahl = False Then       'keine DDE - Verbindung

Exit Sub    'Verbindung absichtlich abgebaut >> keine Reaktion auf Ereignis

Else
    DDE_Verbindung              'Verbindung aufbauen
End If

End Sub

Private Sub lblswitch5_Change()     'Rückgabe der Schalter - Positionen

If flag_SPS_Wahl = False Then       'keine DDE - Verbindung

    Exit Sub
Else
    On Error GoTo DDE_Fehler
    lblswitch5.LinkPoke             'DDE - Verbindung Wertrückgabe
End If

Exit Sub

DDE_Fehler: DDE_Verbindung              'Verbindung aufbauen

End Sub
```

```
Private Sub lblswitch5_LinkClose()

If flag_SPS_Wahl = False Then           'keine DDE - Verbindung

Exit Sub    'Verbindung absichtlich abgebaut >> keine Reaktion auf Ereignis

Else
   DDE_Verbindung                'Verbindung aufbauen
End If

End Sub

Private Sub lblswitch6_Change()       'Rückgabe der Schalter - Positionen

If flag_SPS_Wahl = False Then           'keine DDE - Verbindung

   Exit Sub
Else
   On Error GoTo DDE_Fehler
   lblswitch6.LinkPoke              'DDE - Verbindung Wertrückgabe
End If

Exit Sub

DDE_Fehler: DDE_Verbindung                'Verbindung aufbauen

End Sub

Private Sub lblswitch6_LinkClose()

If flag_SPS_Wahl = False Then           'keine DDE - Verbindung


Exit Sub    'Verbindung absichtlich abgebaut >> keine Reaktion auf Ereignis

Else
   DDE_Verbindung                'Verbindung aufbauen
End If

End Sub

Private Sub lblswitch7_Change()       'Rückgabe der Schalter - Positionen

If flag_SPS_Wahl = False Then           'keine DDE - Verbindung

   Exit Sub
Else
   On Error GoTo DDE_Fehler
   lblswitch7.LinkPoke              'DDE - Verbindung Wertrückgabe
End If

Exit Sub
```

```
DDE_Fehler: DDE_Verbindung                'Verbindung aufbauen

End Sub

Private Sub lblswitch7_LinkClose()

If flag_SPS_Wahl = False Then          'keine DDE - Verbindung


Exit Sub    'Verbindung absichtlich abgebaut >> keine Reaktion auf Ereignis


Else
   DDE_Verbindung                'Verbindung aufbauen
End If

End Sub

Private Sub lblswitch8_Change()       'Rückgabe der Schalter - Positionen

If flag_SPS_Wahl = False Then          'keine DDE - Verbindung

   Exit Sub
Else
   On Error GoTo DDE_Fehler
   lblswitch8.LinkPoke              'DDE - Verbindung Wertrückgabe
End If

Exit Sub

DDE_Fehler: DDE_Verbindung                'Verbindung aufbauen

End Sub

Private Sub lblswitch8_LinkClose()

If flag_SPS_Wahl = False Then          'keine DDE - Verbindung


Exit Sub    'Verbindung absichtlich abgebaut >> keine Reaktion auf Ereignis

Else
   DDE_Verbindung                'Verbindung aufbauen
End If

End Sub

Private Sub MSComm1_OnComm()   'Serielle Schnittstelle
                        'Erklärung s.Eigenschaften RThreshold = 1
                       'nach jedem Zeichen ein OnComm- Ereignis
Dim zahl As Variant
```

```
Dim inputzähler As Integer     'Anzahl von COM1 (MSComm) zurückgegebenen Zeichen


Select Case MSComm1.CommEvent
  Case comBreak               'Break-Zeichen eingetroffen

  Case comCTSTO               'CTS Timeout

  Case comerrxparity          'Paritätsfehler

  Case comEvReceive           'Zeichen angekommen

     rückgabe = MSComm1.Input    'Ist-Position Übergabe vom Roboter
     inputzähler = Len(rückgabe) 'Anzahl von MSComm zurückgegebenen Zeichen
     flag = False

If Startbutton = True Then  'Anfang

If inputzähler = 8 Then     'Rückgabe der Istwerte
   Istwerte_vom_Rob (rückgabe)
End If

If (inputzähler = 3 And flagtimeIstAbfrage = False) Then       'Rückgabe der I/O Testbox
   digout = rückgabe
   Digout_vom_Rob (digout)  'Unterprogramm Auswertung der dig. Outputs
End If

If inputzähler = 2 Then              'Rückgabe Sollposition erreicht
   flag_Position_erreicht = True
   lblPos_erreicht.Caption = 1
   On Error GoTo DDE_Fehler          ' DDE - Kommunikationsfehler >> Sprung zur Marke DDE_Fehler
   lblPos_erreicht.LinkPoke          'Rückgabe an die SPS
   Soll_Pos_erreicht                 'Soll Position erreicht
Else
    lblPos_erreicht.Caption = 0
    On Error GoTo DDE_Fehler          ' DDE - Kommunikationsfehler >> Sprung zur Marke DDE_Fehler
    lblPos_erreicht.LinkPoke
End If

End If       'Ende der If Startbutton

End Select

DDE_Fehler: DDE_Verbindung

End Sub

Private Sub cmdEnde_Click()                    'Programm beenden

    End                     'Projekt beenden

End Sub
```

```
Private Sub timeInOut_Timer()                'Abfrage der I/O Testbox des ROB 3

If Startbutton = True And flag_box_ein = True Then 'nur wenn Startbutton und I/O Box eingeschaltet
   MSComm1.Output = Chr$(&H54)               'Kommando Abfrage der digitalen Sensoren
   MSComm1.Output = Chr$(ETX)
End If

End Sub

Private Sub lblsollpos0_DDE_Change() ' Ansteuerung über SPS
   sollpos0 = lblsollpos0_DDE.Caption
   achse = 0
   speed = lblspeed_DDE.Caption
sendecom0 sollpos0, achse, speed     ' Datenübergabe
End Sub

Private Sub lblsollpos0_DDE_LinkClose()

If flag_SPS_Wahl = False Then            'keine DDE - Verbindung

   Exit Sub    'Verbindung absichtlich abgebaut >> keine Reaktion auf Ereignis

Else
   DDE_Verbindung  'Verbindung aufbauen
End If

End Sub

Private Sub lblsollpos1_DDE_Change()    ' Ansteuerung über SPS

   sollpos1 = lblsollpos1_DDE.Caption
   achse = 1
   speed = lblspeed_DDE.Caption

sendecom1 sollpos1, achse, speed                ' Datenübergabe

End Sub

Private Sub lblsollpos1_DDE_LinkClose()

If flag_SPS_Wahl = False Then            'keine DDE - Verbindung

   Exit Sub    'Verbindung absichtlich abgebaut >> keine Reaktion auf Ereignis

Else
   DDE_Verbindung                'Verbindung aufbauen
End If

End Sub

Private Sub lblsollpos2_DDE_Change()    ' Ansteuerung über SPS
```

```
    sollpos2 = lblsollpos2_DDE.Caption
    achse = 2
    speed = lblspeed_DDE.Caption

sendecom2 sollpos2, achse, speed                ' Datenübergabe

End Sub

Private Sub lblsollpos2_DDE_LinkClose()

If flag_SPS_Wahl = False Then           'keine DDE - Verbindung

    Exit Sub    'Verbindung absichtlich abgebaut >> keine Reaktion auf Ereignis

Else
    DDE_Verbindung              'Verbindung aufbauen
End If

End Sub

Private Sub lblsollpos3_DDE_Change()    ' Ansteuerung über SPS

    sollpos3 = lblsollpos3_DDE
    achse = 3
    speed = lblspeed_DDE.Caption

sendecom3 sollpos3, achse, speed                ' Datenübergabe

End Sub

Private Sub lblsollpos3_DDE_LinkClose()

If flag_SPS_Wahl = False Then           'keine DDE - Verbindung

    Exit Sub    'Verbindung absichtlich abgebaut >> keine Reaktion auf Ereignis

Else
    DDE_Verbindung              'Verbindung aufbauen
End If

End Sub

Private Sub lblsollpos4_DDE_Change()    ' Ansteuerung über SPS

    sollpos4 = lblsollpos4_DDE.Caption
    achse = 4
    speed = lblspeed_DDE.Caption

sendecom4 sollpos4, achse, speed                ' Datenübergabe

End Sub
```

```
Private Sub lblsollpos4_DDE_LinkClose()

If flag_SPS_Wahl = False Then           'keine DDE - Verbindung

   Exit Sub    'Verbindung absichtlich abgebaut >> keine Reaktion auf Ereignis

Else
   DDE_Verbindung                'Verbindung aufbauen
End If

End Sub

Private Sub lblsollpos5_DDE_Change()    ' Ansteuerung über SPS

   sollpos5 = lblsollpos5_DDE.Caption
   achse = 5
   speed = lblspeed_DDE.Caption

sendecom5 sollpos5, achse, speed                ' Datenübergabe

End Sub

Private Sub lblsollpos5_DDE_LinkClose()

If flag_SPS_Wahl = False Then           'keine DDE - Verbindung

   Exit Sub    'Verbindung absichtlich abgebaut >> keine Reaktion auf Ereignis

Else

   DDE_Verbindung                'Verbindung aufbauen
End If

End Sub

Private Sub Form_Close()              'serielle Schnittstelle schließen
   MSComm1.PortOpen = False
   Startbutton = False
End Sub

Private Sub StartProsys_Click()       'Programm (ProSys)von der VB-Oberfläche starten
Dim Ergebnis
        'Progname = "D:\Chrispro.pro"
        'Ergebnis = Shell("D: \Chrispro.pro", 1)
ProgName = "D:\Programme\ProSys\ProSys.exe"       'Programm starten
Ergebnis = Shell(ProgName, 1)
MsgBox "Das Programm" & ProgName & "  ist schon gestartet"
End Sub

Private Sub timeIstAbfrage_Timer()              'Timer für Ist - Position Abfrage

flagtimeIstAbfrage = True
```

```
If Startbutton = True Then

kommando = &H47                         'Abfrage aller Ist - Positionen
   MSComm1.Output = Chr$(kommando)
   MSComm1.Output = Chr$(ETX)

lblAchse1ist.Caption = pos0             'Übergabe der Ist - Positionen des Roboters
lblAchse2ist.Caption = pos1
lblAchse3ist.Caption = pos2
lblAchse4ist.Caption = pos3
lblAchse5ist.Caption = pos4
lblGreiferist.Caption = pos5

End If

flagtimeIstAbfrage = False

'Istwerte_vom_Rob (rückgabe)

scrollspeed.Value = speed     'Übergabe der Ist-Werte an Bildlaufleisten (scrollbars)
scrollpos1.Value = pos0
scrollpos2.Value = pos1
scrollpos3.Value = pos2
scrollpos4.Value = pos3
scrollpos5.Value = pos4
scrollgreifer.Value = pos5

End Sub

Private Sub txtsollpos0_KeyPress(KeyAscii As Integer)  'Eingabe Soll 0 Tasteneingabe Überprüfung

 Char = Chr(KeyAscii)
 sollpos0help1 = Asc(Char)                              ' Umwandlung ASCII->Integer

If sollpos0help1 > 57 Or sollpos0help1 < 48 Or sollpos0help1 = 45 Then 'Zahlen eingegeben
   sollpos0help2 = True
Else
   sollpos0help2 = False                                'keine Zahlen eingegeben
End If

End Sub

Private Sub txtsollpos0_Change()    ' Eingabe Soll 0, nur wenn Zahlen und/oder "-"

If sollpos0help2 = False And sollpos0help1 <> 8 And Startbutton = False Then '

   txtsollpos0.Text = pos0       ' Wert-Übergabe sonst 2xVal->0

Else
    sollpos0 = Val(txtsollpos0.Text) ' Zeichenfolge als numerische Wert zurück
    txtsollpos0.Text = sollpos0
End If
```

```
End Sub

Private Sub txtsollpos1_Change()        'Eingabe Soll 1

If sollpos1help2 = False And sollpos1help1 <> 8 And Startbutton = False Then
  txtsollpos1.Text = sollpos1                    ' Wert-Übergabe sonst 2xVal->0
Else
   sollpos1 = Val(txtsollpos1.Text)              ' Zeichenfolge als numerische Wert zurück
   txtsollpos1.Text = sollpos1
End If

End Sub

Private Sub txtsollpos1_KeyPress(KeyAscii As Integer)

Char = Chr(KeyAscii)
sollpos1help1 = Asc(Char)                      ' Umwandlung ASCII->Integer
If sollpos1help1 > 57 Or sollpos1help1 < 48 Then                ' keine Zahlen eingegeben
   sollpos1help2 = False
Else
   sollpos1help2 = True                        'Zahlen eingegeben
End If

End Sub

Private Sub txtsollpos2_Change()        'Eingabe Soll 2
If sollpos2help2 = False And sollpos2help1 <> 8 And Startbutton = False Then
  txtsollpos2.Text = sollpos2           ' Wert-Übergabe sonst 2xVal->0
Else
   sollpos2 = Val(txtsollpos2.Text)  ' Zeichenfolge als numerische Wert zurück
   txtsollpos2.Text = sollpos2

End If

End Sub

Private Sub txtsollpos2_KeyPress(KeyAscii As Integer)

  Char = Chr(KeyAscii)
 sollpos2help1 = Asc(Char)                 ' Umwandlung ASCII->Integer
If sollpos2help1 > 57 Or sollpos2help1 < 48 Then                ' keine Zahlen eingegeben
   sollpos2help2 = False
Else
   sollpos2help2 = True                        'Zahlen eingegeben
End If

End Sub

Private Sub txtsollpos3_Change()        'Eingabe Soll 3

If sollpos3help2 = False And sollpos3help1 <> 8 And Startbutton = False Then
```

```
   txtsollpos3.Text = sollpos3                          ' Wert-Übergabe sonst 2xVal->0
Else
    sollpos3 = Val(txtsollpos3.Text)                    ' Zeichenfolge als numerische Wert zurück
    txtsollpos3.Text = sollpos3
End If

End Sub

Private Sub txtsollpos3_KeyPress(KeyAscii As Integer)

 Char = Chr(KeyAscii)
 sollpos3help1 = Asc(Char)                              ' Umwandlung ASCII->Integer
If sollpos3help1 > 57 Or sollpos3help1 < 48 Then                    ' keine Zahlen eingegeben
    sollpos3help2 = False
Else
    sollpos3help2 = True                               'Zahlen eingegeben
End If

End Sub

Private Sub txtsollpos4_Change()          'Eingabe Soll 4
If sollpos4help2 = False And sollpos4help1 <> 8 And Startbutton = False Then
    txtsollpos4.Text = sollpos4                         ' Wert-Übergabe sonst 2xVal->0
Else
    sollpos4 = Val(txtsollpos4.Text)                    ' Zeichenfolge als numerische Wert zurück
    txtsollpos4.Text = sollpos4
End If

End Sub

Private Sub txtsollpos4_KeyPress(KeyAscii As Integer)

 Char = Chr(KeyAscii)
 sollpos4help1 = Asc(Char)                              ' Umwandlung ASCII->Integer
If sollpos4help1 > 57 Or sollpos4help1 < 48 Then                    ' keine Zahlen eingegeben
    sollpos4help2 = False
Else
    sollpos4help2 = True                               'Zahlen eingegeben
End If

End Sub

Private Sub txtsollpos5_Change()          'Eingabe Soll 5

If sollpos5help2 = False And sollpos5help1 <> 8 And Startbutton = False Then
    txtsollpos5.Text = sollpos5                         ' Wert-Übergabe sonst 2xVal->0
Else
    sollpos5 = Val(txtsollpos5.Text)                    ' Zeichenfolge als numerische Wert zurück
    txtsollpos5.Text = sollpos5
End If

End Sub
```

```
Private Sub txtsollpos5_KeyPress(KeyAscii As Integer)

 Char = Chr(KeyAscii)
 sollpos5help1 = Asc(Char)                                    ' Umwandlung ASCII->Integer
If sollpos5help1 > 57 Or sollpos5help1 < 48 Then                          ' keine Zahlen eingegeben
   sollpos5help2 = False
Else
   sollpos5help2 = True                                      'Zahlen eingegeben
End If

End Sub

Private Sub cmdWeitergabePos1_Click()          'Weitergabe der Position an die Steuerung
MSComm1.Output = Chr$(&H61)                    'Kommando Motor - Regelung einschalten
MSComm1.Output = Chr$(ETX)

   txtsollpos0.Text = sollpos0
   txtsollpos1.Text = sollpos1
   txtsollpos2.Text = sollpos2
   txtsollpos3.Text = sollpos3
   txtsollpos4.Text = sollpos4
   txtsollpos5.Text = sollpos5

   sendecom0 sollpos0, 0, speed
   sendecom1 sollpos1, 1, speed
   sendecom2 sollpos2, 2, speed
   sendecom3 sollpos3, 3, speed
   sendecom4 sollpos4, 4, speed
   sendecom5 sollpos5, 5, speed

End Sub

Private Sub Schritt1_Click()          'Schritt mal 1
If Schritt1.Enabled = True Then
   Schritt1.BackColor = &HFF&             ' Schritt *1  aktiv >> rot
   Schritt10.BackColor = &HE0E0E0         ' Schritt *10 nicht aktiv >> grau
End If
   klick1 = True
   klick10 = False
End Sub

Private Sub Schritt10_Click()          'Schritt mal 10
If Schritt10.Enabled = True Then
   Schritt10.BackColor = &HFF&            ' Schritt *10  aktiv >> rot
   Schritt1.BackColor = &HE0E0E0          ' Schritt *1  nicht aktiv >> grau

End If
   klick10 = True
   klick1 = False
End Sub
```

```
Private Sub cmdAchse1minus_Click()                  'Achse 1 -

Dim Mldg As String  'Parameter für Message Box
Dim Stil As String
Dim Titel As String
Dim Antwort As String

If klick10 = True And klick1 = False Then      'Achtung Roboter fährt 10 Schritte ab der jeweiligen Ist - Position
sollpos0 = pos0 - 10                        'd.h. bei langsamen Geschw. keine absolute 10-er Schritte möglich
   Else
sollpos0 = pos0 - 1
  End If

   txtsollpos0 = sollpos0
   sendecom0 sollpos0, 0, speed             'Übergabe der Werte an die serielle Schnittstelle

End Sub
Private Sub cmdAchse1plus_Click(Index As Integer)     'Achse 1 +

Dim Mldg As String  'Parameter für Message Box
Dim Stil As String
Dim Titel As String
Dim Antwort As String

If klick10 = True And klick1 = False Then      'Achtung Roboter fährt 10 Schritte ab der jeweiligen Ist - Position
sollpos0 = pos0 + 10
   Else
sollpos0 = pos0 + 1
  End If

   txtsollpos0 = sollpos0
   sendecom0 sollpos0, 0, speed

End Sub
Private Sub cmdAchse2minus_Click(Index As Integer)     'Achse 2-

Dim Mldg As String  'Parameter für Message Box
Dim Stil As String
Dim Titel As String
Dim Antwort As String

If klick10 = True And klick1 = False Then      'Achtung Roboter fährt 10 Schritte ab der jeweiligen Ist - Position
sollpos1 = pos1 - 10
   Else
sollpos1 = pos1 - 1
  End If

   txtsollpos1 = sollpos1
   sendecom1 sollpos1, 1, speed             'Übergabe der Werte an die serielle Schnittstelle

End Sub
```

```
Private Sub cmdAchse2plus_Click(Index As Integer)     'Achse 2+

Dim Mldg As String  'Parameter für Message Box
Dim Stil As String
Dim Titel As String
Dim Antwort As String

If klick10 = True And klick1 = False Then      'Achtung Roboter fährt 10 Schritte ab der jeweiligen Ist - Position
sollpos1 = pos1 + 10
   Else
sollpos1 = pos1 + 1
  End If

 txtsollpos1 = sollpos1
 sendecom1 sollpos1, 1, speed                'Übergabe der Werte an die serielle Schnittstelle

End Sub

Private Sub cmdAchse3minus_Click(Index As Integer)    'Achse 3-

Dim Mldg As String  'Parameter für Message Box
Dim Stil As String
Dim Titel As String
Dim Antwort As String

If klick10 = True And klick1 = False Then      'Achtung Roboter fährt 10 Schritte ab der jeweiligen Ist - Position
sollpos2 = pos2 - 10
   Else
sollpos2 = pos2 - 1
  End If

   txtsollpos2 = sollpos2
   sendecom2 sollpos2, 2, speed

End Sub

Private Sub cmdAchse3plus_Click(Index As Integer)     'Achse 3+

Dim Mldg As String  'Parameter für Message Box
Dim Stil As String
Dim Titel As String
Dim Antwort As String

If klick10 = True And klick1 = False Then      'Achtung Roboter fährt 10 Schritte ab der jeweiligen Ist - Position
sollpos2 = pos2 + 10
   Else
sollpos2 = pos2 + 1
  End If

    txtsollpos2 = sollpos2
    sendecom2 sollpos2, 2, speed
```

```
End Sub

Private Sub cmdAchse4minus_Click(Index As Integer)     'Achse 4-

Dim Mldg As String  'Parameter für Message Box
Dim Stil As String
Dim Titel As String
Dim Antwort As String

If klick10 = True And klick1 = False Then      'Achtung Roboter fährt 10 Schritte ab der jeweiligen Ist - Position
sollpos3 = pos3 - 10
   Else
sollpos3 = pos3 - 1
  End If

   txtsollpos3 = sollpos3
   sendecom3 sollpos3, 3, speed

End Sub

Private Sub cmdAchse4plus_Click(Index As Integer)      'Achse 4+

Dim Mldg As String  'Parameter für Message Box
Dim Stil As String
Dim Titel As String
Dim Antwort As String

If klick10 = True And klick1 = False Then      'Achtung Roboter fährt 10 Schritte ab der jeweiligen Ist - Position
sollpos3 = pos3 + 10
   Else
sollpos3 = pos3 + 1
  End If

   txtsollpos3 = sollpos3
   sendecom3 sollpos3, 3, speed              'Übergabe der Werte an die serielle Schnittstelle

End Sub

Private Sub cmdAchse5minus_Click(Index As Integer)     'Achse 5-

Dim Mldg As String  'Parameter für Message Box
Dim Stil As String
Dim Titel As String
Dim Antwort As String

If klick10 = True And klick1 = False Then      'Achtung Roboter fährt 10 Schritte ab der jeweiligen Ist - Position
sollpos4 = pos4 - 10
   Else
sollpos4 = pos4 - 1
  End If

   txtsollpos4 = sollpos4
```

```
    sendecom4 sollpos4, 4, speed

End Sub

Private Sub cmdAchse5plus_Click(Index As Integer)      'Achse 5 +

Dim Mldg As String  'Parameter für Message Box
Dim Stil As String
Dim Titel As String
Dim Antwort As String

If klick10 = True And klick1 = False Then      'Achtung Roboter fährt 10 Schritte ab der jeweiligen Ist - Position
sollpos4 = pos4 + 10
   Else
sollpos4 = pos4 + 1
  End If

    txtsollpos4 = sollpos4
    sendecom4 sollpos4, 4, speed

End Sub

Private Sub cmdGreiferzu_Click()                    'Greifer schließen

Dim Mldg As String  'Parameter für Message Box
Dim Stil As String
Dim Titel As String
Dim Antwort As String

If klick10 = True And klick1 = False Then      'Achtung Roboter fährt 10 Schritte ab der jeweiligen Ist - Position
sollpos5 = pos5 + 10
   Else
sollpos5 = pos5 + 1
  End If

    txtsollpos5 = sollpos5
    sendecom5 sollpos5, 5, speed

End Sub

Private Sub cmdGreiferauf_Click()                   'Greifen öffnen

Dim Mldg As String  'Parameter für Message Box
Dim Stil As String
Dim Titel As String
Dim Antwort As String

If klick10 = True And klick1 = False Then      'Achtung Roboter fährt 10 Schritte ab der jeweiligen Ist - Position
sollpos5 = pos5 - 10
   Else
sollpos5 = pos5 - 1
  End If
```

```
    txtsollpos5 = sollpos5
    sendecom5 sollpos5, 5, speed

End Sub

Private Sub cmdposition_Click() 'Alle Achsen mit Geschw. ansteuern

Move_Rob                                ' Achsen in Bewegung

    MSComm1.Output = Chr$(&H7F)                 'Kommando alle Achsen mit Geschw.
For ein = 0 To 5                            '
    MSComm1.Output = Chr$(127)                  'Position Übergabe
Next ein

For sped = 0 To 5                           '
    MSComm1.Output = Chr$(speed)                'Geschw. Übergabe
Next sped

    MSComm1.Output = Chr$(ETX)                  'Kommando abschließen

End Sub

Public Sub sendecom0(sollpos0, achse, speed)  'Unterprogramm Positioniervorgabe mit Geschwindigkeit

Dim Mldg, Stil, Titel, Antwort    'Parameter für Message Box
Dim sollpos0real

If flag_Winkel = True Then
                    'Winkelumrechnung (80...0..-80) Grad in Bytes (0 -255)
If sollpos0 > 0 Then        'Winkel positiv
    sollpos0real = 127 - (sollpos0 * 1.5875)
ElseIf sollpos0 < 0 Then    'Winkel negativ
    sollpos0real = (sollpos0 * (-1))
    sollpos0real = 127 + (sollpos0real * 1.5875)
ElseIf sollpos0 = 0 Then    'Winkel zero
    sollpos0real = 127
End If
    sollpos0real = CInt(sollpos0real)     'Nachkommastellen runden
    sollpos0 = sollpos0real

Else

End If

If (sollpos0 > 255) Or (sollpos0 < 0) Then       'Übergabe der Werte nur im zulässigen Bereich

    sollpos0 = pos0
    txtsollpos0.Text = pos0

If flag_Winkel = True Then
```

```
   Mldg = "Bitte wählen ein andere Position für die Achse1 " & Chr$(10) & "    80...0..-80 Grad"              'Message Box >>>
Warnung
Else
   Mldg = "Bitte wählen ein andere Position für die Achse1 " & Chr$(10) & "    0..255 Grad"               'Message Box >>>
Warnung
End If

Stil = vbOKOnly + vbCritical + vbDefaultButton1
    'Schaltflächen definieren.
Titel = "Positionierungsbereich der Achse1 überschritten"   ' Titel definieren.
Antwort = MsgBox(Mldg, Stil, Titel) ' Meldung anzeigen.

Else                        'Übergabe der Werte nur im zulässigen Bereich

   Move_Rob                              ' Achsen in Bewegung

   MSComm1.Output = Chr$(&H61)                  'Kommando Motor - Regelung einschalten
   MSComm1.Output = Chr$(ETX)

achse = 120 + achse

   MSComm1.Output = Chr$(achse)                 '120 + Achsennummer
   MSComm1.Output = Chr$(sollpos0)
   MSComm1.Output = Chr$(speed)
   MSComm1.Output = Chr$(ETX)
flag = True
End If

End Sub

Public Sub sendecom1(sollpos1, achse, speed)  'Unterprogramm Positioniervorgabe mit Geschwindigkeit
Dim Mldg, Stil, Titel, Antwort    'Parameter für Message Box
Dim sollpos1real

If flag_Winkel = True Then
                    'Winkelumrechnung (+70...0...(-30 möglich) Grad in Bytes (0-255)
If sollpos1 > 0 Then        'Winkel positiv
   sollpos1real = 178 - (sollpos1 * 2.55)
ElseIf sollpos1 < 0 Then    'Winkel negativ
   sollpos1real = (sollpos1 * (-1))
   sollpos1real = 178 + (sollpos1real * 2.55)
ElseIf sollpos1 = 0 Then    'Winkel zero
   sollpos1real = 178
End If
   sollpos1real = CInt(sollpos1real)     'Nachkommastellen runden
   sollpos1 = sollpos1real

Else

End If

If (sollpos1 > 255) Or (sollpos1 < 0) Then  'praktisch 0-255 möglich, hier (>>180)Begrenzung wegen Kollisionsgefahr !!!
```

```
If flag_Winkel = True Then
   Mldg = "Bitte wählen ein andere Position für die Achse2  " & Chr$(10) & "    70...0 Grad" 'Message Box >>> Warnung
Else
   Mldg = "Bitte wählen ein andere Position für die Achse2 " & Chr$(10) & "    0..255"              'Message Box >>> Warnung
End If

Stil = vbOKOnly + vbCritical + vbDefaultButton1
     'Schaltflächen definieren.
Titel = "Positionierungsbereich der Achse2 überschritten (Kollisionsgefahr)"   ' Titel definieren.
Antwort = MsgBox(Mldg, Stil, Titel) ' Meldung anzeigen.
   txtsollpos1.Text = pos1

Else                'Übergabe der Werte nur im zulässigen Bereich

   Move_Rob                              ' Achsen in Bewegung

   MSComm1.Output = Chr$(&H61)                  'Kommando Motor - Regelung einschalten
   MSComm1.Output = Chr$(ETX)

achse = 120 + achse

   MSComm1.Output = Chr$(achse)              '120 + Achsennummer
   MSComm1.Output = Chr$(sollpos1)
   MSComm1.Output = Chr$(speed)
   MSComm1.Output = Chr$(ETX)

End If

End Sub

Public Sub sendecom2(sollpos2, achse, speed)  'Unterprogramm Positioniervorgabe mit Geschwindigkeit
Dim Mldg, Stil, Titel, Antwort    'Parameter für Message Box
Dim sollpos2real

If flag_Winkel = True Then

                'Winkelumrechnung (0...-100) Grad in Bytes (0-255)
If sollpos2 < 0 Then    'Winkel negativ
   sollpos2real = (sollpos2 * (-1))
   sollpos2real = sollpos2real * 2.55
Else 'sollpos2 = 0 Then    'Winkel zero
   sollpos2real = 0
End If
   sollpos2real = CInt(sollpos2real)     'Nachkommastellen runden

   sollpos2 = sollpos2real
Else

End If

If (sollpos2 > 255) Or (sollpos2 < 0) Then
```

```
If flag_Winkel = True Then
   Mldg = "Bitte wählen ein andere Position für die Achse3  " & Chr$(10) & "    0..-100 Grad" 'Message Box >>> Warnung
Else
   Mldg = "Bitte wählen ein andere Position für die Achse3 " & Chr$(10) & "    0..255 "              'Message Box >>> Warnung
End If

Stil = vbOKOnly + vbCritical + vbDefaultButton1
    'Schaltflächen definieren.
Titel = "Positionierungsbereich der Achse3 überschritten"   ' Titel definieren.
Antwort = MsgBox(Mldg, Stil, Titel) ' Meldung anzeigen.
   txtsollpos2.Text = pos2


Else                'Übergabe der Werte nur im zulässigen Bereich

   Move_Rob                          ' Achsen in Bewegung

   MSComm1.Output = Chr$(&H61)                  'Kommando Motor - Regelung einschalten
   MSComm1.Output = Chr$(ETX)
   achse = 120 + achse

   MSComm1.Output = Chr$(achse)               '120 + Achsennummer
   MSComm1.Output = Chr$(sollpos2)
   MSComm1.Output = Chr$(speed)
   MSComm1.Output = Chr$(ETX)

End If

End Sub

Public Sub sendecom3(sollpos3, achse, speed)  'Unterprogramm Positioniervorgabe mit Geschwindigkeit
Dim Mldg, Stil, Titel, Antwort    'Parameter für Message Box
Dim sollpos3real

If flag_Winkel = True Then

                'Winkelumrechnung (100...0..-100) Grad in Bytes (0-255)
If sollpos3 > 0 Then        'Winkel positiv
   sollpos3real = 127 - (sollpos3 * 1.275)
ElseIf sollpos3 < 0 Then    'Winkel negativ
   sollpos3real = (sollpos3 * (-1))
   sollpos3real = 127 + (sollpos3real * 1.275)
ElseIf sollpos3 = 0 Then    'Winkel zero
   sollpos3real = 127
End If
   sollpos3real = CInt(sollpos3real)     'Nachkommastellen runden
   sollpos3 = sollpos3real

Else

End If
```

```
If (sollpos3 > 255) Or (sollpos3 < 0) Then

If flag_Winkel = True Then
   Mldg = "Bitte wählen ein andere Position für die Achse4  " & Chr$(10) & "    100...0..-100 Grad" 'Message Box >>> Warnung
Else
   Mldg = "Bitte wählen ein andere Position für die Achse4 " & Chr$(10) & "    0..255 "              'Message Box >>> Warnung
End If

Stil = vbOKOnly + vbCritical + vbDefaultButton1
    'Schaltflächen definieren.
Titel = "Positionierungsbereich der Achse4 überschritten"   ' Titel definieren.
Antwort = MsgBox(Mldg, Stil, Titel) ' Meldung anzeigen.

   txtsollpos3.Text = pos3

Else

   Move_Rob                              ' Achsen in Bewegung

   achse = 120 + achse
   MSComm1.Output = Chr$(achse)              '120 + Achsennummer
   MSComm1.Output = Chr$(sollpos3)
   MSComm1.Output = Chr$(speed)
   MSComm1.Output = Chr$(ETX)

End If

End Sub

Public Sub sendecom4(sollpos4, achse, speed)  'Unterprogramm Positioniervorgabe mit Geschwindigkeit
Dim Mldg, Stil, Titel, Antwort    'Parameter für Message Box
Dim sollpos4real

If flag_Winkel = True Then
                'Winkelumrechnung (100...0..-100) Grad in Bytes (0-255)
If sollpos4 > 0 Then       'Winkel positiv
   sollpos4real = 127 - (sollpos4 * 1.275)
ElseIf sollpos4 < 0 Then    'Winkel negativ
   sollpos4real = (sollpos4 * (-1))
   sollpos4real = 127 + (sollpos4real * 1.275)
ElseIf sollpos4 = 0 Then    'Winkel zero
   sollpos4real = 127
End If
   sollpos4real = CInt(sollpos4real)    'Nachkommastellen runden
    sollpos4 = sollpos4real

Else

End If

If (sollpos4 > 255) Or (sollpos4 < 0) Then
```

```
If flag_Winkel = True Then
   Mldg = "Bitte wählen ein andere Position für die Achse5  " & Chr$(10) & "    100...0..-100 Grad" 'Message Box >>> Warnung
Else
   Mldg = "Bitte wählen ein andere Position für die Achse5 " & Chr$(10) & "    0..255 "              'Message Box >>> Warnung
End If

Stil = vbOKOnly + vbCritical + vbDefaultButton1
     'Schaltflächen definieren.
Titel = "Positionierungsbereich der Achse5 überschritten"   ' Titel definieren.
Antwort = MsgBox(Mldg, Stil, Titel) ' Meldung anzeigen.
   txtsollpos4.Text = pos4

Else              'Übergabe der Werte nur im zulässigen Bereich

   Move_Rob                              ' Achsen in Bewegung

   achse = 120 + achse
   MSComm1.Output = Chr$(achse)              '120 + Achsennummer
   MSComm1.Output = Chr$(sollpos4)
   MSComm1.Output = Chr$(speed)
   MSComm1.Output = Chr$(ETX)

End If

End Sub

Public Sub sendecom5(sollpos5, achse, speed)  'Unterprogramm Positioniervorgabe mit Geschwindigkeit
Dim Mldg, Stil, Titel, Antwort    'Parameter für Message Box
Dim sollpos5real

If (sollpos5 > 255) Or (sollpos5 < 0) Then
      Mldg = "Bitte wählen ein andere Position für den Greifer (0 - 255)"              'Message Box >>> Warnung
Stil = vbOKOnly + vbCritical + vbDefaultButton1
     'Schaltflächen definieren.
Titel = "Positionierungsbereich der Greifer überschritten"   ' Titel definieren.
Antwort = MsgBox(Mldg, Stil, Titel) ' Meldung anzeigen.
   txtsollpos5.Text = pos5
Else

   Move_Rob                              ' Greifer in Bewegung

   achse = 120 + achse
   MSComm1.Output = Chr$(achse)              '120 + Achsennummer
   MSComm1.Output = Chr$(sollpos5)
   MSComm1.Output = Chr$(speed)
   MSComm1.Output = Chr$(ETX)

End If

End Sub
```

```
Public Sub Istwerte_vom_Rob(rückgabe)  'Rückgabe der Istwerte + Umrechnung in Winkel

    pos0 = rückgabe                    'Achse 1 Ist - Rückgabe
    pos0 = Mid(pos0, 2, 1)             'String Zeichen wählen -> ZeichenStart 2, Anzahl 1
    pos0 = Asc(pos0)                   'TestZahl = Asc("A")   ' Gibt 65 zurück.

If flag_Winkel = True Then     'Winkelumrechnung

' Bytes (0 -255) in (80...0..-80) Grad

If pos0 < 127 Then        'Winkel positiv
  pos0real = (127 - pos0) / 1.5875
ElseIf pos0 > 127 Then    'Winkel negativ
   pos0real = 80 - (pos0 / 1.5875)
ElseIf pos0 = 127 Then    'Winkel zero
   pos0 = 0
End If
   pos0real = CInt(pos0real)     'Nachkommastellen runden
   pos0 = pos0real

Else       'keine Winkelumrechnung

End If

If flag_SPS_Wahl = False Then           'keine DDE - Verbindung

Else
   On Error GoTo DDE_Fehler
   lblAchse1ist.LinkPoke       'DDE - Verbindung Wertrückgabe

End If

    pos1 = rückgabe                'Achse 2 Ist  - Rückgabe
    pos1 = Mid(pos1, 3, 1)
    pos1 = Asc(pos1)

If flag_Winkel = True Then     'Winkelumrechnung

' Bytes (0 -255) in (70...0..-30) Grad

If pos1 < 178 Then        'Winkel positiv
  pos1real = (178 - pos1) / 2.55
ElseIf pos1 > 178 Then    'Winkel negativ
  pos1real = (-1) * ((pos1 - 178) / 2.55)
ElseIf pos1 = 178 Then    'Winkel zero
   pos1 = 0
End If
   pos1 = CInt(pos1real)     'Nachkommastellen runden

Else       'keine Winkelumrechnung
End If
```

```
If flag_SPS_Wahl = False Then            'keine DDE - Verbindung

Else
   On Error GoTo DDE_Fehler
   lblAchse2ist.LinkPoke                 'DDE - Verbindung Wertrückgabe

End If

    pos2 = rückgabe                      'Achse 3 Ist - Rückgabe
    pos2 = Mid(pos2, 4, 1)
    pos2 = Asc(pos2)

If flag_Winkel = True Then     'Winkelumrechnung

' Bytes (0 -255) in (0..-100) Grad

If pos2 > 0 Then    'Winkel negativ
   pos2real = 0 - (pos2 / 2.55)
ElseIf pos2 = 0 Then    'Winkel zero
   pos2 = 0
End If
   pos2 = CInt(pos2real)     'Nachkommastellen runden

Else       'keine Winkelumrechnung
End If

If flag_SPS_Wahl = False Then            'keine DDE - Verbindung

Else
   On Error GoTo DDE_Fehler
   lblAchse3ist.LinkPoke                 'DDE - Verbindung Wertrückgabe

End If

    pos3 = rückgabe                      'Achse 4 Ist - Rückgabe
    pos3 = Mid(pos3, 5, 1)
    pos3 = Asc(pos3)

If flag_Winkel = True Then     'Winkelumrechnung

' Bytes (0 -255) in (100...0..-100) Grad

If pos3 < 127 Then        'Winkel positiv
  pos3real = (127 - pos3) / 1.275
ElseIf pos3 > 127 Then    'Winkel negativ
   pos3real = 100 - (pos3 / 1.275)
ElseIf pos3 = 127 Then    'Winkel zero
   pos3 = 0
End If
   pos3 = CInt(pos3real)     'Nachkommastellen runden

Else       'keine Winkelumrechnung
```

```
End If

If flag_SPS_Wahl = False Then            'keine DDE - Verbindung

Else
   On Error GoTo DDE_Fehler
   lblAchse4ist.LinkPoke               'DDE - Verbindung Wertrückgabe

End If


    pos4 = rückgabe                     'Achse 5 Ist - Rückgabe
    pos4 = Mid(pos4, 6, 1)
    pos4 = Asc(pos4)

If flag_Winkel = True Then     'Winkelumrechnung

' Bytes (0 -255) in (100...0..-100) Grad

If pos4 < 127 Then        'Winkel positiv
  pos4real = (127 - pos4) / 1.275
ElseIf pos4 > 127 Then    'Winkel negativ
   pos4real = 100 - (pos4 / 1.275)
ElseIf pos4 = 127 Then    'Winkel zero
   pos4 = 0
End If
   pos4 = CInt(pos4real)     'Nachkommastellen runden

Else       'keine Winkelumrechnung
End If

If flag_SPS_Wahl = False Then            'keine DDE - Verbindung

Else
   On Error GoTo DDE_Fehler
   lblAchse5ist.LinkPoke               'DDE - Verbindung Wertrückgabe

End If

    pos5 = rückgabe                     'Greifer  - Rückgabe
    pos5 = Mid(pos5, 7, 1)
    pos5 = Asc(pos5)

Exit Sub
DDE_Fehler: DDE_Verbindung                'Verbindung aufbauen

End Sub

Public Sub Digout_vom_Rob(digout)          'Rückgabe der I/O Box

   digout = Mid(digout, 2, 1)           'String Zeichen wählen -> ZeichenStart 2, Anzahl 1
   digout = Asc(digout)                'Zeichen in ASCII dezimal z. B. TestZahl = Asc("A") Gibt 65 zurück
```

```
If (digout And &H1) Then     'switch1       ' Achtung Bit - Vergleich, nicht Zahlen
   lblswitch1.BackColor = &HFF&        'Farbe ändern >> rot

If flag_SPS_Wahl = False Then             'keine DDE - Verbindung
   'keine Rückgabe an die SPS
Else
    lblswitch1.Caption = 1
    On Error GoTo DDE_Fehler
    lblswitch1.LinkPoke             'DDE - Verbindung Wertrückgabe

End If

Else
      lblswitch1.BackColor = &HC0C0C0     'Farbe ändern >> grau

If flag_SPS_Wahl = False Then             'keine DDE - Verbindung
   'keine Rückgabe an die SPS
Else
     lblswitch1.Caption = 0

     On Error GoTo DDE_Fehler
     lblswitch1.LinkPoke             'DDE - Verbindung Wertrückgabe

End If

End If

If (digout And &H2) Then     'switch2

       lblswitch2.BackColor = &HFF&        'Farbe ändern >> rot
      'keine Rückgabe an die SPS

If flag_SPS_Wahl = False Then             'keine DDE - Verbindung

Else
    lblswitch2.Caption = 1
    On Error GoTo DDE_Fehler
    lblswitch2.LinkPoke             'DDE - Verbindung Wertrückgabe

End If

Else
      lblswitch2.BackColor = &HC0C0C0     'Farbe ändern >> grau
     'keine Rückgabe an die SPS

If flag_SPS_Wahl = False Then             'keine DDE - Verbindung

Else
     lblswitch2.Caption = 0

    On Error GoTo DDE_Fehler
```

```
    lblswitch2.LinkPoke             'DDE - Verbindung Wertrückgabe

End If

End If
If (digout And &H4) Then     'switch3

  lblswitch3.BackColor = &HFF&        'Farbe ändern >> rot
  'keine Rückgabe an die SPS

If flag_SPS_Wahl = False Then             'keine DDE - Verbindung

Else
   lblswitch3.Caption = 1
   On Error GoTo DDE_Fehler
   lblswitch3.LinkPoke             'DDE - Verbindung Wertrückgabe

End If

Else
      lblswitch3.BackColor = &HC0C0C0     'Farbe ändern >> grau

  'keine Rückgabe an die SPS
If flag_SPS_Wahl = False Then             'keine DDE - Verbindung

Else
    lblswitch3.Caption = 0

      On Error GoTo DDE_Fehler
      lblswitch3.LinkPoke             'DDE - Verbindung Wertrückgabe

End If

End If
If (digout And &H8) Then     'switch4
        lblswitch4.BackColor = &HFF&        'Farbe ändern >> rot

  'keine Rückgabe an die SPS
If flag_SPS_Wahl = False Then             'keine DDE - Verbindung

Else
   lblswitch4.Caption = 1
 On Error GoTo DDE_Fehler
   lblswitch4.LinkPoke             'DDE - Verbindung Wertrückgabe

End If

Else
            lblswitch4.BackColor = &HC0C0C0     'Farbe ändern >> grau

  'keine Rückgabe an die SPS
If flag_SPS_Wahl = False Then             'keine DDE - Verbindung
```

```
Else
    lblswitch4.Caption = 0

 On Error GoTo DDE_Fehler
      lblswitch4.LinkPoke            'DDE - Verbindung Wertrückgabe

End If

End If
If (digout And &H10) Then    'switch5
  lblswitch5.BackColor = &HFF&       'Farbe ändern >> rot

  'keine Rückgabe an die SPS
If flag_SPS_Wahl = False Then            'keine DDE - Verbindung

Else
   lblswitch5.Caption = 1
 On Error GoTo DDE_Fehler
   lblswitch5.LinkPoke            'DDE - Verbindung Wertrückgabe

End If

Else
           lblswitch5.BackColor = &HC0C0C0    'Farbe ändern >> grau

  'keine Rückgabe an die SPS
If flag_SPS_Wahl = False Then            'keine DDE - Verbindung

Else
    lblswitch5.Caption = 0

 On Error GoTo DDE_Fehler
      lblswitch5.LinkPoke            'DDE - Verbindung Wertrückgabe

End If

End If
If (digout And &H20) Then    'switch6
  lblswitch6.BackColor = &HFF&       'Farbe ändern >> rot

  'keine Rückgabe an die SPS
If flag_SPS_Wahl = False Then            'keine DDE - Verbindung

Else
   lblswitch6.Caption = 1
 On Error GoTo DDE_Fehler
   lblswitch6.LinkPoke            'DDE - Verbindung Wertrückgabe

End If

Else
```

```
      lblswitch6.BackColor = &HC0C0C0     'Farbe ändern >> grau

  'keine Rückgabe an die SPS
If flag_SPS_Wahl = False Then             'keine DDE - Verbindung

Else
    lblswitch6.Caption = 0
    On Error GoTo DDE_Fehler
    lblswitch6.LinkPoke             'DDE - Verbindung Wertrückgabe

End If

End If

If (digout And &H40) Then     'switch7
   lblswitch7.BackColor = &HFF&        'Farbe ändern >> rot

  'keine Rückgabe an die SPS
If flag_SPS_Wahl = False Then             'keine DDE - Verbindung

Else
   lblswitch7.Caption = 1
   On Error GoTo DDE_Fehler
   lblswitch7.LinkPoke             'DDE - Verbindung Wertrückgabe

End If

Else
          lblswitch7.BackColor = &HC0C0C0     'Farbe ändern >> grau

  'keine Rückgabe an die SPS
If flag_SPS_Wahl = False Then             'keine DDE - Verbindung

Else
    lblswitch7.Caption = 0
    On Error GoTo DDE_Fehler
    lblswitch7.LinkPoke             'DDE - Verbindung Wertrückgabe

End If

End If

If (digout And &H80) Then       'switch8
  lblswitch8.BackColor = &HFF&        'Farbe ändern >> rot

  'keine Rückgabe an die SPS
If flag_SPS_Wahl = False Then             'keine DDE - Verbindung

Else
   lblswitch8.Caption = 1
   On Error GoTo DDE_Fehler
   lblswitch8.LinkPoke             'DDE - Verbindung Wertrückgabe
```

```
End If

Else
        lblswitch8.BackColor = &HC0C0C0     'Farbe ändern >> grau

  'keine Rückgabe an die SPS
If flag_SPS_Wahl = False Then          'keine DDE - Verbindung

Else
    lblswitch8.Caption = 0
    On Error GoTo DDE_Fehler
    lblswitch8.LinkPoke          'DDE - Verbindung Wertrückgabe
End If

End If
   Exit Sub

DDE_Fehler: DDE_Verbindung              'Verbindung aufbauen

End Sub

Public Sub Soll_Pos_erreicht()       'Soll Position erreicht>> weiss

flag_Position_erreicht = True
flag_move = False

   lblAchse1ist.BackColor = &HFFFFFF
   lblAchse2ist.BackColor = &HFFFFFF
   lblAchse3ist.BackColor = &HFFFFFF
   lblAchse4ist.BackColor = &HFFFFFF
   lblAchse5ist.BackColor = &HFFFFFF
   lblGreiferist.BackColor = &HFFFFFF

End Sub

Public Sub Move_Rob()                'Achsen in Bewegung >> gelb

flag_move = True
flag_Position_erreicht = False

   lblAchse1ist.BackColor = &H80FFFF
   lblAchse2ist.BackColor = &H80FFFF
   lblAchse3ist.BackColor = &H80FFFF
   lblAchse4ist.BackColor = &H80FFFF
   lblAchse5ist.BackColor = &H80FFFF
   lblGreiferist.BackColor = &H80FFFF

End Sub

Private Sub DDE_Verbindung()    'DDE Verbindung herstellen + Fehler Behandlung
```

```
Dim Mldg As String  'Parameter für Message Box
Dim Stil As String
Dim Titel As String
Dim Antwort As String

  ProjektName = "PROSYS|" & NameSPS        'D:\DDE.pro"
  'ProjektName = "PROSYS|D:\Rob5.pro"      'Übergabe alt

On Error GoTo DDE_Fehler          ' DDE  - Kommunikationsfehler >> Sprung zur Marke DDE_Fehler

                                  'Achsensoll Vorgabe ProSys

  lblsollpos0_DDE.LinkTopic = ProjektName                          '
  lblsollpos0_DDE.LinkItem = ".achse1soll"                         '
  lblsollpos0_DDE.LinkMode = vbLinkAutomatic                   'akt iviert Verbindung automatisch

  lblsollpos1_DDE.LinkTopic = ProjektName
  lblsollpos1_DDE.LinkItem = ".achse2soll"                         '
  lblsollpos1_DDE.LinkMode = vbLinkAutomatic

  lblsollpos2_DDE.LinkTopic = ProjektName
  lblsoll pos2_DDE.LinkItem = ".achse3soll"                        '
  lblsollpos2_DDE.LinkMode = vbLinkAutomatic

  lblsollpos3_DDE.LinkTopic = ProjektName
  lblsollpos3_DDE.LinkItem = ".achse4soll"                         '
  lblsollpos3_DDE.LinkMode = vbLinkAutomatic

  lblsollpos4_DDE.LinkTopic = ProjektName
  lblsollpos4_DDE.LinkItem = ".achse5soll"                         '
  lblsollpos4_DDE.LinkMode = vbLinkAutomatic

  lblsollpos5_DDE.LinkTopic = ProjektName
  lbl sollpos5_DDE.LinkItem = ".Greifersoll"                       '
  lblsollpos5_DDE.LinkMode = vbLinkAutomatic

                                  ' Geschwindigkeit Vorgabe ProSys
  lblspeed_DDE.LinkTopic = ProjektName
  lblspeed_DDE.LinkItem = ".speed"                                 '
  lblspeed_DDE.LinkMode = vbLinkAutomatic


                                  ' Achsen Ist_Rückgabe vom ROB3
  lblAchse1ist.LinkTopic = Projekt Name                            '
  lblAchse1ist.LinkItem = ".achse1ist"                             '
  lblAchse1ist.LinkMode = vbLinkAutomatic                      'aktiviert Verbindung automatisch

  lblAchse2ist.LinkTopic = ProjektName                             '
  lblAchse2ist.LinkItem = ".achse2ist"                             '
  lblAchse2ist.LinkMode = vbLinkAutomatic

  lblAchse3ist.LinkTopic = ProjektName                             '
```

```
lblAchse3ist.LinkItem = ".achse3ist"                    '
lblAchse3ist.LinkMode = vbLinkAutomatic

lblAchse4ist.LinkTopic = ProjektName                    '
lblAchse4ist.LinkItem = ".achse4ist"                    '
lblAchse4ist.LinkMode = vbLinkAutomatic

lblAchse5ist.LinkTopic = ProjektName                    '
lblAchse5ist.LinkItem = ".achse5ist"                    '
lblAchse5ist.LinkMode = vbLinkAutomatic

lblGreiferist.LinkTopic = ProjektName                   '
lblGreiferist.LinkItem = ".Greiferist"                  '
lblGreiferist.LinkMode = vbLinkAutomatic

                        'Schalter I/O Box

lblswitch1.LinkTopic = ProjektName
lblswitch1.LinkItem = ".Schalter1"                      '
lblswitch1.LinkMode = vbLinkAutomatic

lblswitch2.LinkTopic = ProjektNam e
lblswitch2.LinkItem = ".Schalter2"                      '
lblswitch2.LinkMode = vbLinkAutomatic

lblswitch3.LinkTopic = ProjektName
lblswitch3.LinkItem = ".Schalter3"                      '
lblswitch3.LinkMode = vbLinkAutomatic

lblswitch4.LinkTopic = ProjektName
lblswitch4.LinkItem = ".Schalter4"                      '
lblswitch4.LinkMode = vbLinkAutomatic

lblswitch5.LinkTopic = ProjektName
lblswitch5.LinkItem = ".Schalter5"                      '
lblswitch5.LinkMode = vbLinkAutomatic

lblswitch6.LinkTopic = ProjektName
lblswitch6.LinkItem = ".Schalter6"                      '
lblswitch6.LinkMode = vbLinkAutomatic

lblswitch7.LinkTopic = ProjektName
lblswitch7.LinkItem = ".Schalter7"                      '
lblswitch7.LinkMode = vbLinkAutomatic

lblswitch8.LinkTopic = ProjektName
lblswitch8.LinkItem = ".Schalter8"                      '
lblswitch8.LinkM ode = vbLinkAutomatic

                        'LED 's I/O Box

lblLED1.LinkTopic = ProjektName
```

```
  lblLED1.LinkItem = ".OUTLED1"                    '
  lblLED1.LinkMode = vbLinkAutomatic

  lblLED2.LinkTopic = ProjektName
  lblLED2.LinkItem = ".OUTLED2"                    '
  lblLED2.LinkMode = vbLinkAutomatic

  lblLED3.LinkTopic = ProjektName
  lblLED3.LinkItem = ".OUTLED3"                    '
  lblLED3.LinkMode = vbLinkAutomati c

  lblLED4.LinkTopic = ProjektName
  lblLED4.LinkItem = ".OUTLED4"                    '
  lblLED4.LinkMode = vbLinkAutomatic

  lblLED5.LinkTopic = ProjektName
  lblLED5.LinkItem = ".OUTLED5"                    '
  IblLED5.LinkMode = vbLinkAutomatic

  lblLED6.LinkTopic = ProjektName
  lblLED6.LinkItem = ".OUTLED6"                    '
  lblLED6.LinkMode = vbLinkAutomatic

  lblLED7.LinkTopic = ProjektName
  lblLED7.LinkItem = ".OUTLED7"                    '
  lblLED7.LinkMode = vbLinkAutomatic

  lblLED8.LinkTopic = ProjektName
  lblLED8.LinkItem = ".OUTLED8"                    '
  lblLED8.LinkMode = vbLinkAutomatic

  lblPos_erreicht.LinkTopic = ProjektName
  lblPos_erreicht.LinkItem = ".flag_Pos_erreicht"                    '
  lblPos_erreicht.LinkMode = vbLinkAutomatic


  DDE_Oberfläche              'DDE - Oberfläche aufrufen

 'Abfrage nach Art der Sollwerte >> Grad oder Byte 0..255

Mldg = "Wollen die Sollwerte der Achsen als Grad angeben dann bestätigen Sie mit >>OK " & Chr$(10) & "Wollen Sie die Werte
Byte(0..255) angeben, dann wählen Sie >> Nein "
Stil = vbYesNo + vbQuestion + vbDefaultButton1
    'Schaltflächen definieren.
Titel = "Wie wollen Sie den Roboter ansteuern ?"   ' Titel definieren.
Antwort = MsgBox(Mldg, Stil, Titel) ' Hilfe, Ktxt)   ' Meldung anzeigen.

If Antwort = vbYes Then   'Benutzer hat "Ja" gewählt >> Sollwerte als Grad

   flag_Winkel = True      'Übergabe der Sollwerte als Grad

ElseIf Antwort = vbNo Then 'Benutzer hat "Ja" gewählt >> Sollwerte als Grad
```

```
    flag_Winkel = False     'Übergabe der Sollwerte als Byte 0...255

    cmdDDE_beenden.BackColor = &HFF&            'Button aktiv >> rot
    cmdDDE_Verbindung.BackColor = &HE0E0E0      'Button inaktiv >> grau

End If

   cmdDDE_Verbindung.BackColor = &HFF&     'Button aktiv >> rot
   cmdDDE_beenden.BackColor = &HE0E0E0   'Button inaktiv >> grau

                          'Hilfsflag Sollposition erreicht

Exit Sub

DDE_Fehler:                       ' DDE  - Fehler

'If flag_SPS_Wahl = False Then           'keine DDE - Verbindung

'Else                  'Message Box parametrieren

Mldg = "Starten Sie bitte " & NameSPS & "  und bestätigen Sie mit OK " & Chr$(10) & "      oder geben Sie den neuen Namen
des SPS  ->> Nein " & Chr$(10) & " für die Ansteuerung des Roboters ohne SPS wählen Sie >>Abbrechen" ' Meldung
definieren.
Stil = vbYesNoCancel + vbQuestion + vbDefaultButton1
     'Schaltflächen definieren.
Titel = "DDE - Verbindung Fehler !!!"   ' Titel definieren.
Antwort = MsgBox(Mldg, Stil, Titel) ' Hilfe, Ktxt)    ' Meldung anzeigen.

If Antwort = vbYes Then   'Benutzer hat "Ja" gewählt >> Ansteuerung über SPS .

    DDE_Verbindung ' SPS noch mal starten
    'DialogSPS.Show
    'frmhaupt.Visible = False

ElseIf Antwort = vbNo Then 'Benutzer hat "Nein" gewählt >> Ansteuerung über SPS >> neuer Name

    flag_SPS_Wahl = True
    frmhaupt.Visible = False
    DialogProSys.Show      ' Gebe den neuen Namen ein

ElseIf Antwort = vbCancel Then 'Benutzer hat "Abbrechen" gewählt >> Ansteuerung ohne SPS

    flag_SPS_Wahl = False
    DDE_Schließen
    No_DDE_Oberfläche

Exit Sub                  'verlasse  sub
'End If

End If
```

```
Exit Sub

End Sub
Public Sub DDE_Schließen()        'DDE  - Kanäle schließen

lblspeed_DDE.LinkMode = None
lblsollpos0_DDE.LinkMode = None                  'beendet DDE - Verbindung
lblsollpos1_DDE.LinkMode = None                  'beendet DDE - Verbindung
lblsollpos2_DDE.LinkMode = None                  'beendet DDE - Verbindung
lblsollpos3_DDE.LinkMode = None                  'beendet DDE - Verbindung
lblsollpos4_DDE.LinkMode = None                  'beendet DDE - Verbindung
lblsollpos5_DDE.LinkMode = None                  'beendet DDE - Verbindung

End Sub

Public Sub DDE_Oberfläche()     'Bedienoberfläche bei DDE -Verbindung

If flag_Winkel = True Then         ' Sollvorgabe als Winkel oder Byte

   lblBereich1.Caption = "80     0     -80"
   lblBereich2.Caption = "70     0       "
   lblBereich3.Caption = "       0   -100"
   lblBereich4.Caption = "100    0   -100"
   lblBereich5.Caption = "100    0   -100"
   lblBereich6.Caption = " 0.........255"
   cmdWinkel.Caption = "Byte"

Else
   lblBereich1.Caption = " 0.........255"
   lblBereich2.Caption = " 0.........255"
   lblBereich3.Caption = " 0.........255"
   lblBereich4.Caption = " 0.........255"
   lblBereich5.Caption = " 0.........255"
   lblBereich6.Caption = " 0.........255"
   cmdWinkel.Caption = "Winkel"

End If

lbl_1.Left = 480
lbl_1.Top = 2520

lbl_2(0).Left = 480
lbl_2(0).Top = 3360

lbl_3(1).Left = 480
lbl_3(1).Top = 4200

lbl_4(2).Left = 360
lbl_4(2).Top = 5040

lbl_5(3).Left = 480
lbl_5(3).Top = 5880
```

```
lbl_6(4).Left = 600
lbl_6(4).Top = 6720

lblsollpos0_DDE.Visible = True      'Steuerelemente anzeigen
lblsollpos1_DDE.Visible = True
lblsollpos2_DDE.Visible = True
lblsollpos3_DDE.Visible = True
lblsollpos4_DDE.Visible = True
lblsollpos5_DDE.Visible = True
lblspeed_DDE.Visible = True

cmdübergabe_ist_txt.Visible = False     'Steuerelemente verstecken
cmdWeitergabePos1.Visible = False
cmdspeed1.Visible = False
cmdspeed2.Visible = False
cmdspeed3.Visible = False
cmdspeed4.Visible = False
cmdspeed5.Visible = False
cmdspeed6.Visible = False
cmdspeed7.Visible = False

cmdAchse1minus.Visible = False
cmdAchse1plus(0).Visible = False
cmdAchse2minus(2).Visible = False
cmdAchse2minus(2).Visible = False
cmdAchse2plus(1).Visible = False
cmdAchse3minus(4).Visible = False
cmdAchse3plus(3).Visible = False
cmdAchse4minus(6).Visible = False
cmdAchse4plus(5).Visible = False
cmdAchse5minus(8).Visible = False
cmdAchse5plus(7).Visible = False
cmdGreiferzu.Visible = False
cmdGreiferauf.Visible = False
Schritt10.Visible = False
Schritt1.Visible = False
cmdposition.Visible = False

txtsollpos0.Visible = False
txtsollpos1.Visible = False
txtsollpos2.Visible = False
txtsollpos3.Visible = False
txtsollpos4.Visible = False
txtsollpos5.Visible = False

End Sub

Public Sub No_DDE_Oberfläche()     'Bedienoberfläche bei DDE -Verbindung

If flag_Winkel = True Then          ' Sollvorgabe als Winkel oder Byte
```

```
    lblBereich1.Caption = "80     0    -80"
    lblBereich2.Caption = " 70     0       "
    lblBereich3.Caption = "       0   -100"
    lblBereich4.Caption = "100    0   -100"
    lblBereich5.Caption = "100    0   -100"
    lblBereich6.Caption = "0.....255"
    cmdWinkel.Caption = "Byte"

Else
    lblBereich1.Caption = " 0.....255"
    lblBereich2.Caption = " 0.....255"
    lblBereich3.Caption = " 0.....255"
    lblBereich4.Caption = " 0.....255"
    lblBereich5.Caption = " 0.....255"
    lblBereich6.Caption = " 0.....255"
    cmdWinkel.Caption = "Winkel"
End If

lbl_1.Left = 8925
lbl_1.Top = 2640

lbl_2(0).Left = 8925
lbl_2(0).Top = 3480

lbl_3(1).Left = 8925
lbl_3(1).Top = 4320

lbl_4(2).Left = 8820
lbl_4(2).Top = 5160

lbl_5(3).Left = 8925
lbl_5(3).Top = 6000

lbl_6(4).Left = 9030
lbl_6(4).Top = 6840

cmdübergabe_ist_txt.Visible = True  'Steuerelemente anzeigen
cmdWeitergabePos1.Visible = True

cmdspeed1.Visible = True
cmdspeed2.Visible = True
cmdspeed3.Visible = True
cmdspeed4.Visible = True
cmdspeed5.Visible = True
cmdspeed6.Visible = True
cmdspeed7.Visible = True

cmdAchse1minus.Visible = True
cmdAchse1plus(0).Visible = True
cmdAchse2minus(2).Visible = True
cmdAchse2minus(2).Visible = True
cmdAchse2plus(1).Visible = True
```

```
cmdAchse3minus(4).Visible = True
cmdAchse3plus(3).Visible = True
cmdAchse4minus(6).Visible = True
cmdAchse4plus(5).Visible = True
cmdAchse5minus(8).Visible = True
cmdAchse5plus(7).Visible = True
cmdGreiferzu.Visible = True
cmdGreiferauf.Visible = True
Schritt10.Visible = True
Schritt1.Visible = True
cmdposition.Visible = True

txtsollpos0.Visible = True
txtsollpos1.Visible = True
txtsollpos2.Visible = True
txtsollpos3.Visible = True
txtsollpos4.Visible = True
txtsollpos5.Visible = True

lblsollpos0_DDE.Visible = False     'Steuerelemente verstecken
lblsollpos1_DDE.Visible = False
lblsollpos2_DDE.Visible = False
lblsollpos3_DDE.Visible = False
lblsollpos4_DDE.Visible = False
lblsollpos5_DDE.Visible = False
lblspeed_DDE.Visible = False

End Sub
```