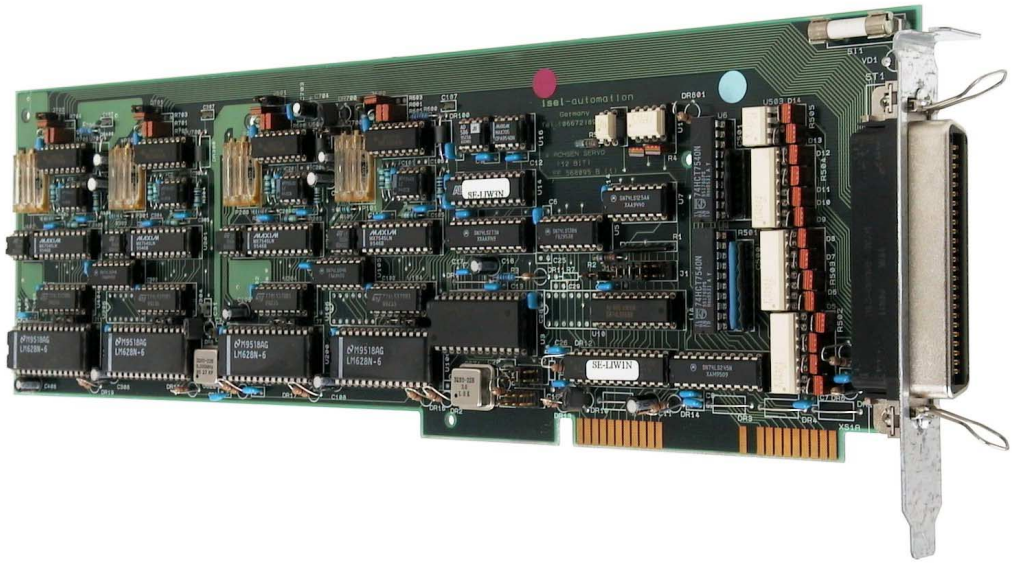
isel-4-Achs-Servomotor-Steuerkarte UPMV 4/12

Für Version 3.00 - 3.10 - 3.20



Bedienungsanleitung

B.326 003.03/2000.24

Zu dieser Anleitung

In dieser Anleitung finden Sie verschiedene Symbole, die Ihnen schnell wichtige Informationen anzeigen.

Gefahr	Achtung	Hinweis	Beispiel	Zusatz-Infos
				

Trotz aller Sorgfalt können Druckfehler und Irrtümer nicht ausgeschlossen werden.
Für Verbesserungsvorschläge und Hinweise auf Fehler sind wir dankbar.

Hersteller: Fa. ***isel****automation* GmbH & Co.KG
Bürgermeister-Ebert-Str. 40
D-36124 Eichenzell

Fax: (06659) 981-776
e-mail: automation@isel.com
http://www.isel.com

Wichtige Hinweise für die Benutzung der *isel*-Servomotor-Steuerkarte UPMV 4/12

Dieses Handbuch ist für die PC-Einsteckkarte UPMV 4/12 bestimmt. In diesem Handbuch sind enthalten:

- die Beschreibung der PC-Einsteckkarte, sowohl für die Hardware als auch für die mitgelieferte Treibersoftware
- die Beschreibung für das Konfigurationsprogramm PARKON.EXE
- die Beschreibung für das Einstellprogramm PAREIN.EXE

Die aufeinander folgenden Schritte bei der Installation der Hardware und Software sind hier beschrieben.

1. Installation der Hardware

- Einstellen der Basisadresse für die Karte
  (siehe Kapitel 2.2 in der Hardware-Beschreibung)
- Einstellen der Hardware-Interruptnummer
  (siehe Kapitel 2.3 und 2.4 in der Hardware-Beschreibung)
- Anpassen der Encoder-Signale und der Encoder-Spannung
  (siehe Kapitel 2.7 in der Hardware-Beschreibung)
- Einstecken der Karte in einen Rechner mit dem DOS/WINDOWS-Betriebssystem von *Microsoft*.
  Es wird ein verlängerter ISA-Bus-Steckplatz benötigt. Falls Sie einen Pentium-Rechner haben, müssen Sie wahrscheinlich Änderungen im BIOS des Rechners vornehmen (siehe Kapitel 2.4 in der Hardware-Beschreibung und 3.2.12.2 in der Software-Beschreibung).
- Verbinden der Karte mit den Leistungseinheiten und der Mechanik.
  Der 50-polige Steckverbinder ist das Hardwareinterface der Karte zur Außenwelt (siehe Kapitel 2.9 in der Hardware-Beschreibung).
  Falls Sie einen *isel*-Servocontroller benutzen, erfolgt die Verbindung über das mitgelieferte Kabel.
- Wenn Sie von der Treiberversion 3.00 oder 3.00A auf die Version 3.10 umsteigen, sollten Sie unbedingt den Punkt 6 lesen.
- Weil wir standardmäßig nur noch die Treiberversion 3.10 ausliefern, sollten Sie unbedingt den Punkt 7 lesen, wenn Sie weiterhin die Treiberversion 3.00 oder 3.00A benutzen wollen.

2. Übertragen der Software auf den Rechner

- Verwenden Sie das Programm INSTALL.BAT, um die mitgelieferte Software zu kopieren. Das Verzeichnis *\SERVO* auf Ihrem Rechner enthält alle Programme, die Sie für die Benutzung der Karte benötigen.

3. Inbetriebnahme der Karte

- Erstellen einer Initialisierungsdatei mit dem Programm PARKON.EXE (siehe dazugehörige Beschreibung).
  Die mitgelieferte Initialisierungsdatei EXAMPLE.INI soll als Beispiel für Sie dienen. Falls Sie einen *isel*-Servocontroller benutzen, sollten Sie unbedingt das entsprechende Handbuch dafür lesen.
- Inbetriebnahme der Karte und Ihrer Anlage mit dem Programm PAREIN.EXE (siehe dazugehörige Beschreibung).
  Es ist unbedingt notwendig, alle Untermenüs in diesem Programm für jede Achse durchzuarbeiten.

4. Installation der Treibersoftware

- Installieren der Treibersoftware ISELDRV.EXE für die Karte (siehe Kapitel 3.1 in der Software-Beschreibung).
  Der Installationsprozess lässt sich automatisieren, wenn Sie die entsprechende Kommandozeile in die Datei AUTOEXEC.BAT Ihres Rechners eintragen.

5. Starten des Anwenderprogramms

- Starten Sie Ihr Anwenderprogramm.
  Ohne ein entsprechendes Anwenderprogramm können Sie die Dienste des Treibers und der Karte nicht benutzen. Als Anwenderprogramm können Sie unsere Software wie z. B. die Programme HPREMOTE, REMOTE, PRO-DIN, PRO-PAL, etc. einsetzen.
  Wenn Sie Ihr Anwenderprogramm selbst entwickeln wollen, sollten Sie die Hardware- und Software-Beschreibung ausführlich lesen. Dort finden Sie entsprechende Hinweise und die Programmierschnittstellen.
  Falls Sie von der Treiberversion 3.00 auf die Version 3.10 umsteigen, sollten Sie unbedingt den Punkt 6 lesen.

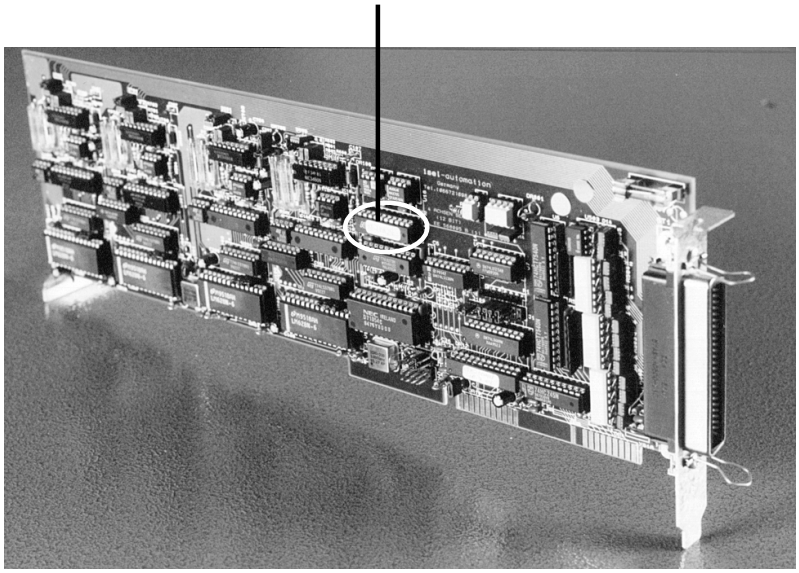
6. Umsteigen von der Treiberversion 3.00 oder 3.00A auf die Version 3.10

Um von der Treiberversion 3.00 oder 3.00A auf die Version 3.10 umsteigen zu können, sind folgende Schritte notwendig:

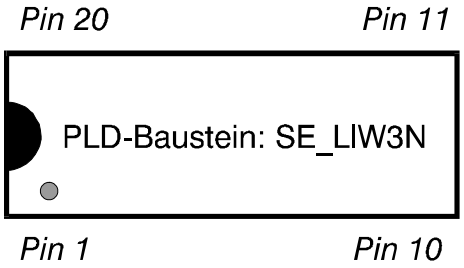
6.1 Hardware-Änderung auf der PC-Einsteckkarte

- Fordern Sie von uns einen neuen PLD-Baustein mit der Kennzeichnung SE_LIW3N an.
- Bauen Sie die PC-Einsteckkarte aus Ihrem PC aus.
- Tauschen Sie den PLD-Baustein auf der PC-Einsteckkarte aus.

Umzutauschender PLD-Baustein



Achten Sie beim Umtausch unbedingt auf die richtige Pin-Position des Bausteins.



Die Pin-Nummer des PLD-Bausteins können Sie anhand der Halbkreis-Markierung am Rand erkennen. Auf der Platine der PC-Einsteckkarte (s. Zeichnung) muss der Pin 1 des neuen PLD-Bausteins in der Ecke unten links sein (weitere Informationen in der Hardware-Beschreibung, Kapitel 2.8).

- Bauen Sie die PC-Einsteckkarte wieder ein.

6.2 Änderung der Treibersoftware

- Installieren Sie die neue Treibersoftware.

  Sie sollten unbedingt eine Sicherheitskopie von der alten Treibersoftware einschließlich der Initialisierungsdatei machen. Danach installieren Sie die neue Treiberversion. Starten Sie dazu das Programms INSTALL.BAT auf der mitgelieferten Diskette. Alle notwendigen Programme werden in das Verzeichnis /*SERVO* kopiert.
  Falls Sie bis z. Z. ein anderes Verzeichnis für den Servo-Treiber benutzen, können Sie die neuen Programme von dem Verzeichnis /*SERVO* in das alte Verzeichnis kopieren.

- Umwandeln der Initialisierungsdatei der alten Treiberversion in eine Treiberversion 3.10.

  Starten Sie das neue Konfigurationsprogramm PARKON.EXE, um die alte Initialisierungsdatei *.INI Ihrer Anlage in eine Initialisierungsdatei der Treiberversion 3.10 zu konvertieren.
  Das Konvertieren findet im Untermenü *Konvertieren* des Hauptmenüs *Datei* statt. Beim Konvertieren werden alle Anlagenparameter in die neue Initialisierungsdatei übernommen. Hier gibt es aber einige Erweiterungen.

  a) Im Untermenü *Ref_Schalter* des Hauptmenüs *Software* können Sie den Scan-Code der von Ihnen gewünschten Taste für die Unterbrechung der Referenzfahrt definieren. Standardmäßig wird dafür die ESC-Taste genutzt. Mit dem Einstellprogramm PAREIN.EXE können Sie jederzeit den Scan-Code jeder anderen Taste ermitteln.

  b) Im Untermenü *Überwachungsport* des Hauptmenüs *Software* können Sie die Ein- und Ausgabeports für die Überwachung der Anlage konfigurieren. Im Fall, dass Ihre Anlage keine *isel*-Anlage ist, sollten Sie die Beschreibung für das Konfigurationsprogramm PARKON.EXE lesen. Ansonsten sollten Sie uns kontaktieren. Wir werden Ihnen weiterhelfen.

  c) Im Untermenü *V_Achsfaktor* des Hauptmenüs *Hardware* werden die Geschwindigkeitsverstärkungsfaktoren der Achsen definiert. Diese Faktoren können Sie mit Hilfe des Einstellprogramms PAREIN.EXE ermitteln.

  d) Beim Konvertieren werden der Kreisreduktionsfaktor und der Bahnreduktionsfaktor im Untermenü *V_Reduktion* des Hauptmenüs *Hardware* gleich 1 gesetzt. In der neuen Treiberversion sollten Sie diese Werte nicht mehr ändern.

e) Im Untermenü *Regler* des Hauptmenüs *Hardware* besteht die Möglichkeit zu definieren, ob die Achsregler im Fall eines Nachlauffehlers ausgeschaltet werden sollen oder nicht. Standardmäßig ist immer „Aus“ eingestellt, d.h. die Achsregler werden ausgeschaltet.

Das Programm wird erst nach dem Abspeichern der Initialisierungsdatei beendet.

Lesen Sie die Beschreibung des Konfigurationsprogramms PARKON.EXE, um mehr Informationen zu bekommen.

- Vervollständigen der neuen Initialisierungsdatei mit dem neuen Einstellprogramm PAREIN.EXE.

a) Im Menü *Einsteckkarte* des Hauptmenüs *Anlage* wird die Hardware der Einsteckkarte geprüft. Beachten Sie, dass die Software nicht kontrollieren kann, ob der PLD-Baustein schon umgetauscht ist oder nicht.

b) Im Untermenü *Überwachung* des Hauptmenü *Anlage* können Sie die Konfiguration der Überwachungsports kontrollieren. Im Fall, dass Ihre Anlage eine *isel*-Anlage ist, können Sie die Standardeinstellung nehmen, die Sie mit dem Konfigurationsprogramm PARKON.EXE eingegeben haben.

c) Im Untermenü *Ref._Taste* des Hauptmenüs *Anlage* ermitteln Sie den Scan-Code der gewünschten Taste, mit der Sie die Referenzfahrt jederzeit unterbrechen können. Sie sollten die Standardtaste (ESCAPE) benutzen.

d) Im Untermenü *Posi._Regler* des Hauptmenüs *Achse* können Sie die Geschwindigkeitsverstärkungsfaktoren der Achsen ermitteln. Die Reglerparameter brauchen Sie nicht neu zu bestimmen. Lesen Sie unbedingt das entsprechende Kapitel in der Beschreibung des Einstellprogramms PAREIN.EXE.

6.3 Änderung der Anwendersoftware

- Anpassen der Oberflächenprogramme

Falls Sie Ihre eigenen Anwenderprogramme schreiben, müssen Sie einige Änderung vornehmen, sonst laufen Ihre Programme mit der neuen Treiberversion nicht mehr zusammen. Alle notwendigen Informationen für Ihre Änderungsarbeiten finden Sie in Kapitel 2.8 der Hardware-Beschreibung und in Kapitel 3.2.15 der Software-Beschreibung.

Im Fall, dass Sie unsere Oberflächensoftware wie z. B. die Programme HPREMOTE, REMOTE, PRO-DIN, PRO-PAL etc. einsetzen, müssen Sie von uns die neuen Versionen anfordern. Unsere alte Oberflächensoftware läuft mit der neuen Treiberversion nicht zusammen.

7. Benutzung der alten Treiberversionen 3.00 oder 3.00A

Standardmäßig liefern wir nur noch die PC-Einsteckkarte und die dazugehörige Treibersoftware der Version 3.10 aus. Wenn Sie bei der alten Treiberversion 3.00 bzw. 3.00A bleiben wollen, müssen Sie zusätzlich noch den alten PLD-Baustein mit der Kennzeichnung SE_LIW2N und die alte Treibersoftware anfordern. Analog zu Punkt 6.1 können Sie den alten PLD-Baustein gegen den neuen PLD-Baustein umtauschen.

8. Benutzung der Treiberversion 3.20

Die Treiberversion 3.20 ist eine Software-Erweiterung im Vergleich zu der Treiberversion 3.10. Bei der Version 3.20 besteht unter anderem die Möglichkeit, das unterschiedliche Motor-Beschleunigungsverhalten im niedrigen und hohen Drehzahlbereich zu berücksichtigen. Es gibt keinen hardwaremäßigen Unterschied zwischen diesen beiden Versionen. Das Umsteigen von der Treiberversion 3.10 auf 3.20 geschieht sehr einfach, indem Sie die Initialisierungsdatei mit dem Programm PARKON.EXE in die neuere Version konvertieren. Das Umsteigen von der Treiberversion 3.00 und 3.00A auf der Version 3.20 ist ähnlich wie beim Übergang zu der Version 3.10.

Inhaltsverzeichnis

Hardware-Beschreibung

Software-Beschreibung

PARKON.EXE

PAREIN.EXE

Hardware-Beschreibung

1 Einleitung

1.1 Schrittmotor oder Servomotor?

Zuerst möchten wir Ihnen die wichtigsten Unterschiede zwischen einem Steuerungssystem für Schrittmotoren und einem solchen für Servomotoren erläutern.
Wenn die Schrittmotoren in definierten Arbeitsbereichen betrieben werden, kann eine Rückmeldung der Verfahrdaten entfallen. Es ergibt sich somit eine offene Steuerkette, die sehr einfach zu handhaben ist (siehe Bild 1.1). Das ist der entscheidende Grund für die starke Verbreitung der Schrittmotoren. Außerdem sind Schrittmotoren wesentlich billiger als Servomotoren.

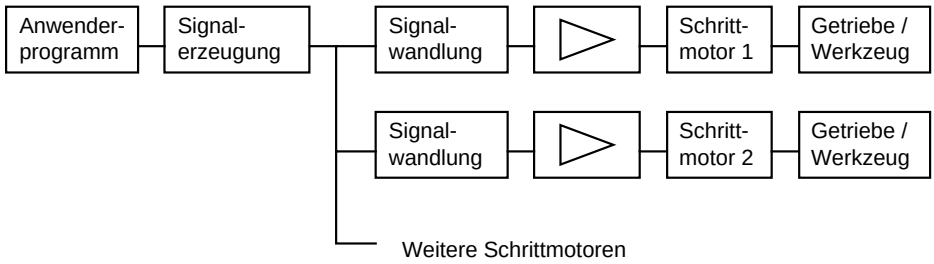


Bild 1.1: Schrittmotoransteuerung als offene Steuerketten

Im Bild 1.1 können Sie das Beispiel eines Steuerungssystems für mehrere Schrittmotoren sehen. Zum Funktionsblock *Signalerzeugung* gehören u. a. die Rampengenerierung, die Interpolation und die Synchronisation der Motoren. Die Ausgangssignale dieses Funktionsblocks sind meistens Impuls und Richtungen für die einzelnen Motoren. Der Funktionsblock *Signalwandlung* benutzt diese Signale, um die Ströme in den einzelnen Motorspulen ein- bzw. auszuschalten und bei Mikroschrittbetrieb außerdem noch die Stromstärke zu ändern. Die offene Steuerkette ermöglicht eine sehr einfache Inbetriebnahme des Steuerungssystems. Sie kaufen, unter Berücksichtigung der Belastbarkeit der Motoren und der Leistungsendstufen, ein Schrittmotor-Steuerungssystem, an das Sie nur noch an Ihre Anlage anzuschließen brauchen und schon haben Sie ein funktionsfähiges Antriebssystem. Einfacher geht es wirklich nicht mehr.

Bei einem Steuerungssystem für Servomotoren ist alles nicht mehr so einfach. Bild 1.2 zeigt Ihnen ein Steuerungssystem mit Servomotoren.

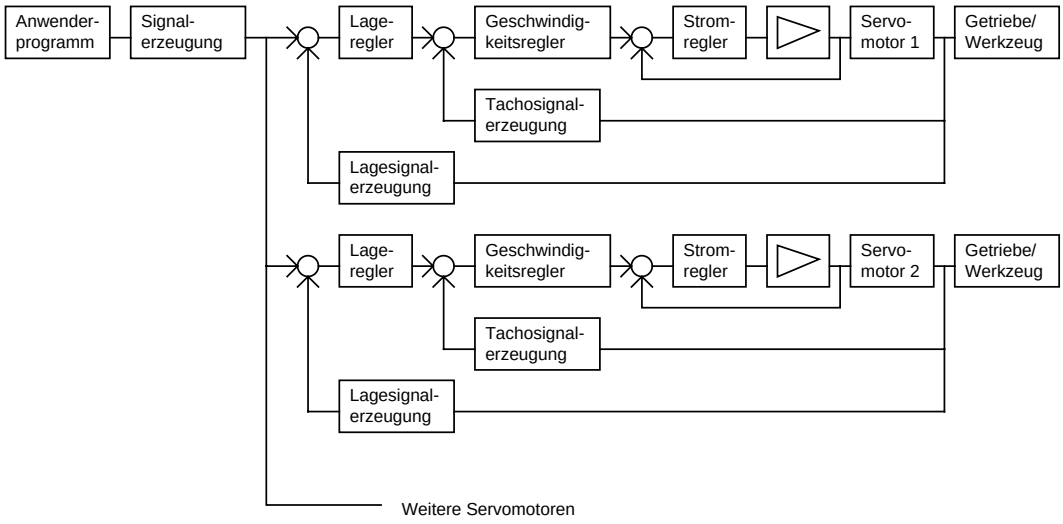


Bild 1.2: Servomotorsteuerung mit geschlossenen Regelkreisen

Das Blockschaltbild einer Servomotorsteuerung ist viel komplexer als das einer Schrittmotorsteuerung. Der Funktionsblock *Signalerzeugung* hat hier die gleichen Aufgaben wie bei den Schrittmotoren und liefert meistens die Lageinformationen als Ausgangssignale. Die Servomotoren werden in Regelkreisen über die Rückführung der Ausgangsinformationen gesteuert.
Der äußere Regelkreis ist der Lageregelkreis, der für das Positionieren zwingend notwendig ist. Aus dem Vergleich zwischen der Soll- und der Ist-Position bildet der Lageregler den Sollwert der Geschwindigkeit. Der Geschwindigkeitsregler für das Positionieren ist nicht unbedingt notwendig. Dieser Regelkreises ist optional, er bringt Ihnen aber einen großen Gewinn in Bezug auf die Dynamik und den Gleichlauf des Antriebssystems. Dies ist vor allem bei hochwertigen Antriebssystemen üblich. Es gibt aber auch Systeme, bei denen der Geschwindigkeitsregelkreis für die Systemstabilität notwendig ist.
Im Gegensatz zu dem Lageregler, der digital arbeitet, wird der Geschwindigkeitsregler meist analog realisiert. Der Ausgang des Geschwindigkeitsreglers dient wiederum als Eingang für den Stromregler (d. h. den Strom-Soll-Wert). Dieser Stromregler ist meist schon in die Leistungsendstufe integriert.
Eine Reglerstruktur wie im Bild 1.2 nennt man in der Regelungstechnik auch einen Kaskadenregler.
Für die Regelung braucht man immer Messsysteme, um die Ausgangssignale erfassen zu können. Das ist u. a. ein Grund, warum ein Antriebssystem mit Servomotoren teuerer ist als eines mit Schrittmotoren. Man kann aber auch die Geschwindigkeitsinformationen aus den Lageinformationen durch entsprechende Soft- oder Hardwarelösungen ableiten und so das Messsystem für die Geschwindigkeitserfassung ersetzen.

Die Qualität eines Servoantriebssystems hängt sehr stark von den benutzten Reglerstrukturen und von der Dimensionierung der Reglerparameter ab.

Aufgrund der Besonderheiten eines geschlossenen Regelkreises kann es bei ungünstiger Wahl unter Umständen zur Instabilität des Systems führen.
Die Auswahl der Reglerstruktur ist nicht schwierig. Abgesehen von einigen exotischen Strukturen, wie z. B. der Zustandsregler, können Sie immer wieder auf die schon lange in der Praxis bewährte PID-Struktur zurückgreifen.
Dagegen ist die Dimensionierung der Reglerparameter eine Wissenschaft für sich. (Aber auch hier lassen wir Sie nicht im Regen stehen. Sie bekommen von uns das Einstellprogramm PAREIN.EXE, das Sie bei der Inbetriebnahme Ihrer Servoanlage einschließlich beim Dimensionieren der PID-Regler unterstützt.)

Wo liegen die Stärken des Servomotors bzw. wo liegen die Schwächen des Schrittmotors?
Die größte Stärke eines Schrittmotors (die offene Steuerkette) ist gleichzeitig seine größte Schwäche. Weil es keine Kontrolle über die Anzahl der tatsächlich ausgeführten Schritte gibt, muss man darauf achten, dass ein Schrittmotor nur deutlich unterhalb seiner Leistungsgrenze betrieben werden darf, da es sonst zu Schrittverlusten kommt. In der Praxis treten diese Schrittverluste häufig auf. Die Gefahr ist besonders groß beim starken Beschleunigen der Anlage.
Eine weitere Schwäche tritt bei kleinen Geschwindigkeiten zu Tage, da Schrittmotoren hier deutlich ruckartige Bewegungen ausführen.
Schrittmotoren eignen sich zudem nicht für hohe Drehzahlen, weil die Schrittverluste in solchen Fällen sehr zunehmen. Außerdem ist es aufgrund des lauten Geräusches sehr unangenehm, sich in der Nähe eines Antriebssystems mit Schrittmotoren aufzuhalten. Weitere Nachteile, wie die starke Neigung zu Resonanz, der kleine Wirkungsgrad, der schlechte Gleichlauf usw. sprechen für den zunehmenden Einsatz von Servomotoren.

1.2 Der Unterschied zwischen einem Handlingsystem und einer CNC-Bearbeitungseinheit bezüglich der Steuerung

Bei einem Handlingsystem geht es um die Bewegung von einem Punkt zu einem anderen Punkt im Raum. Wie das geschieht, interessiert nicht weiter und es gibt dabei keine Synchronisation zwischen den Achsen. Das kleine Beispiel im Bild 1.3 soll Ihnen verdeutlichen, worum es geht.

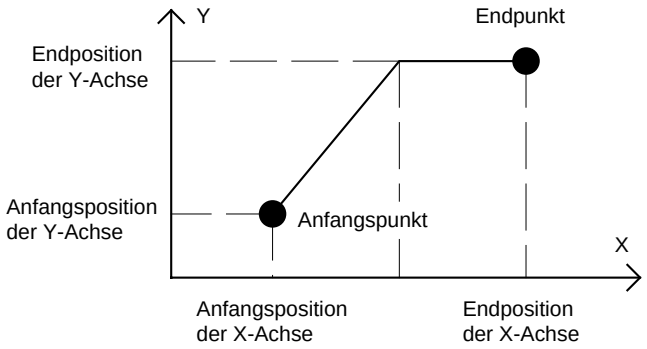


Bild 1.3: PTP-Bewegungsablauf am Beispiel eines Handlingsystem mit zwei Achsen

Während hier die Y-Achse ihre Endposition schon erreicht hat, ist die X-Achse erst auf dem Weg dahin. Dies ist ein typischer Bewegungsablauf bei einem Handlingsystem. Eine solche Art der Steuerung nennt man PTP-Steuerung (**P**oint **T**o **P**oint).

Neben der einfachen PTP-Steuerung gibt es noch die Synchron-PTP-Steuerung (siehe Bild 1.4).

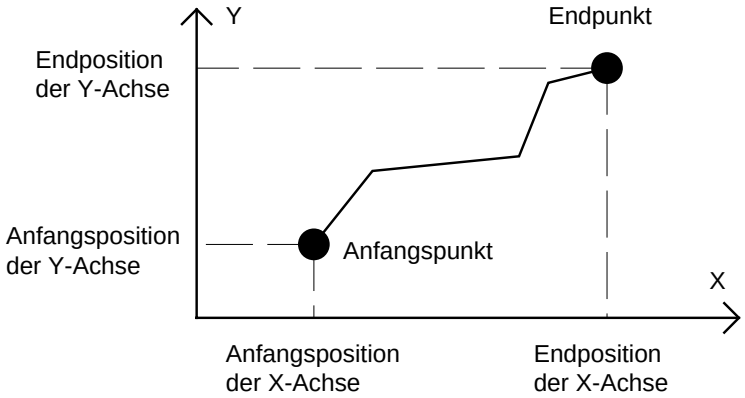


Bild 1.4: Synchron-PTP-Bewegungsablaufes am Beispiel eines Handlingsystems mit zwei Achsen

Durch das - zugegeben ein wenig übertriebene - Beispiel im Bild 1.4 können Sie deutlich das Hauptmerkmal einer Synchron-PTP-Steuerung erkennen. Alle Achsen erreichen gleichzeitig ihre Endpositionen. Dabei ist es unerheblich, wie sich das Werkzeug (z. B. ein Greifer) im Raum bewegt. Das Werkzeug bewegt sich vom Anfangspunkt zum Endpunkt auf einer nicht definierten Kurve im Raum. Die einfache PTP-Steuerung und die Synchron-PTP-Steuerung sind die Hauptsteuerungsarten für Handlingsysteme.

Bei einer CNC-Steuerung ist das nicht mehr so einfach. Hier wird gefordert, dass sich das Werkzeug vom Anfangspunkt zum Endpunkt auf einer vorher definierten Kurve bewegt (siehe Bild 1.5). Man spricht hier von Interpolation. Die zur Interpolation meist benutzten Kurven sind die Gerade (Linear-interpolation) und der Kreis (Kreisinterpolation).

Bei der Kreisinterpolation müssen wir aber noch ergänzen, dass es sich meistens um einen Kreis auf einer der Hauptebenen XY, YZ und XZ handelt. Falls sich die anderen Achsen, die nicht zu der Hauptebene gehören, während der Kreisinterpolation mit bewegen, entsteht die Helixinterpolation. Man kann also sagen, dass die Kreisinterpolation ein Sonderfall der Helixinterpolation ist.

Andere Interpolationsarten, wie z. B. die elliptische Interpolation oder die dreidimensionale Kreisinterpolation sind nicht besonders gebräuchlich, obwohl sie nicht wesentlich schwerer zu realisieren wären.

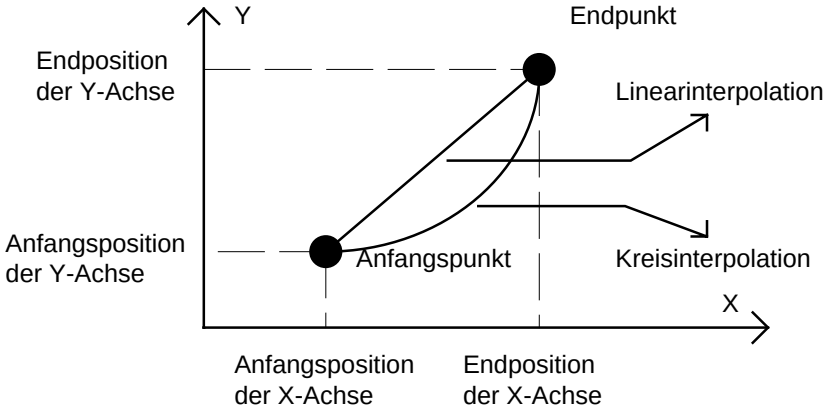


Bild 1.5: Linear- und Kreisinterpolation am Beispiel einer CNC-Bearbeitungseinheit mit zwei Achsen

Im Vergleich zur CNC-Steuerung ist die PTP-Steuerung ein Kinderspiel, weil die Interpolation viel schwerer zu realisieren ist. Außerdem erfordern die unterschiedlichen Strukturen von CNC-Maschinen verschiedene Interpolationsalgorithmen mit unterschiedlichen Rechenaufwänden. Die meisten CNC-Maschinen haben die sogenannte TTT-Struktur, d. h. die Hauptachsen X , Y und Z sind Linearachsen und stellen ein kartesisches Koordinatensystem im Raum dar. Der Buchstabe "T" steht für das englische Wort "Translate".
Für andere Strukturen, deren Hauptachsen teils Linearachsen und teils Drehachsen sind, verlangen die Interpolationsalgorithmen eine immense Rechenleistung, weil man hier verschiedene Transformationen in Echtzeit durchführen muss, um die Synchronisation zwischen den Achsen zu realisieren.

1.3 Lieferumfang

Im Rahmen dieses Produkts liefern wir Ihnen:

- PC-Einsteckkarte mit AT-Bus
- Treiberprogramm ISELDRV.EXE
- Initialisierungsprogramm ISELINI.EXE
- Assemblerroutine KON_INS.ASM mit der entsprechenden Objektdatei KON_INS.OBJ
- Konfigurationsprogramm PARKON.EXE
- Initialisierungsdatei EXAMPLE.INI als Beispiel
- Einstellprogramm PAREIN.EXE
- Installationsprogramm INSTALL.BAT mit den dazu gehörigen Hilfsdateien ISELINST.EXE und ISELDRV.CFG
- Hilfsdatei _ENTER_
- Objektfiles BAHN_S.OBJ, BAHN_M.OBJ, BAHN_C.OBJ und BAHN_L.OBJ sowie das ausführbare Programm BAHN.EXE zur Berechnung der Bahndaten für die 3D-Bahnberabeitung (optional erhältlich)
- Handbuch

Unser Produkt ist für die Steuerungen von bis zu vier Servomotoren über Leistungsendstufen mit dem ± 10 V-Eingang konzipiert.

Die PC-Einsteckkarte benötigt in dem PC einen ISA-16-Bit-Slot. Für jeden Motor ist bereits ein digitaler PID-Lageregler auf der PC-Einsteckkarte implementiert. Außerdem hat die PC-Einsteckkarte noch frei programmierbare 16-Bit-Timer sowie getrennte Ein- und Ausgänge, die Sie für SPS-Funktionen oder End- bzw. Referenzschalter nutzen können.
Abgesehen von den digitalen Lagereglern, hat die Einsteckkarte keine eigene Intelligenz (siehe Kapitel 1.4).

Ein 50-poliger RIBBON-Steckverbinder dient als Schnittstelle zur Außenwelt. Das Treiberprogramm, das als ein TSR-Programm (Terminated Stay Resident) ausgeführt ist, können Sie auf einem PC mit dem MS/DOS-Betriebssystem ab Version 4.1 installieren. Auf einem PC mit WINDOWS 3.xx bzw. WINDOWS 95 oder OS/2 können Sie das Treiberprogramm ISELDRV.EXE ohne weiteres in einem *DOS-Vollbild-Task* benutzen.

Der Treiber übernimmt die Funktionen des Blocks *Signalerzeugung* (siehe Bild 1.1 und 1.2). Die 4-Achsen-Linear- und die 2-Achsen-Kreis- sowie die Helixinterpolation sind für Anlagen mit einer TTT-Struktur. Für alle anderen Strukturen können Sie mit dem Treiber problemlos eine Synchron-PTP-Steuerung realisieren. Jede Achse kann entweder als eine Dreh- oder als eine Linearachse konfiguriert sein.

Um Ihnen die Arbeit zu erleichtern, stellt der Treiber noch verschiedene zusätzliche Funktionen zur Verfügung. Mit diesen Funktionen sind Sie in der Lage, mühelos eine CNC-Anlage oder ein Positioniersystem zu realisieren.

Das Initialisierungsprogramm ISELINI.EXE ist ein Hilfsprogramm für den Treiber. Es wird vom Treiber beim Installieren aufgerufen, um die Anlagenparameter aus einer Initialisierungsdatei zu lesen. Auf diese Art und Weise können wir den residenten Treiber klein halten.

Die Assemblerroutine KON_INS.ASM mit der Objektdatei KON_INS.OBJ können Sie in Ihrem Anwenderprogramm verwenden, um festzustellen, ob unser Treiber in Ihrem PC schon installiert ist oder nicht und über welchen Interrupt der Treiber angesprochen werden kann.

Das Konfigurationsprogramm PARKON.EXE hilft Ihnen bei der Anpassung des Treibers an Ihre Anlage (siehe das entsprechende Handbuch).

Die mitgelieferte Initialisierungsdatei EXAMPLE.INI dient ausschließlich als Beispiel für Sie.

Das Einstellprogramm PAREIN.EXE ist ein wertvolles Hilfsmittel während der Inbetriebnahme einer Servoanlage. Mit diesem Programm können Sie unter anderem sehr schnell die Reglerparameter dimensionieren oder die Rampen der einzelnen Achsen bestimmen (siehe im entsprechenden Handbuch).

Das Installationsprogramm INSTALL.BAT mit den Hilfsdateien ISELINST.EXE und ISELDRV.CFG installiert unsere Software auf Ihrem PC. Wobei wird ein Verzeichnis mit dem Name „SERVO“ erstellt. In diesen Verzeichnis werden alle benötigten Dateien automatisch herein kopiert.

Die Hilfsdatei _ENTER_ ist eine einfache Datei mit dem einzigen Zeichen ENTER. Im Zusammenhang mit dem Umleitungszeichen „<“ bildet diese Datei das Bestätigen der Taste ENTER nach. Damit kann der Treiber ohne ein Bestätigen der Taste ENTER installiert werden. Diese Art des Starts ist sehr sinnvoll, falls der Treiber in der Datei AUTOEXEC.BAT platziert werden soll.

Mit den optional erhältlichen Objektdateien BAHN_S.OBJ, BAHN_M.OBJ, BAHN_C.OBJ und BAHN_L.OBJ sowie dem ausführbaren Programm BAHN.EXE können Sie die notwendigen Bahndaten für eine 3D-Bahnbearbeitung erzeugen.
Sie können Ihr eigenes Anwenderprogramm schreiben, um dem Treiber zu sagen, was er für Sie machen soll. Die Kommunikation zwischen Ihrem Programm und dem Treiber erfolgt über einen von Ihnen selbst definierten Interrupt.

1.4 Warum diese Konzeption?

Das Bild 1.6 zeigt Ihnen, wie Ihr Programm und Ihre Anlage zusammen mit unserem Treiber und unserer PC-Einsteckkarte arbeiten werden.

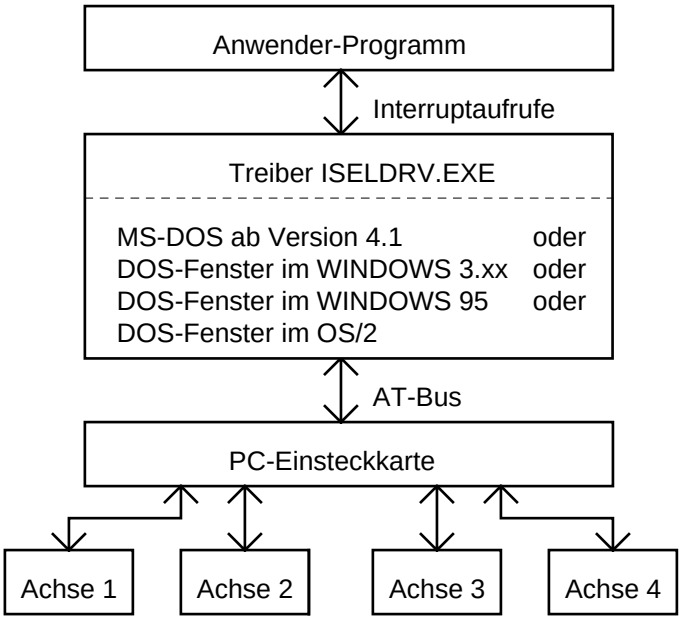


Bild 1.6: Funktionsstruktur einer mit dem Treiber gesteuerten Anlage

Unser Treiber ist ein MS-DOS-Programm. Im Zeitalter von WINDOWS und OS/2 scheint ein MS-DOS-Programm ein bisschen altmodisch zu sein. Aber wegen der starken Verbreitung von MS-DOS ist jedes neue PC-Betriebssystem gezwungen, die Kompatibilität zu MS-DOS zu gewährleisten. Aus diesem Grund können Sie ohne Problem unseren Treiber auf jedem PC mit den gängigen Betriebssystemen wie WINDOWS 3.xx oder WINDOWS 95 oder OS/2 installieren. Die Installation muss aber in einem DOS-Vollbild-Task erfolgen.

Jeder Programmierer hat seine Lieblingssprache. Um diese Vielfalt zu berücksichtigen sowie um einen reibungslosen Datenaustausch zwischen einem Anwenderprogramm und unserer Software zu gewährleisten, haben wir unsere Software als ein residentes Programm ausgeführt, mit dem ein Anwenderprogramm die Daten über Interrupt-Aufrufe austauscht.

Die Benutzung der Interrupt-Aufrufe für die Kommunikation ist eine übliche Methode bei der Programmierung unter MS-DOS. Fast alle Programme, die die Ressourcen vom BIOS oder von MS-DOS nutzen wollen, müssen auch entsprechende Interrupts aufrufen. In C , PASCAL oder TURBO BASIC haben Sie bereits entsprechende Befehle, um einen Interrupt aufzurufen. In anderen Programmiersprachen, die solche Befehle nicht zur Verfügung stellen, müssen Sie entsprechende Ersatzroutinen benutzen, um Interrupts aufrufen zu können.

Die PC-Einsteckkarte besitzt keine eigene Intelligenz. Die notwendige Intelligenz, die eine CNC-Maschine oder ein Positioniersystem für die Interpolation, die Koordinierung usw. braucht, wird vom Treiber durch die Nutzung der PC-Ressourcen geschaffen.

Warum sind wir diesen Weg gegangen?
Der PC ist heutzutage eine Massenware und wird immer billiger. Im Gegenzug dazu nimmt seine Leistungsfähigkeit ständig zu. Angenommen, unsere PC-Einsteckkarte hätte eine eigene Intelligenz und Interpolation, Koordinierung etc. würden ausschließlich mit dieser Intelligenz durchgeführt, wäre der PC nur noch für die Kommunikation zwischen dem Bediener und der Maschine zuständig. Das wäre natürlich reine Verschwendung. Mit dem PC brauchen Sie sich außerdem keine Gedanken darüber zu machen, ob die Leistungsfähigkeit Ihres PC ausreicht oder nicht, eine CNC-Maschine oder ein Positioniersystem zu steuern.
Bei einem 286/15 MHz arbeitet unser Treiber zusammen mit unserer Benutzeroberfläche, selbst während der Interpolationsphase für vier Achsen, zeitlich gesehen problemlos!

Die Nutzung der PC-Intelligenz bringt Ihnen und uns noch einen anderen großen Vorteil. Für eine zukünftige Erweiterung werden wir nicht mehr durch irgendeine schwer erweiterbare Intelligenz auf der PC-Einsteckkarte begrenzt. Einzig und allein die PC-Leistungsfähigkeit ist unsere obere Grenze.

2 Die PC-Einsteckkarte

2.1 Technische Daten und das Layout der PC-Einsteckkarte

- Vier PID-Regler mit einer Abtastzeit von 0,35 ms und 12-Bit-D/A-Wandler zur Ansteuerung von bis zu vier Servoachsen über ± 10 V-Signalen
- Spannungsquelle von 5 V oder 12 V (z. B. für Encoder, ...)
- TTL-, oder RS422-Interface für Inkrementalgeber (Encoder)
- 32-Bit Positions-, Geschwindigkeits- und Beschleunigungsregister
- 32-Bit Ist-Positionserfassung
- 14 optoisolierte Eingänge (z. B. für Endlagen-, Referenzschalter, ...)
- 3 optoisolierte Ausgänge
- 50-poliger RIBBON Stecker

Die PC-Einsteckkarte ist der Hardwareteil unseres Produktes. Sie ist für den Betrieb von bis zu vier Achsen vorgesehen. Als Ausgangssignal wird ein Einheitssignal von ± 10 V mit einer Auflösung von 12-Bit für die Ansteuerung nachfolgender Leistungsverstärker zur Verfügung gestellt.
Alle Komponenten zur Auswertung der für eine Rückkopplung notwendigen Encoder-Signale sind ebenfalls auf der Einsteckkarte vorhanden.
Zur Verbindung mit der Peripherie werden 14 optoisolierte Eingänge und 3 optoisolierte Ausgänge für einen 24 V-Betrieb bereitgestellt. Prinzipiell kann mit einem auf der Karte vorhandenen Timer-Baustein ein Real-Time-Clock für allgemeine Anwendungen genutzt werden. Der Real-Time-Clock ist interruptfähig. Ein Interrupt-Betrieb der LM-Chip ist hardwaremäßig ebenfalls möglich.

Bitte beachten Sie, dass die Timerkanäle sowie ein Teil der Ein- und Ausgänge bei Verwendung des Treibers ISELDRV.EXE intern benutzt werden und Ihnen nicht mehr zur Verfügung stehen.
Unsere Einsteckkarte arbeitet nur mit inkrementalen Encodern. Sie können Encoder mit einer Versorgungsspannung von 5 V oder 12 V benutzen. Die Signale können TTL- oder RS422-Signale sein.

Zum Einbau innerhalb eines Steuerrechners ist ein ISA-16-Bit-Slot erforderlich. Wir haben unsere Karte so gebaut, um ein möglichst großes Spektrum von Anwendungsfällen abdecken zu können. Sie als Anwender müssen die Karte durch das Einstellen der Jumper entsprechend Ihren Aufgabenstellungen und Ihrem System konfigurieren. Die Positionen von allen für Sie wichtigen Bauteilen zeigen wir Ihnen im Bild 2.1.

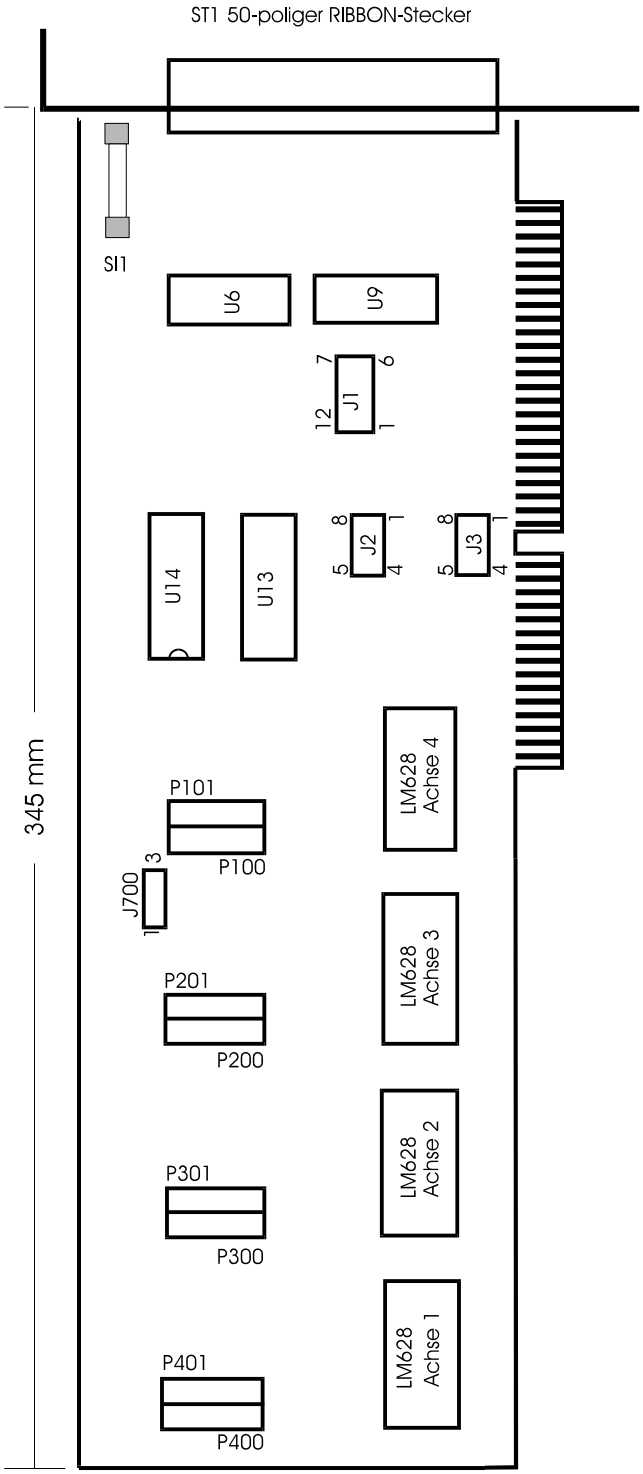


Bild 2.1: Layout der PC-Einsteckkarte

Überblick über die verschiedenen Jumper, Stecker und Potentiometer.

Name	Bez.	Aufgabe
Potentiometer	P100	Offset der 1. Verstärkerstufe X-Achse
Potentiometer	P200	Offset der 1. Verstärkerstufe Y-Achse
Potentiometer	P300	Offset der 1. Verstärkerstufe Z-Achse
Potentiometer	P400	Offset der 1. Verstärkerstufe A-Achse
Potentiometer	P101	Offset der 2. Verstärkerstufe X-Achse
Potentiometer	P201	Offset der 2. Verstärkerstufe Y-Achse
Potentiometer	P301	Offset der 2. Verstärkerstufe Z-Achse
Potentiometer	P401	Offset der 2. Verstärkerstufe A-Achse
Jumper	J1	Basisadresse der Karte
Jumper	J2	Interrupt Timerbaustein
Jumper	J3	Interrupt LM628-Baustein
Jumper	J700	Inkrementalgeberversorgung 5 V bzw. 12 V
	U6	Eingabeport
	U9	Eingabeport
	U13	Ausgabeport
	U14	PLD
Stecker	ST1	50-poliger RIBBON-Stecker

2.2 Die Basisadresse der PC-Einsteckkarte und die einzelnen Portadressen

Die Einsteckkarte belegt einen Ein/Ausgabe-Portadressenbereich von 0Fh im PC. Die Achskontroller, Timer sowie die Ein- und Ausgabeports können Sie durch entsprechende Portadressen ansprechen. Durch den Jumper J1 legen Sie eine Basisadresse für die Karte fest. Damit werden die Adressen aller Ports auf der Karte definiert. Beim Erzeugen der Initialisierungsdatei durch das Konfigurationsprogramm PARKON (siehe Handbuch dafür) brauchen Sie nur diese Basisadresse einzugeben. Daraus berechnet der Treiber alle für ihn notwendigen Portadressen selbst.

Zum Einstellen der Basisadresse an Jumper J1 müssen Sie wissen, welcher Adressenraum in Ihrem Rechner noch frei ist. Sonst kann es zu einem Adressenkonflikt zwischen unserer Einsteckkarte und den anderen Karten im PC kommen. Die Reset-Taste hat dann wieder mal darunter zu leiden.
Hier können wir Ihnen einen Tipp geben:
In einem AT-Rechner ist der Adressbereich 300h - 31Fh immer für Anwender reserviert. Falls Sie keine andere zusätzliche Einsteckkarte im Rechner haben, können Sie diesen Adressbereich sorglos nutzen.

Das Einstellen der Basisadresse am Jumper J1 ist im Folgenden erläutert.
(alle angegebenen Adressbezüge beziehen sich auf Hexadezimal-Darstellung).
Beispiel der Basisadresse 300h:

J1

0 : Pin offen
C : Pin gesteckt

A9	A8	A7	A6	A5	A4
0	0	C	C	C	C

Alle folgenden Beschreibungen beziehen sich auf die bei Auslieferung eingestellte Basisadresse 300h!

LM-Servo-Regler:

A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Belegung	Adr.
Basisadresse						0	0	0	0	LM1	300
						0	0	0	1	LM1	301
						0	0	1	0	LM2	302
						0	0	1	1	LM2	303
						0	1	0	0	LM3	304
						0	1	0	1	LM3	305
						0	1	1	0	LM4	306
						0	1	1	1	LM4	307

Bei der Adressierung der LM-Servo-Bausteine gilt:

--> geradzahlige Adresse Kommando-Byte z. B. Adr. 300h
ungeradzahlige Adresse Daten-Byte Adr. 301h

Timerbaustein I82C54

A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Belegung	Adr.
Basisadresse						1	0	0	0	Kanal 0	308
						1	0	0	1	Kanal 1	309
						1	0	1	0	Kanal 2	30A
						1	0	1	1	Steuerport	30B

Ein-/Ausgabeport

A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Belegung	Adr.
						1	1	0	0		30C
Basisadresse						1	1	0	1		30D

Die Adresse 30C und 30D haben bei einem Schreib-/ Lesezugriff auf die Ein-/ Ausgabeports die gleiche Bedeutung.

Adresse 30C/30D schreiben Ausgabeport U13
Adresse 30C/30D lesen Eingabeport U6

A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Belegung	Adr.
						1	1	1	0		30E
Basisadresse						1	1	1	1		30F

Die Adressen 30E und 30F haben beim Lesezugriff die gleiche Bedeutung.

Adresse 30E/30F lesen Eingabeport U9

Ein Schreibzugriff auf die Adressen 30E/30F hat keine Wirkung, da physikalisch kein Ausgabeport vorhanden ist.

2.3 Die Achscontroller

Für jede Achse steht ein Achscontroller vom Typ LM628 (Hersteller: National Semiconductor) auf der PC-Einsteckkarte zur Verfügung. Der LM-Chip wird mit einer Quarzfrequenz von 6 MHz versorgt. Der Chip LM628 erzeugt über einen nachgeschalteten 12-Bit D/A-Wandler ein ± 10 V Stellsignal.
Bei Bedarf ist der Einbau des LM629 möglich. Dieser Baustein ermöglicht statt der oben ± 10 V-Schnittstelle den Betrieb mit einem PWM-Signal und einem zugeordneten Direction-Bit (positive oder negative Richtungswahl). In dieser Version ist die Beschaltung des D/A-Wandlerparts nicht notwendig (optionale Bestellung).
Die Adressen der einzelnen Achscontroller können Sie im vorherigen Kapitel nachlesen. Die Positionen der einzelnen Achscontroller auf der Karte sind in Bild 2.1 gezeigt.
Die vom LM628 zur Auswertung der Ist-Positionen benötigten Signale, die von einem am Stellglied angebrachten Encoder geliefert werden müssen, werden auf der PC-Karte entsprechend aufbereitet und dem LM628 zur Verfügung gestellt. Bei Auslieferung der PC-Karte ist die Beschaltung der Encoder-Kanäle für die im Hause ***isel****automation* beziehbaren Servomotoren mit integriertem Encoder mit RS422 Ausgang vorbereitet. Die Beschaltung der Empfänger für die Encoder-Signalaufbereitung kann auch für massebezogene Encoder-Ausgänge ohne zusätzlich gelieferte Invers-Signale erfolgen.
Die maximale Encoder-Signalspannung darf 12 V nicht überschreiten.

Die Ausgänge der Servoachsen sind durch interne Verstärker-Einstellungen auf ± 10 V normiert. Eine Offset-Einstellung ist für die erste Analog-Verstärkerstufe mittels Potentiometer möglich.

z. B. X-Achse P100 für Offset-Einstellung;
Y-Achse P200 für Offset-Einstellung;

Der Nullpunkt der zweiten Verstärkerstufe kann ebenfalls durch Einstellung an dem der jeweiligen Achse zugeordneten Potentiometer erfolgen.

z. B. X-Achse P101 Einstellung der Steilheit;
Y-Achse P201 Einstellung der Steilheit;

Das klingt etwas zeitaufwendig, ist aber nur für eventuelle Nachregelungen erforderlich, da wir Offset und Referenzspannung für Sie bereits ab Werk eingestellt haben.

- Änderungen sind fachgerecht durchzuführen.
- Unsachgemäßer Umgang mit der Einsteckkarte kann zu nachhaltigen Beeinträchtigungen der Funktionen führen!
- Wenn Sie Änderungen an der Einsteckkarte vorgenommen haben und der Originalzustand nicht mehr beibehalten wurde, können Sie keine Garantieansprüche mehr geltend machen!

Durch den Jumper J700 kann eine Inkrementalgeber-Versorgung von 5 V bzw. 12 V eingestellt werden. Für die über *iselautomation* beziehbaren DC-Servomotoren ist zwingend die Geberversorgung auf 5 V zu jumpern. Bevor Sie den Geber der Servomotoren an den Peripherie-Steckverbinder anstecken, prüfen Sie die an der Sicherung SI1 anliegende Spannung!
Die am Peripherie-Steckverbinder zur Verfügung gestellte Spannung zur Geberversorgung wird durch das PC-Netzteil zur Verfügung gestellt und ist auf der PLUS-Versorgungsseite durch eine Sicherung SI1 abgesichert.

J700	Brücke 2-1	5 V Geberversorgung
	Brücke 2-3	12 V Geberversorgung

Gegen auftretende Transienten sollten vor dem Steckverbinder entsprechende Schutzmaßnahmen vorgesehen werden! Intern ist ein Schutz der über die Sicherung SI1 nach außen zur Verfügung gestellten Geberversorgung vorgesehen.

Die negierten Signale der Encoder-Signale A, B und Index sind nicht unbedingt erforderlich. Aber ihr Vorhandensein kann die Sicherheit der Datenübertragung zwischen Encodern und Achscontrollern wesentlich erhöhen. Das Gleiche gilt auch, wenn Sie Encoder mit einer Versorgungsspannung von 12 V benutzen.

Nur ein abgeschirmtes Kabel zwischen der PC-Einsteckkarte und der Anlagen kann eine störungsfreie Datenübertragung garantieren.

Wenn Sie den Treiber ISELDRV.EXE nicht benutzen, besteht für Sie die Möglichkeit, die Achscontroller einzeln zu programmieren und zu benutzen. Bei der Programmierung der Achscontroller LM628 bzw. LM629 benutzen Sie bitte die entsprechenden Datenblätter der Firma National Semiconductor. Benutzen Sie den Jumper 3, wenn Sie wollen, dass die Achscontroller Interrupts auslösen (siehe Bild 2.2).

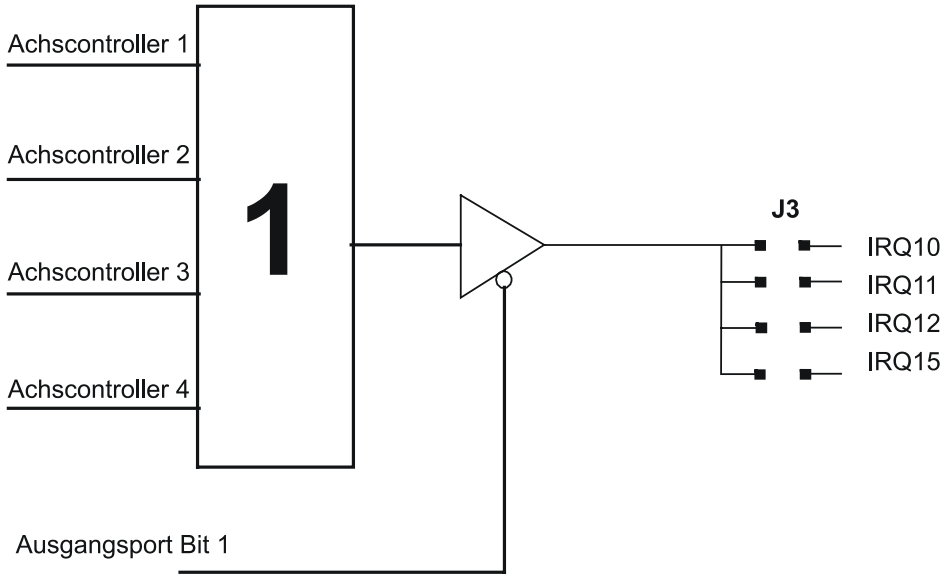


Bild 2.2: Interrupt-Ausgänge der Achscontroller

Durch den Jumper 3 können Sie den Interrupt für die Achscontroller wählen. Alle Achscontroller benutzen gemeinsam einen Interrupt. Durch das Auswerten der internen Statusregister der Achscontroller können Sie aber jederzeit bestimmen, welcher Achscontroller den Interrupt auslöst. Über das Datenbit 1 des Ausgabeports können Sie den Interrupt aus- oder einschalten. Der Timer auf unserer PC-Einsteckkarte ist auch in der Lage, Interrupts auszulösen (siehe Kapitel 2.4). Deswegen darf es bei der Konfiguration der Jumper 2 und 3 keinen Konflikt geben. Bei der Benutzung des Treibers ISELDRV.EXE dürfen Sie den Jumper 3 nicht stecken.

2.4 Die Timer, die Hardware-Interrupts und die glücklosen Besitzer von Pentium-Rechnern mit PCI-Bus

Für Zeitmessungen sowie Generierung zyklischer Timer-Interrupts steht auf der PC-Karte der Timerbaustein 8254 (Hersteller INTEL) zur Verfügung. Gemäß oberer Beschreibung belegt der Timer vier aufeinander folgende Adressen ab der eingestellten Basisadresse +08h, z. B. die Adressen 308h, 309h, 30Ah, 30Bh.

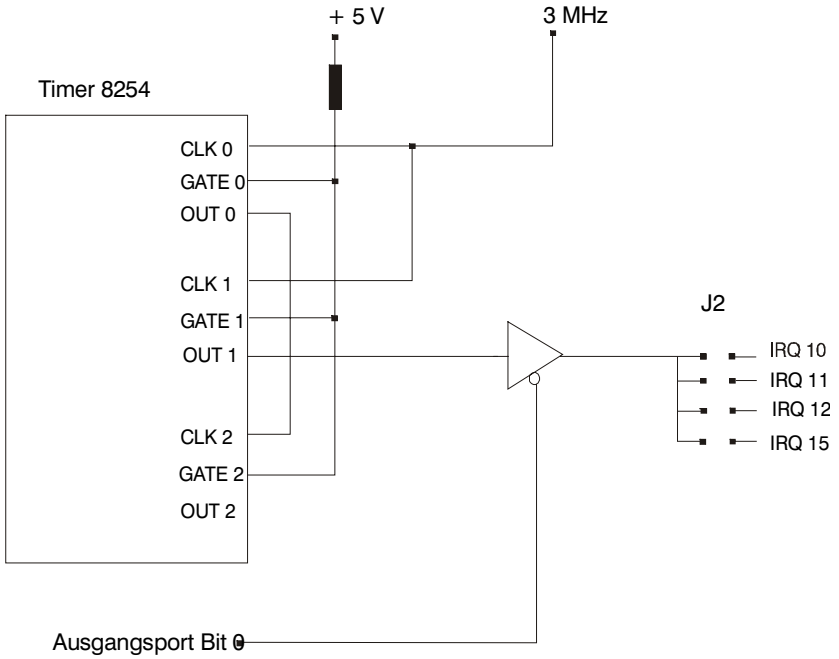


Bild 2.3: Blockschaltung des Timerbausteins 8254 auf der PC-Einsteckkarte

Der Timer ist für den internen Gebrauch bestimmt, falls Sie unseren Treiber ISELDRV.EXE benutzen. Der Treiber benutzt den Timer für die Generierung von Hardware-Interrupts und für die Zeitmessung. Sie haben die Wahl, den Ausgang des Timers mit IRQ10 (Pin 1 mit Pin 8) oder mit IRQ11 (Pin 2 mit Pin 7) zu verbinden. Die Einstellung des Hardware-Interrupts IRQ10 oder IRQ11 erfolgt an Jumper 2 und muss mit der Eingabe im Konfigurationsprogramm PARKON übereinstimmen (siehe entsprechendes Handbuch). Über das Datenbit 0 des Ausgabeports kann der Treiber den Hardware-Interrupt ein- oder ausschalten.
Die Hardware-Interrupt-Eingänge IRQ12 und IRQ15 stehen zwar zur Verfügung, dürfen von Ihnen aber nicht benutzt werden.

Beachten Sie, dass die Achscontroller auch in der Lage sind, Hardware-Interrupts auszulösen (siehe Kapitel 2.3). Deswegen darf es bei der Konfiguration der Jumper 2 und 3 keinen Konflikt geben.
Bei der Benutzung des Treiber ISELDRV.EXE dürfen Sie den Jumper 3 nicht stecken.

Der Timer 0 und der Timer 2 werden kaskadiert geschaltet. Der Clock-Eingang von Timer 0 wird mit einer Frequenz von 3 MHz versorgt. Die Timer 0 und 2 werden intern von dem Treiber ISELDRV.EXE für die Zeitmessung benutzt.
Der zyklische Interrupt, generiert vom IRQ10 oder IRQ11, ist zuständig für die Erzeugung des Interpolationstaktes innerhalb unseres Treibers ISELDRV.EXE. Falls dieser Interrupt aus irgendwelchen Gründen nicht ausgelöst werden kann, sind eine Bewegung, abgesehen von der Referenzfahrt, und die Ausgabe des Watch-Dog-Signals nicht möglich.

Für das Ausführen der Referenzfahrt benötigt der Treiber keinen zyklischen Interrupt. Beim Pentium-Rechner der neueren Generation mit PCI-Bus werden die IRQ's einschließlich IRQ10 und IRQ11 intern vom BIOS verwaltet. Hier im BIOS können Sie selbst definieren, ob Sie den einen oder den anderen IRQ benutzen wollen oder nicht. Standardmäßig werden IRQ10 und IRQ11 im BIOS gesperrt. D. h. um unsere PC-Einsteckkarte und den Treiber ISELDRV.EXE benutzen zu können, müssen Sie den BIOS Ihres Pentium-Rechners entsprechend konfigurieren.

2.5 Sicherheitsmaßnahmen eines Servo-Controllers

Um die Mechanik zu schonen, hat jede Achse einer Anlage normalerweise zwei Hardware-Endschalter, an jedem Ende einen.
Wenn einer dieser Endschalter betätigt ist, muss die Bewegung sofort unterbrochen werden. Bei einem Fehler der Hardware-Endschalter ist entweder das Ausschalten der Leistungsendstufen oder das Sperren des Stroms in der Richtung des aktiven Hardware-Schalters notwendig (siehe Kapitel 3.2.6.2).

Die zweite Methode ist sicherlich einfacher und besser. Sie können diese Methode aber nur benutzen, wenn die Leistungsendstufen mitmachen. Falls die Leistungsendstufen bei einem Hardware-Endschalter komplett ausgeschaltet werden müssen, müssen Sie dafür sorgen, dass die Leistungsendstufen wieder eingeschaltet werden können, um aus den Hardware-Endschaltern heraus fahren zu können. Das Gleiche gilt auch für die Referenzfahrt, falls Ihre Anlage aus Spargründen die Hardware-Endschalter als Referenzschalter benutzt.
Im Folgenden wollen wir eine der Möglichkeiten beschreiben, wie Sie dieses Problem lösen können (siehe Bild 2.4).

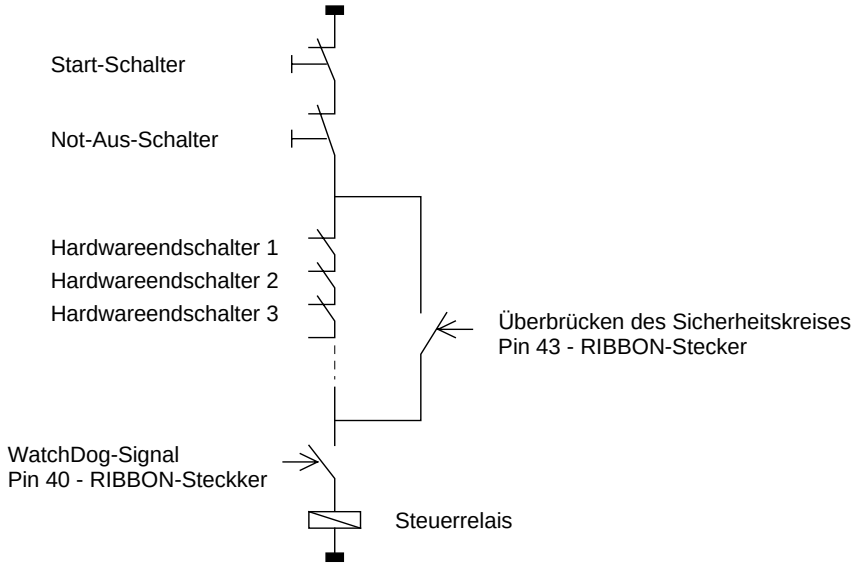


Bild 2.4: Prinzipschaltbild des Sicherheitskreises

Der Pin 43 des RIBBON-Steckers ist der optoisolierte Ausgang des Ausgabeports auf der PC-Einsteckkarte (siehe Kapitel 2.5). Sie können jederzeit einen Wert an diesem Pin ausgeben, um den Sicherheitskreis zu überbrücken, wie es im Bild 2.4 beschrieben ist. Wenn der Sicherheitskreis geschlossen ist, haben die Hardware-Endschalter keine Wirkung mehr. Das Einschalten der Leistungsendstufen ist möglich, selbst bei einem aktiven Hardware-Endschalter. Beachten Sie, dass die Überbrückung des Sicherheitskreises nur so lange dauert, wie es notwendig ist, d. h. so schnell wie möglich sollten Sie den entsprechenden Wert an dem Pin 43 ausgeben, um den Sicherheitskreis wieder zu aktivieren. Wenn Sie unseren Treiber benutzen, sollten Sie die Funktion 48 benutzen, um den Sicherheitskreis ein- bzw. auszuschalten (siehe Kapitel 3.3.48).

Der Pin 40 des RIBBON-Steckers ist auch ein optoisolierter Ausgang des Ausgabeports auf der PC-Einsteckkarte. Wenn Sie den Treiber benutzen, wird ein Watch-Dog-Signal regelmäßig an diesem Pin ausgegeben. Wenn Sie das Watch-Dog-Signal in den Sicherheitskreis so integrieren wie es im Bild 2.4 beschrieben ist, wird der Servocontroller sofort ausgeschaltet, wenn das Watch-Dog-Signal nicht mehr da ist. So was kann bei einem Softwareabsturz passieren. Wenn Sie den Treiber nicht benutzen, müssen Sie das Watch-Dog-Signal selbst generieren (siehe Kapitel 2.5), d. h. eine wechselnde Ausgabe von 0, 1, 0, 1 an diesem Ausgabeport müssen Sie selbst organisieren, um das Steuerrelais zum Anzug zu bringen.

Bei überbrückten Hardware-Endlageschaltern sind die Sicherheitsschalter der numerischen Achsen abgeschaltet!
Somit sind insbesondere alle Vorsichtsmaßnahmen der Unfallverhütungsvorschriften zu berücksichtigen!
Unsachgemäße Bedienung der Controller kann in dieser Betriebsart zu Beeinträchtigungen der Maschinenfunktion führen!

2.6 Die Belegung der Ein- und Ausgabeports

Die auf der Karte realisierten digitalen Ein-/Ausgänge werden galvanisch getrennt am 50-poligen RIBBON-Steckverbinder zur Verfügung gestellt. Es sind 14 optoisolierte Eingänge und 3 optoisolierte Ausgänge für 24 V-Betrieb vorhanden.
Aus der Belegung der Ein- und Ausgabeports können Sie sofort herausfinden, an welchem Pin des Peripheriesteckers das jeweilige Datenbit liegt. Beachten Sie, welche Ports intern vom Treiber benutzt werden und Ihnen nicht mehr zur Verfügung stehen.
Die Eingänge werden mit 24 V extern aufgeschaltet (intern = 1).
Bei den Ausgängen werden die Opto-Koppler mit max. 50 mA durchgeschaltet (offener Emitter).

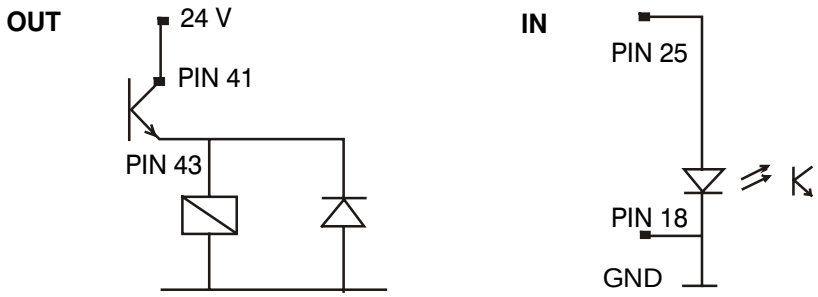


Bild 2.5: optoisolierte Ein- und Ausgänge auf der PC-Einsteckkarte

Im Folgenden werden die Bedeutung der einzelnen Ein- und Ausgabebits beschrieben. Die Belegung dieser Ein/Ausgabebits ist mit dem *isel*-Servocontroller verbunden. Andere Belegungen sind durchaus möglich.

Ausgabeport U13: (Basisadresse + 0Ch)

Das Ausgabeport U13 wird zur Systemsteuerung und für allgemeine Ausgaben zur Verfügung gestellt. Nach einem RESET und POWER-ON sind alle Ausgänge dieses Ausgabeports gelöscht.

Datenbit 0	Enable Timer Interrupt DB0 = 1
Datenbit 1	Enable (LM628 Achse 1- 4) Interrupt DB1 = 1
Datenbit 2	Watch-Dog Initialisierung 0/1 bzw. 1/0 Wechsel; innerhalb einer Zeitspanne von kleiner 1,8 s solange DB3 = 0, garantiert dieser Wechsel den Einzustand des Opto-Kopplers (Pin 43)
Datenbit 3	Sperre Watch-Dog DB3 b= 1 --> Opto-Koppler immer durchgesteuert
Datenbit 4	OUT 1 DB4 = 1 --> Opto-Koppler durchgesteuert Enable/ Disable Leistungsendstufen (Pin 16)
Datenbit 5	OUT 2 DB5 = 1 --> Opto-Koppler durchgesteuert Überbrücken des Sicherheitskreises im Controller (Pin 40) 0 --> Disable 1 --> Enable (Überbrücken des Sicherheitskreises)
Datenbit 6	nicht belegt
Datenbit 7	nicht belegt

Eingabeport U6: (Basisadresse + 0Ch)

Bit am Datenbus	Pin am RIBBON-Stecker	Bezeichnung
Datenbit 0	21	INPUT 9 (Referenzschalter)
Datenbit 1	45	INPUT 10 (Referenzschalter)
Datenbit 2	20	INPUT 11 (Referenzschalter)
Datenbit 3	44	INPUT 12 (Referenzschalter)
Datenbit 4	19	INPUT 13 (Error Endstufe)
Datenbit 5		Verbunden mit Watch-Dog
Datenbit 6		nicht belegt, beim Rücklesen 1
Datenbit 7		nicht belegt, beim Rücklesen 1

Eingabeport U9: (Basisadresse + 0Eh)

Bit am Datenbus	Pin am RIBBON-Stecker	Bezeichnung
Datenbit 0	25	INPUT 1 (Endschalter)
Datenbit 1	49	INPUT 2 (Endschalter)
Datenbit 2	24	INPUT 3 (Endschalter)
Datenbit 3	48	INPUT 4 (Endschalter)
Datenbit 4	23	INPUT 5 (Endschalter)
Datenbit 5	47	INPUT 6 (Endschalter)
Datenbit 6	22	INPUT 7 (Endschalter)
Datenbit 7	46	INPUT 8 (Endschalter)

Die Eingabekanäle INPUT 1 bis INPUT 13 werden intern vom Treiber zur Abfrage der Referenz- und Endschalter benutzt.

2.7 Beschaltung der Encoder

In diesem Kapitel wollen wir die Beschaltungsmöglichkeiten unserer Steuerkarte für die verschiedenen Encoder beschreiben. Diese Information ist nur für denjenigen Anwender wichtig, der unsere PC-Einsteckkarte als Einzelstück erwerben und in Eigenentwicklung einsetzen will.

Unsere PC-Einsteckkarte arbeitet nur mit Inkrementalgebern. Ein Betrieb mit Absolutgebern ist nicht möglich. Die Signale der Spuren A und B müssen einen Phasenversatz von 90° aufweisen. Aus diesen beiden Signalen wird die Drehrichtung des Motors abgeleitet und die Auflösung des Encoders vervierfacht. Das Referenzsignal Z der Dreh-Inkremmentalgeber, das eine volle Umdrehung der Motorwelle signalisiert, ist aus Platzgründen nicht zurückgeführt.

Um die Encoder mit Strom zu besorgen, können Sie über unsere PC-Einsteckkarte die 5 V- oder 12 V-Spannung des PC's benutzen. Hier haben Sie die Möglichkeit, mit dem Jumper J700 die Versorgungsspannung für die

Encoder zu definieren (siehe Bild 2.1). Die entsprechenden Stellungen dafür sind:

Stellung 1-2 5 V-Versorgungsspannung der externen Encoder
Stellung 3-2 12 V-Versorgungsspannung der externen Encoder.

Bei einer externen Spannungsversorgung der Encoder müssen Sie Ihr GND-Signal auf die entsprechende Klemme am RIBBON-Stecker legen, um ein gemeinsames Bezugspotential sicherzustellen (Pin 50).

Grundsätzlich ist die Encoder-Eingangsbeschaltung auf unserer PC-Einsteckkarte anpassbar, sowohl für einen Betrieb von Encodern mit Signalen nach RS422 Standard als auch für einen Betrieb von Encodern, deren Signale einen Massenbezug haben. In jedem Fall ist der RS422 für einen störungsfreien Betrieb zu empfehlen.
Für den Fall, dass die Signale A, /A und B, /B nach der RS422 Spezifikation vom Encoder geliefert werden, muss auf der Einsteckkarte ein Leitungsabschlusswiderstand zur Unterbindung von Leitungsreflexionen (je Kanal) gesetzt werden, um einen störungsfreien Empfang zu garantieren. Der Widerstand sollte im Bereich von 220 Ohm liegen (siehe Bild 2.6). Dieser Widerstand wird zwischen die Signalleitungen A und /A sowie B und /B geschaltet. Die Bestückungspfads sind auf der PC-Einsteckkarte vorhanden.

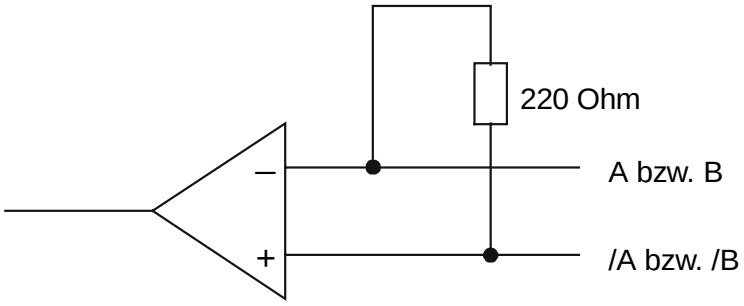


Bild 2.6: Signalanpassung für Encoder nach RS422 Spezifikation

Für den Fall, dass die Encoder die Signale A, /A und B, /B mit Massenbezug liefern, müssen die nicht benutzten Leitungssignale über internen Spannungsteiler auf LOW-Spannungspegel gelegt werden. Bei einer Versorgungsspannung von z. B. 5 V müssen die nicht benutzten Leitungssignale auf ca. 2,2 V herabgesetzt werden (siehe Bild 2.7). Sollten Ihre Encoder mit anderen Signalpegeln arbeiten, so muss deren externen Anpassung entsprechend erfolgen.

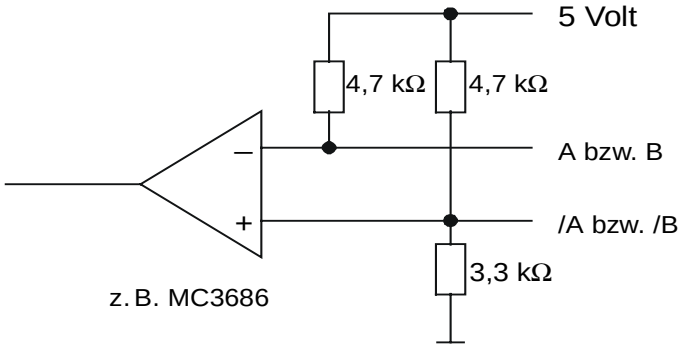
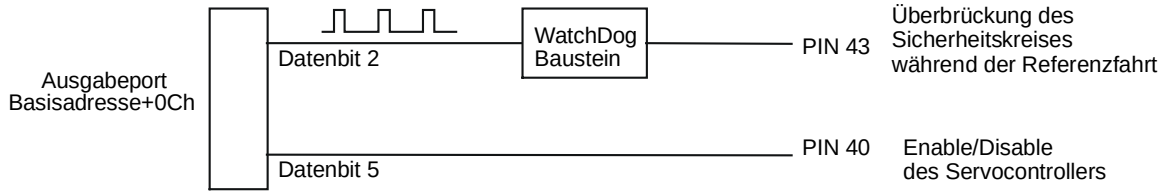


Bild 2.7: Signalanpassung für Encoder mit Massenbezug

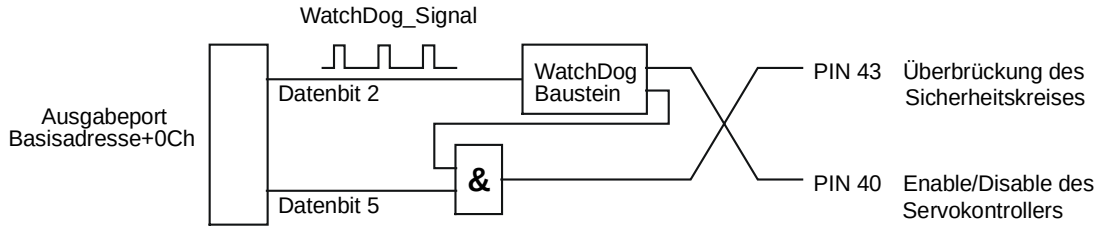
Falls Sie die Anpassung für die Encoder selbst machen müssen, sollten Sie es fachgerecht durchführen. Im Anhang A ist die Encoder-Eingangschaltung zu sehen.

2.8 Treiberversion 3.10 und Hardware-Unterschiede zu der Version 3.00

Im Vergleich zu der Treiberversion 3.00 benötigt die Version 3.10 eine kleine Hardware-Änderung auf der PC-Einsteckkarte UPMV 4/12. Diese Änderung bezieht sich ausschließlich auf die beiden Ausgabeports an den Pin 40 und Pin 43 des RIBBON-Steckers. Das Bild 2.8 zeigt Ihnen den Unterschied.



Hardware für die Treiberversion 3.00



Hardware für die Treiberversion 3.10

Bild 2.8: Hardwareunterschiede zwischen den Treiberversionen 3.00 und 3.10

In der Version 3.00 wird der *isel*-Servocontroller durch das Signal an Pin 40 freigegeben oder gesperrt. Das Watch-Dog-Signal an Pin 43 überbrückt des Sicherheitskreises automatisch während der Referenzfahrt.

Bei der Version 3.10 wird der Servocontroller durch das Watch-Dog-Signal an Pin 40 freigegeben. Falls die Treibersoftware abgestürzt ist, wird kein Watch-Dog-Signal ausgelöst. Der Servocontroller wird sofort abgeschaltet.
Das Überbrücken des Sicherheitskreises für die Referenzfahrt und für das Herausfahren aus einem aktiven Hardware-Endschalter wird durch das Setzen des Datenbits 5 an Pin 43 realisiert. Die Änderung bei der Version 3.10 ist notwendig wegen der ständigen Überwachung der Software durch das Watch-Dog-Signal.
Die Hardwareänderung wird nur durch den Umtausch des Bausteins U14 realisiert (siehe Bild 2.1). Es ist ein PLD. Falls Sie von der Treiberversion 3.00 auf der Version 3.10 umsteigen wollen, können Sie jederzeit von uns den neuen PLD bekommen.
Bei der Bestellung sollten Sie auf den Namen des alten und des neuen PLD achten:

Treiberversion	PLD-Namen
Version 3.00	SE_LIW2N
Version 3.10	SE_LIW3N

Beim Stecken des neuen PLD's müssen Sie auf die Pin-Nummer des PLD's achten. Die Halbkreis-Markierung auf der Platine und die Halbkreis-Markierung auf dem Gehäuse des PLD's müssen in die gleiche Richtung zeigen.
Die Software und die Hardware der beiden Versionen 3.00 und 3.10 sind zu einander nicht kompatibel.

2.9 Die Belegung des 50-poligen RIBBON-Steckverbinders

Bei dem eingesetzten Steckverbinder handelt es sich um einen 50-poligen RIBBON Steckverbinder mit folgender Pin-Belegung:

Pin	Richtung	Bedeutung
1	IN	Index-Spur Encoder Servo 4
2	IN	Negat Index-Spur Encoder Servo 4
3	IN	Negat Spur A Encoder Servo 4
4	IN	Negat Spur B Encoder Servo 4
5	IN	Negat Index-Spur Encoder Servo 3
6	IN	Negat Spur A Encoder Servo 3
7	IN	Negat Spur B Encoder Servo 3
8	IN	Negat Index-Spur Encoder Servo 2
9	IN	Negat Spur A Encoder Servo 2
10	OUT	± 10 V Sollwert Servo 4
11	IN	Negat Spur B Encoder Servo 2
12	IN	Negat Index-Spur Encoder Servo 1
13	IN	Negat Spur A Encoder Servo 1
14	IN	Negat Spur B Encoder Servo 1
15	OUT	± 10 V Sollwert Servo 2

16	OUT	Emitter Opto-Koppler OUT 1
		Enable/Disable der Leistungsendstufen
17	OUT	Analog-GND
18	IN	GND externe I/O-Spannungsversorgung
		GND-Masse I/O-Versorgung
19	IN	INPUT 13 Anode Opto-Koppler
20	IN	INPUT 11 Anode Opto-Koppler
21	IN	INPUT 9 Anode Opto-Koppler
22	IN	INPUT 7 Anode Opto-Koppler
23	IN	INPUT 5 Anode Opto-Koppler
24	IN	INPUT 3 Anode Opto-Koppler
25	IN	INPUT 1 Anode Opto-Koppler
26	OUT	+ Versorgung Encoder
27	IN	Spur A Encoder Servo 4
28	IN	Spur B Encoder Servo 4
29	IN	Index-Spur Encoder Servo 3
30	IN	Spur A Encoder Servo 3
31	IN	Spur B Encoder Servo 3
32	IN	Index-Spur Encoder Servo 2
33	IN	Spur A Encoder Servo 2
34	IN	Spur B Encoder Servo 2
35	OUT	± 10 V Sollwert Servo 3
36	IN	Index-Spur Encoder Servo 1
37	IN	Spur A Encoder Servo 1
38	IN	Spur B Encoder Servo 1
39	OUT	± 10 V Sollwert Servo 1
40	OUT	Emitter Opto-Koppler OUT
		Watch-Dog-Signal
41	IN	Spannung der externen I/O-Versorgung
		18 V < I/O-Versorgung < 26 V
42	OUT	Analog-GND
43	OUT	Emitter Opto-Koppler OUT
		Freigabe des Sicherheitskreises des Controllers
44	IN	INPUT 12 Anode Opto-Koppler
45	IN	INPUT 10 Anode Opto-Koppler
46	IN	INPUT 8 Anode Opto-Koppler
47	IN	INPUT 6 Anode Opto-Koppler
48	IN	INPUT 4 Anode Opto-Koppler
49	IN	INPUT 2 Anode Opto-Koppler
50	IN	GND Encoder-Versorgung

Software-Beschreibung

3 Der Softwaretreiber für die Steuerung von Servomotoren

3.1 Installation des Treibers

Das Herzstück des Steuerungssystems ist der Treiber ISELDRV.EXE (*isel*-drive). Er bleibt nach dem Laden resident im Hauptspeicher**,** belegt ungefähr 110 KByte und übernimmt ab diesem Zeitpunkt für Sie als Anwender solche Arbeiten wie z. B. die Interpolation und die Koordinierung der Achsbewegungen, die Verwaltung des Systems, die Kommunikation mit der Hardware, etc. Abgesehen davon, dass es im Hauptspeicher des PC weniger Speicherplatz gibt, merken Sie als PC-Anwender absolut keinen Unterschied zwischen vor und nach dem Laden des Treibers, d. h. Sie können weiterhin normal mit dem PC arbeiten. Es ist eigentlich egal, ob Sie jetzt mit anderen Programmen arbeiten oder ob Sie jetzt Ihre eigenen Programme starten, um die Dienste des Treibers in Anspruch nehmen zu können.
Im Fall, dass es nicht mehr genügend Speicher für den Start Ihrer eigenen Programme gibt, ist es empfehlenswert, Speicheroptimierungsprogramme wie z. B. MEMMAKER.EXE von MICROSOFT zu benutzen oder gegebenenfalls andere residente Treiber zu entfernen.
Das Laden des Treibers bzw. das Entfernen des bereits installierten Treibers aus dem PC-Speicher ist sehr einfach, weil Sie nichts anderes als die DOS-Konventionen anwenden müssen. Sie starten den Treiber mit der Kommandozeile:

[Laufwerk:\] [Pfad] ISELDRV.EXE *.INI

oder mit der Kommandozeile:

[Laufwerk:\] [Pfad] ISELDRV.EXE *.INI < _ENTER_ .

*.INI steht für die Initialisierungsdatei, die Sie mit unserem Konfigurationsprogramm PARKON.EXE erzeugt haben (siehe das entsprechende Handbuch). Beachten Sie, dass Treiber und Initialisierungsdatei im selben Verzeichnis stehen.
Beim Start ruft der Treiber das Initialisierungsprogramm ISELINI.EXE auf. Dieses Initialisierungsprogramm muss ebenfalls im diesem Verzeichnis stehen. Es hat die Aufgabe, alle notwendigen Parameter und Standardwerte aus der angegebenen Initialisierungsdatei zu lesen und den Treiber mit den gelesenen Werten zu initialisieren.

Wenn Sie den Treiber mit dem Umleitungszeichen < und mit der von uns mitgelieferten Datei _ENTER_ starten, brauchen Sie nachher die Taste ENTER nicht zu betätigen. In diesem Fall läuft die Installation des Treibers automatisch ohne Ihr Tun.

Anhand der auf dem Monitor erscheinenden Meldung können Sie dann feststellen, ob der Treiber erfolgreich geladen bzw. entfernt wurde oder nicht. Eine der folgenden Meldungen ist auf dem Monitor zu erwarten.

Falsches Format der Initialisierungsdatei
Die Initialisierungsdatei ist schreibgeschützt und hat intern ein eigenes Format. Diese Meldung deutet darauf hin, dass das Format der Datei aus irgendeinem Grunde verändert wurde. Der Treiber kann die Parameter nicht mehr ordnungsgemäß aus der Datei einlesen und damit auch nicht installiert bzw. entfernt werden. Es bleibt Ihnen nicht anderes übrig, als die Initialisierungsdatei mit Hilfe des Konfigurationsprogramms neu zu erzeugen.

Falscher Name der Initialisierungsdatei
Die von Ihnen angegebene Initialisierungsdatei kann in dem Verzeichnis, in dem der Treiber steht, nicht gefunden werden. Damit kann der Treiber nicht installiert bzw. entfernt werden.

Fehlende Initialisierungsdatei
Sie haben vergessen, den Namen der Initialisierungsdatei anzugeben. Der Treiber kann somit nicht installiert bzw. entfernt werden.

Zuviel Parameter
Auf der Kommandozeile dürfen Sie außer dem Namen des Treibers nur noch den Namen der Initialisierungsdatei angeben. Hier haben Sie offensichtlich auf der Kommandoebene etwas mehr angegeben. Der Treiber kann diese Kommandozeile nicht mehr richtig interpretieren. Damit kann der Treiber auch nicht installiert bzw. entfernt werden.

Initialisierungsprogramm nicht vorhanden
Das Initialisierungsprogramm ISELINI.EXE existiert nicht in dem Verzeichnis, in dem der Treiber und die Initialisierungsdatei stehen. Der Treiber kann somit nicht installiert bzw. entfernt werden. Sie müssen das Initialisierungsprogramm in dieses Verzeichnis kopieren, um den Treiber starten zu können.

Initialisierungsprogramm nicht ausführbar
Diese Fehlermeldung deutet in den meisten Fällen darauf hin, dass Ihr Rechner zu wenig freien Speicher hat. Deswegen kann der Treiber das Initialisierungsprogramm ISELINI.EXE nicht mehr aufrufen. In diesem Fall sollen Sie einige residente Programme in Ihrem Rechner entfernen, um Platz zu schaffen.

Der Zugriff auf den Datenträger ist nicht möglich
Beim Start ruft der Treiber das Initialisierungsprogramm ISELINI.EXE, um die Anlagenparameter in der Initialisierungsdatei zu lesen. Mit Hilfe des Datenträgers (Diskette oder Festplatte), auf dem der Treiber steht, werden die eingelesenen Werte dem Treiber übergeben. Falls dieser Datenträger nicht lesbar oder beschreibbar ist, bekommen Sie diesen Fehler zu sehen.

Der Datenträger ist voll
Beim Start ruft der Treiber das Initialisierungsprogramm ISELINI.EXE, um die Anlagenparameter in der Initialisierungsdatei zu lesen. Mit Hilfe des Datenträgers (Diskette oder Festplatte), auf dem der Treiber steht, werden die eingelesenen Werte dem Treiber übergeben. Falls dieser Datenträger voll ist, kommt diese Fehlermeldung auf dem Monitor.

Rechner ist zu langsam.
Treiber wurde nicht installiert.
Beim Erstellen der Initialisierungsdatei mit dem Initialisierungsprogramm PARKON.EXE können Sie definieren, wie viel Prozent der Rechenleistung Ihres Rechners dem Treiber zur Verfügung stehen soll. Beim Starten kontrolliert der Treiber, ob die ihm zugeteilte Rechenleistung ausreicht, um seine Aufgaben zu bewältigen. Falls es nicht der Fall ist, wird diese Fehlermeldung ausgegeben, der Treiber wurde nicht installiert. In diesem Fall müssen Sie den dem Treiber zur Verfügung stehenden Anteil der Rechenleistung erhöhen. Der dem Treiber zugehörige Anteil der Rechenleistung soll den Grenzwert 75 % nicht überschreiten, weil Ihr Anwenderprogramm auch eine bestimmte Rechenzeit braucht. Falls es trotzdem nicht geht, müssen Sie einen Co-Prozessor in ihren Rechner einbauen. Das Fehlen eines Co-Prozessors ist in den meisten Fällen der Grund für diese Fehlermeldung. Bei unseren Anlagen haben wir meistens den Wert 50 % benutzt.

Warnung: Rechner ist langsam.
Installation des Treibers erfolgt mit der Taste „J“.
Diese Fehlermeldung stellt einen Grenzfall dar. Falls Sie trotzdem den Treiber mit der Taste 'j' oder mit der Taste 'J' installieren, müssen Sie damit rechnen, dass der Treiber aufgrund der fehlenden Rechenleistung die Bewegung Ihrer Anlage nicht mehr gleichförmig ausführen kann. Jede andere Taste bewirkt, dass der Treiber nicht installiert wird. Mit Hilfe des Konfigurationsprogrammes PARKON.EXE können Sie den dem Treiber zur Verfügung stehenden Anteil der Rechenleistung erhöhen, um diese Warnung zu vermeiden. Der dem Treiber zugehörige Anteil der Rechenleistung soll den Grenzwert 75 % nicht überschreiten, weil Ihr Anwenderprogramm auch eine bestimmte Rechenzeit braucht. Falls es trotzdem nicht geht, müssen Sie einen Co-Prozessor in ihren Rechner einbauen. Das Fehlen eines Co-Prozessors ist in den meisten Fällen der Grund für diese Fehlermeldung.

Zu viel residente Programme im Speicher
Im Speicher vom PC stehen schon zu viele residente Programme. Der Treiber konnte nicht installiert werden. Um die Installation zu ermöglichen, müssen Sie einige residente Programme entfernen. Damit hat der PC wieder Platz für den Treiber.

Keine bzw. defekte PC-Einsteckkarte oder falsche Konfiguration
Sie haben entweder vergessen, die mitgelieferte Karte einzustecken oder eine falsche Basisadresse der Karte in die Initialisierungsdatei eingegeben (siehe Handbuch für das Konfigurationsprogramm PARKON.EXE). Es kann auch sein, dass die PC-Einsteckkarte defekt ist. Der Treiber kann natürlich nicht installiert werden.

Unbekannter Fehler
Die Fehlerquelle kann vom Treiber nicht lokalisiert werden. Im Fall dieser Fehlermeldung wird der Treiber nicht installiert.

Einschalten der Endstufe (Weiter mit ENTER)
Endlich haben Sie es geschafft, den Treiber zu installieren. Hier erinnert der Treiber Sie an das Einschalten der Endstufe. Nachdem die Taste ENTER bestätigt wurde, erscheint der DOS-Prompt wieder auf dem Monitor. Der Treiber ist ordnungsgemäß installiert. Der PC gehört wieder Ihnen. Ab diesem Zeitpunkt können Sie mit dem Treiber über die von Ihnen selbst definierte Interrupt-Schnittstelle kommunizieren. Falls Sie den Treiber mit der Option [< _ENTER_] starten, brauchen Sie die Taste ENTER nicht zu bestätigen. Das Bestätigen mit der Taste ENTER wird durch das Umleitungszeichen < und der Datei _ENTER_ nachgebildet. Trotzdem bekommen Sie diese Zeile zu sehen

Treiber ist schon installiert.
Entfernen mit der Taste E
Der Treiber ist schon installiert. Wenn Sie den Treiber aus dem Speicher entfernen wollen, müssen Sie entweder die Taste 'e' oder die Taste 'E' betätigen. Wenn Sie eine dieser Tasten betätigen, versucht der Treiber, sich aus dem Speicher selbst zu entfernen.
Wenn es ihm gelingt, erscheint die Nachricht: **Treiber wurde entfernt** auf dem Monitor. Wenn es mit dem Entfernen nicht klappen sollte, bekommen Sie entweder die Meldung: **Falsche Initialisierungsdatei. Treiber wurde nicht entfernt.**
oder die Meldung: **Treiber kann nicht mehr entfernt werden**.
Im ersten Fall haben Sie offensichtlich die falsche Initialisierungsdatei benutzt. Im zweiten Fall bleibt das Neustarten des PC der einzige Ausweg, um den Treiber loszuwerden.
Wenn Sie eine andere Taste als die beiden Tasten 'e' und 'E' betätigen, erscheint die Meldung: **Treiber bleibt resident im Speicher** auf dem Monitor. Der Treiber bleibt dann weiterhin resident im Speicher.

3.2 Einige Vorerläuterungen zum Softwaretreiber

3.2.1 Das Bewegungssegment

Ein Bewegungssegment ist eine Kurve, auf der sich das Werkzeug von seinem momentanen Anfangspunkt zu seinem Endpunkt bewegt. Falls es sich um eine vorher definierte Kurve handelt, haben wir dann eine Interpolation der Bewegung. In diesem Fall sind die meist benutzten Bewegungssegmente (kurz gesagt: Segmente) die Linear- und die Kreissegmente. Entsprechend haben wir dann eine Linearinterpolation und eine Kreisinterpolation. Die Kreisinterpolation ist ein Sonderfall der Helixinterpolation (siehe Kapitel 3.3.30). Weil die Helixinterpolation nicht so sehr gebräuchlich ist, sprechen wir nur von der Kreisinterpolation, obwohl unsere Steuerung sowohl die Kreis- als auch die Helixinterpolation anbietet. Außerdem haben wir noch ein sogenanntes PTP-Segment. Ein PTP-Segment liegt vor, falls Sie eine PTP-Steuerung haben. Die Kurve, auf der sich das Werkzeug bewegt, ist eine undefinierte Kurve. Hier müssen wir aber ergänzen, dass der Begriff "PTP-Segment" ein von uns definierter Begriff ist. Jede Segmentart wird noch einmal in Absolut- und Relativsegmente geteilt (siehe Bild 3.1).

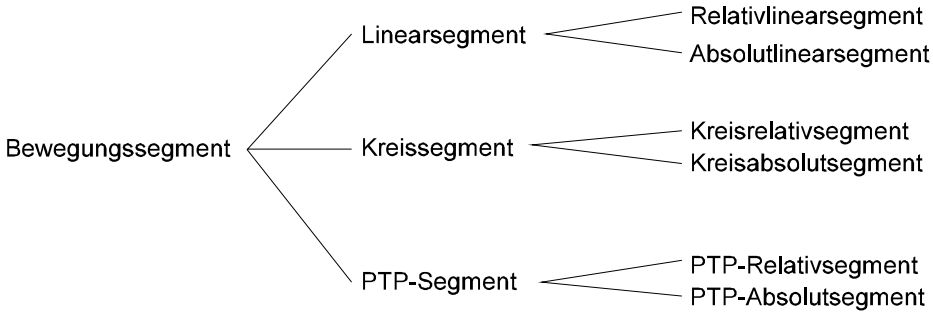
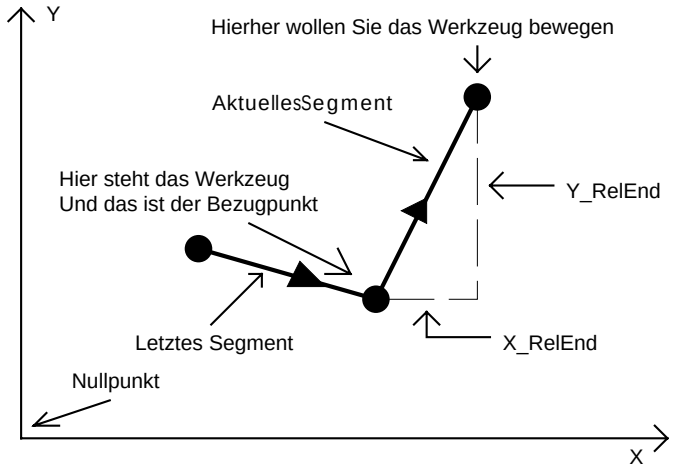


Bild 3.1: Verschiedene Arten der häufig benutzten Bewegungssegmente

Im Folgenden wollen wir Ihnen die Unterschiede zwischen einem Relativ- und einem Absolutsegment erklären.
Um ein Bewegungssegment eindeutig zu bestimmen, muss natürlich zuerst bekannt sein, welche Segmentart (ein Linear- ein Kreissegment oder ein PTP-Segment) vorliegt. Der Anfangspunkt eines Segments ist immer bekannt. Das ist der Punkt, in dem das Werkzeug steht, direkt bevor das Segment abgefahren wird. Während der Bearbeitungsphase, wenn der sich das Werkzeug auf verschiedenen aufeinander folgenden Segmenten bewegt, ist der Endpunkt eines Segments gleichzeitig der Anfangspunkt des nächsten Segments. Bei einem Linearsegment oder einem PTP-Segment müssen Sie noch die Koordinaten des Endpunkts angeben. Damit ist ein Linearsegment bzw. ein PTP-Segment eindeutig bestimmt. Da die notwendigen Angaben für ein Linearsegment sowie für ein PTP-Segment identisch sind, werden wir im Weiteren nur noch vom Linearsegment sprechen. Es hängt dann nur noch von

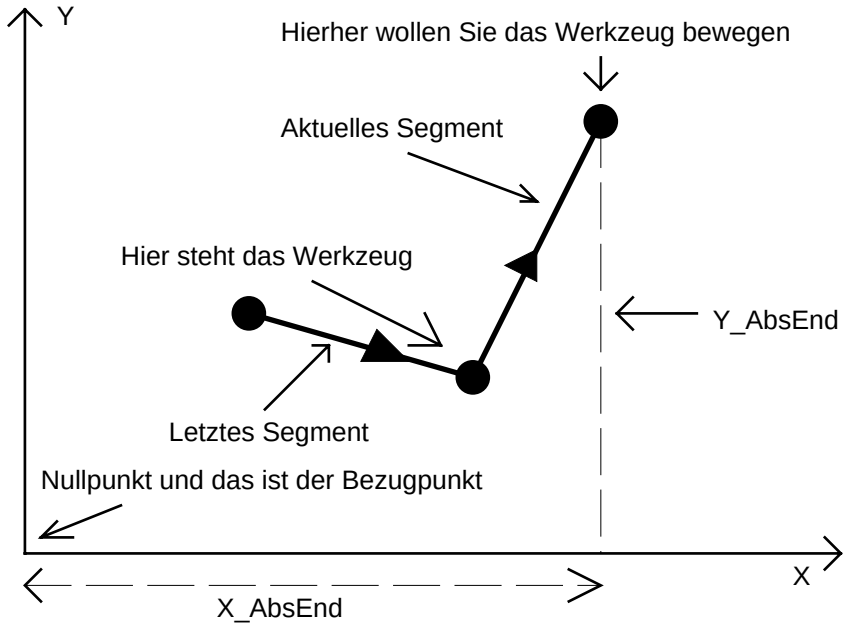
der Anlagenstruktur ab, ob dann ein Linearsegment oder ein PTP-Segment entsteht.
Um ein Kreissegment eindeutig zu bestimmen, müssen Sie außer den Koordinaten des Endpunktes noch die Koordinaten des Kreismittelpunktes sowie die Bewegungsrichtung auf dem Kreis (im Uhrzeigersinn oder gegen Uhrzeigersinn) angeben.



X-RelEnd und Y-RelEnd sind die Koordinaten des Segmentendpunktes, die relativ zum Segmentanfangspunkt sind.

Bild 3.2: Berechnung der Relativkoordinaten am Beispiel eines Linearrelativsegments in der XY-Ebene

Das Wort "Relativ" im Begriff "Relativsegment" hat mit dem Bezugspunkt für die Berechnung der Koordinaten etwas zu tun. Der Bezugspunkt für diese Koordinaten ist der Punkt, in dem das Werkzeug direkt vor dem Abfahren des Segments steht. Während der Bearbeitungsphase ist der Bezugspunkt für jedes neue Segment im Prinzip sein eigener Anfangspunkt und gleichzeitig der Endpunkt des vorigen Segments. Das Bild 3.2 hilft Ihnen weiter, wie dies zu verstehen ist. Dieses Bild ist nur ein Beispiel, wie man die Koordinaten eines Linearrelativsegments in der XY-Ebene berechnen kann. Auf die gleiche Weise können Sie auch die Koordinaten für verschiedene Achsen bestimmen. Bei einem Kreisrelativsegment dient sein Anfangspunkt bzw. der Endpunkt des letzten Segments als Bezugspunkt nicht nur für die Berechnung der Koordinaten des Segmentendpunktes, sondern auch für die Berechnung der Koordinaten des Kreismittelpunktes.
Sie sehen, dass sich der Bezugspunkt für die Berechnung von Koordinaten der Relativsegmente ständig ändert. Jedes Relativsegment hat seinen eigenen Bezugspunkt und das ist nichts anderes als sein eigener Anfangspunkt.
Bei Absolutsegmenten ist das nicht mehr der Fall. Hier haben alle Segmente immer den selben Bezugspunkt für die Berechnung der Koordinaten. Dies ist der Koordinatenursprung (Nullpunkt).
Das Bild 3.3 soll verdeutlichen, was hier gemeint ist.



X-AbsEnd und Y-AbsEnd sind die Absolutkoordinaten des Segmentendpunktes. Der Bezugspunkt ist der Koordinatenursprung bzw. der Nullpunkt.

Bild 3.3: Berechnung der Absolutkoordinaten des Endpunktes am Beispiel eines Linearabsolutsegments in der XY-Ebene.

Im Bild 3.3 ist ein Beispiel für die Berechnung der Koordinaten eines Linearabsolutsegments in der XY-Ebene aufgezeigt. Auf ähnliche Weise können Sie auch die Koordinaten für andere Achsen bzw. für ein Kreisabsolutsegment bestimmen. Hier müssen wir aber ergänzen, dass man den Koordinatenursprung bzw. Nullpunkt jederzeit neu definieren kann. Dann gilt der neue Nullpunkt solange als Bezugspunkt für alle Absolutsegmente, bis wieder ein neuer Nullpunkt definiert wird.

3.2.2 Wie wird ein Bewegungssegment realisiert?

Aufgrund der Trägheit sowie des begrenzten Beschleunigungsvermögens eines Antriebsystems muss die Interpolation ein geeignetes Geschwindigkeitsprofil für das Werkzeug generieren, damit das Werkzeug nicht über den Segmentendpunkt hinausfährt (siehe Bild 3.4).

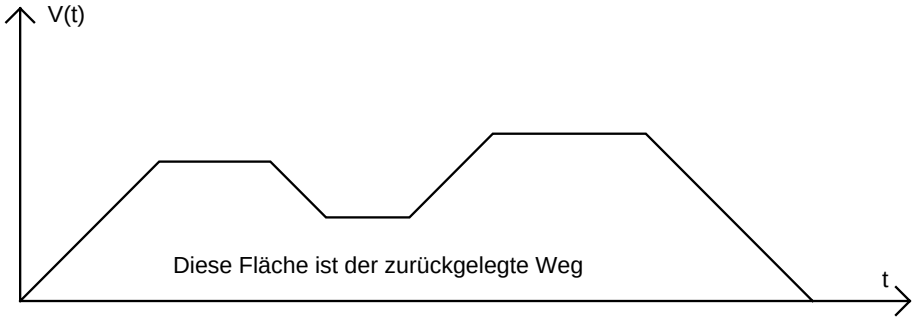


Bild 3.4: Geschwindigkeitsprofil für das Fahren eines Segments

Im Bild 3.4 können Sie das von unserer Interpolation generierte Profil der Werkzeuggeschwindigkeit sehen. Es ist egal, ob es sich hier um ein Linear- oder ein Kreissegment handelt. Der Unterschied zwischen einem Linear- und einem Kreissegment besteht viel mehr im Profil der Achsgeschwindigkeiten. Bei einem Linearsegment haben die Achsgeschwindigkeiten ähnliche Profile, wie es im Bild 3.4 gezeigt ist. Bei einem Kreissegment haben die Achsgeschwindigkeiten Profile, die einer Sinus- bzw. Cosinus-Funktion ähnlich sind. Aber über die Profile der Achsgeschwindigkeiten brauchen Sie sich keine Gedanken zu machen.

Die Integration des Geschwindigkeitsprofils über die Zeit ist der zurückgelegte Weg. Mit unserer Interpolation ist es möglich, die Geschwindigkeiten während des Fahrens eines Segments im Bereich zwischen 0 % ... 140 % zu verändern. Der Fachbegriff dafür heißt "Override".

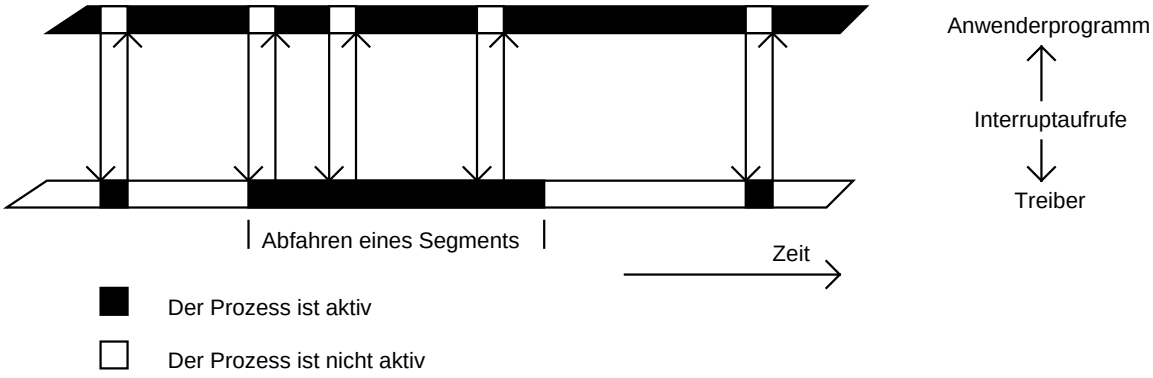


Bild 3.5: Paralleles Arbeiten zwischen einem Anwenderprogramm und dem Treiber

Wir wollen nochmals betonen, dass die Generierung der Geschwindigkeitsprofile eine Aufgabe der Interpolation unseres Treibers ist. Sie als Anwender brauchen dem Treiber nur die Segmentparameter durch entsprechende Funktionsaufrufe mitzuteilen. Danach realisiert der Treiber das Bewegungssegment selbständig.

In Abhängigkeit von der gewünschten Werkzeuggeschwindigkeit und auch von dem zu fahrenden Segment kann das Abfahren eines Bewegungssegments sehr lange dauern. Es wäre witzlos, wenn der Treiber in dieser ganzen Zeit die Kontrolle über den PC hat. Wenn das der Fall wäre, hätten Sie keine Möglichkeiten mehr, den Bewegungsablauf zu beeinflussen, d. h. Sie könnten z. B. die Geschwindigkeit nicht ändern oder die Bewegung nicht anhalten. Aus diesem Grund haben wir den Treiber so konzipiert, dass der Treiber das Abfahren eines Bewegungssegments im Hintergrund realisiert.

Das Bild 3.5 soll Ihnen die Parallelität zwischen einem Anwenderprogramm und dem Treiber verdeutlichen. Dabei spielt der Treiber eine passive Rolle. Bei den meisten Interrupt-Aufrufen wird der Treiber sehr kurz aktiviert, weil die zu lösenden Aufgaben sehr schnell erledigt werden können. Nur beim Abfahren eines Bewegungssegments kann die Ausführungszeit länger sein. In diesem Fall hat der Interrupt-Aufruf nur die Aufgabe, dem Treiber die Segmentparameter mitzuteilen. Dann hat das Anwenderprogramm wieder die Kontrolle über den PC. Ab diesem Zeitpunkt arbeitet der Treiber selbständig im Hintergrund.

Die Interpolation, die Ausgabe der Soll-Werte an die Achsregler usw. werden automatisch durchgeführt. Wenn das Bewegungssegment irgendwann zu Ende ist, deaktiviert sich der Treiber selbst. Das Anwenderprogramm braucht sich darum nicht zu kümmern. Statt dessen kann das Anwenderprogramm durch die noch zugelassenen Interrupt-Aufrufe die Kommunikation zwischen dem Benutzer und dem Treiber realisieren, wie z. B. Abfragen und Anzeigen der Ist-Positionen sowie der Ist-Geschwindigkeit oder Veränderungen der Bearbeitungsgeschwindigkeit oder Unterbrechen des Bearbeitungsvorgangs usw. Natürlich kann das Anwenderprogramm in dieser Zeit auch die Daten für das nächsten Segment vorbereiten, wie z. B. das Lesen der Daten aus einem Datenträger und das Konvertieren der Daten in das entsprechende Format.

Das Anwenderprogramm und der Treiber teilen sich die Rechenzeit des PC untereinander. Deswegen sollten Sie die rechenintensiven Funktionsaufrufe, wie das Abfragen der Ist-Positionen (siehe Kapitel 3.3.18) oder das Abfragen der Ist-Geschwindigkeit (siehe Kapitel 3.3.19) nicht in kurzen Zeitabständen hintereinander ausführen. Sonst bekommt der Treiber nicht mehr die für ihn notwendige Rechenzeit. Es kann dann zu einer ungleichmäßigen Bewegung führen. Der Zeitabstand zwischen zwei Aufrufen sollte schon etwa 50 ... 100 ms betragen. Eine Zeitverzögerung können Sie mit der Funktion 7 problemlos realisieren (siehe Kapitel 3.3.7).

3.2.3 Wie kann man eine Kontur über das *isel*-Zwischenformat erzeugen

Mit einer CNC-Anlage will man normalerweise Werkstücke mit vordefinierten Konturen herstellen. Bei einer einfachen Kontur wie z. B. Rechteck oder Kreis ist es im Prinzip kein Problem, diese Kontur in einzelne Bewegungssegmente

zu zerlegen und über entsprechenden Treiberfunktionen "zu Fuß" zu programmieren.
Bei einer komplexen Kontur ist dieser Weg nicht mehr möglich bzw. nicht mehr sinnvoll. Hier wird normalerweise ein CAD- oder ein Zeichenprogramm benutzt, um die gewünschten Konturen zu erzeugen. Die erzeugten Konturen werden automatisch von dem benutzten Programm in entsprechende Bewegungssegmente zerlegt und dann in einer Geometriedatei abgespeichert.

Wenn Sie einen Interpreter besitzen, der unter anderem diese Geometriedateien lesen, interpretieren und in entsprechende Treiberfunktionen umsetzen kann, steht Ihnen nichts im Wege, eine komplexe Kontur mit Ihrer Anlage zu erzeugen. Der Nachteil dieser Methode liegt in der Vielfalt der auf dem Markt vorhandenen CAD- und Zeichenprogramme, die die Geometriedaten in unterschiedlichen Formaten in den entsprechenden Geometriedateien abspeichern. Mit der Zeit haben sich solche Standards wie z. B. HP/GL-, ADI-, NCI-Dateiformat etc. ergeben. Weil es diese unterschiedlichen Formate gibt, brauchen Sie logischerweise auch unterschiedliche Interpreter. Dadurch entsteht ein Mehraufwand, wenn Sie mit Ihrer CNC-Anlage verschiedene Dateiformate bearbeiten wollen (siehe Bild 3.6).

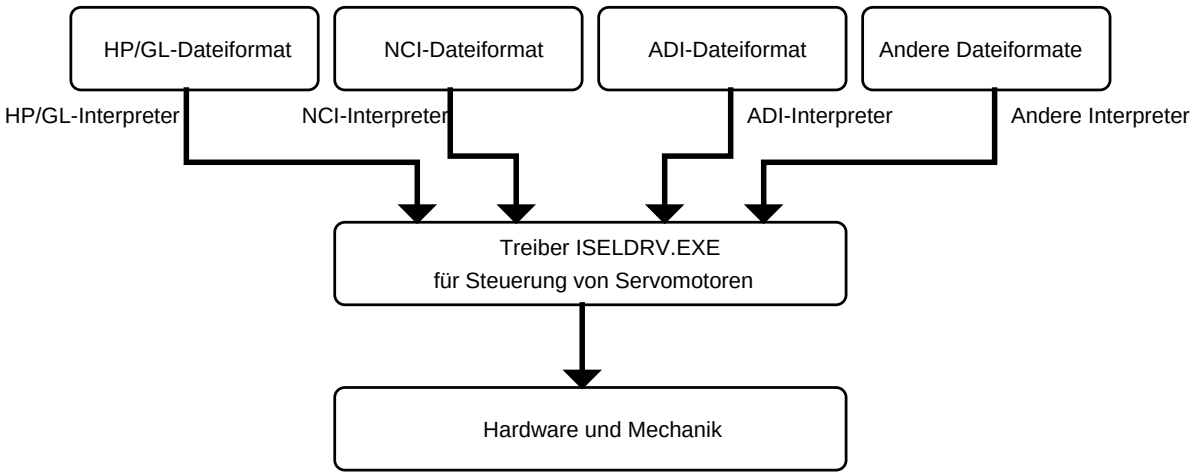


Bild 3.6: Anlagensteuerung mit unterschiedlichen Interpretern für verschiedene Dateiformate

Um den Mehraufwand durch verschiedene Interpreter zu vermeiden, haben wir ein so genanntes *isel*-Zwischenformat definiert. Die verschiedenen Dateiformate werden zuerst durch entsprechende Umsetzer in das *isel*-Zwischenformat konvertiert. Um Ihre Servoanlage über unseren Treiber zu steuern, brauchen Sie schließlich nur noch einen Interpreter, der das *isel*-Zwischenformat liest, interpretiert und dann entsprechende Treiberfunktionen aufruft (siehe Bild 3.7).

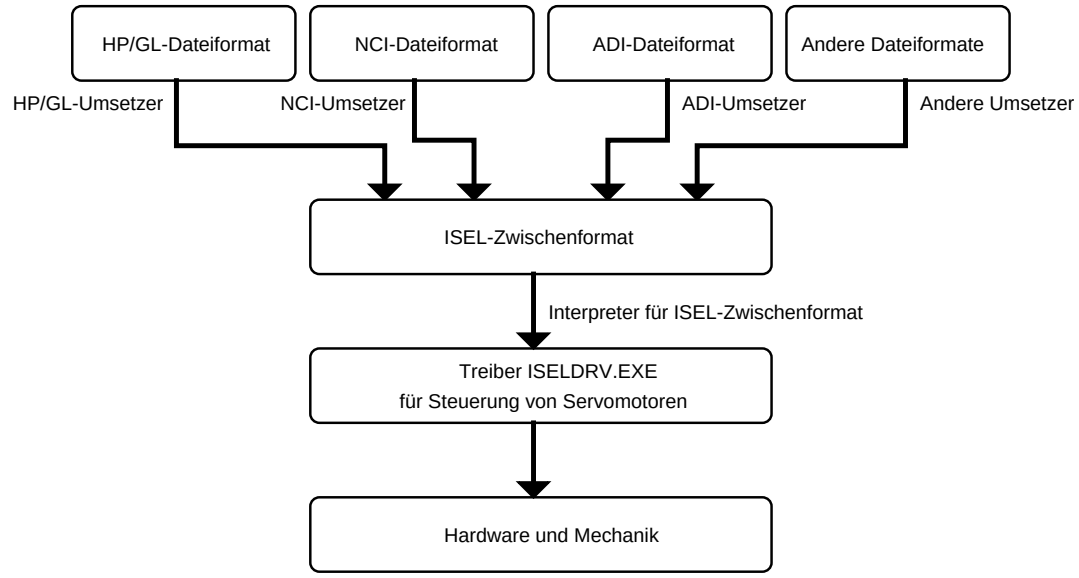


Bild 3.7: Anlagensteuerung mit unterschiedlichen Umsetzern und einem Interpreter

Aus dem ersten Blick scheint die Anlagensteuerung mit dem *isel*-Zwischenformat komplizierter. Aber Sie sollten nicht vergessen, dass der Aufwand für einen Interpreter mit einer entsprechenden Bedienoberfläche wesentlich höher ist, als der für einen einzigen Umsetzer. Ein anderer Vorteil liegt in der Bahnbearbeitung, was Gegenstand des nächsten Kapitels ist.

Wenn Sie Details über das *isel*-Zwischenformat wissen wollen, nehmen Sie das entsprechende Handbuch zu Rate. Hier wollen wir unser Zwischenformat nur ganz kurz erläutern, damit Sie in der Lage sind, die im nächsten Kapitel behandelte Bahnbearbeitung zu verstehen.

Das isel-Zwischenformat ist ein ASCII-Format. Jeder Befehl liegt separat auf einer Zeile. Am Zeilenanfang steht die Satznummer. Danach kommt ein Klartext-Kenncode für den Befehl. Es folgen dann die entsprechenden Befehlsparameter, falls die Parameter notwendig sind. Der Befehlssatz ist erweiterbar und umfasst momentan verschiedene Befehlstypen wie z. B. Bewegungsbefehle, Befehle für das Wechseln der Werkzeuge, Befehle für das Handhaben der Ein-/Ausgabeports etc. Im Folgenden können Sie einige Befehle als Beispiele sehen.

```
N000064  MOVEABS  X1000  Y1000  Z1000  A0
```

Dieser Befehl beschreibt ein Absolutlinearsegment, das im Normalgang gefahren wird (siehe Treiberfunktion 25). Die Absolutkoordinaten des Segmentendpunktes werden durch die Buchstaben X, Y, Z und A gekennzeichnet.

```
N000011  FASTREL  X5000  Y1000  Z11000  A100
```

Dieser Befehl beschreibt ein Relativlinearsegment, das im Eilgang gefahren wird (siehe Treiberfunktion 26). Die Relativkoordinaten des Segmentendpunktes werden durch die Buchstaben X, Y, Z und A gekennzeichnet.

```
N000008  CWREL  I0  J5000  X0  Y10000  Z1000  A20000
```

Dieser Befehl beschreibt ein Relativkreissegment, das im Uhrzeigersinn gefahren wird (siehe Treiberfunktion 28). Die Koordinaten des Kreismittelpunkts werden durch I und J angegeben. Die Endpunktkoordinaten werden durch die Buchstaben X, Y, Z und A gekennzeichnet.

```
N000016  GETTOOL  7
```

Dieser Befehl verlangt, das Werkzeug mit der Nummer 7 zu holen.

```
N000005  COOLANT  ON
```

Dieser Befehl verlangt, die Kühlmittelpumpe einzuschalten.

```
N000062  REF XYZA
```

Dieser Befehl verlangt, die Referenzfahrt für die Achsen X, Y, Z und A auszuführen (siehe Treiberfunktion 6).

Abgesehen von der Bahnbearbeitung ist die Benutzung des *isel*-Zwischenformates nicht zwingend aber empfehlenswert. Wenn Sie unser Format benutzen, können Sie dann jederzeit auf unsere fertigen Software-Parkete wie z. B. Umsetzer für verschiedenen Formate oder den Interpreter für unser Zwischenformat zurückgreifen. Dadurch können Sie eine Menge Geld und Zeit sparen.

3.2.4 Wie wird die Bahnbearbeitung realisiert

Sie können also eine komplexe Kontur erzeugen, indem Sie die Anlage die einzelnen Segmente der Kontur nacheinander abarbeiten lassen. Bei dieser segmentweisen Bearbeitung entstehen Zwischenstops zwischen zwei aufeinander folgenden Segmenten (siehe Bild 3.4).
Mit dem segmentweisen Abfahren können Sie die meisten Aufgaben im Handlingbereich sehr gut lösen. Für viele Bearbeitungsaufgaben reicht die segmentweise Bearbeitung völlig aus, obwohl die Zwischenstops zwischen den Segmenten in den meisten Fällen negative Einflüsse auf die Bearbeitungsqualität haben.
Bei vielen anspruchsvollen Bearbeitungsaufgaben besteht der Wunsch, eine aus mehreren Segmenten bestehende Kontur vom Anfang bis zum Ende in einem Gang ohne Zwischenstops zu erzeugen. Diese Art der Bearbeitung heißt *Bahnbearbeitung*.

Inoffiziell unterteilt man die Bahnbearbeitung wieder in 2D-Bahnbearbeitung und 3D-Bahnbearbeitung. Bei einer 2D-Bahnbearbeitung kann man das Bahnfahren in einer Ebene realisieren. Die 3D-Bahnbearbeitung entspricht dem Bahnfahren im Raum.
Eine Maschine mit der Fähigkeit einer 3D-Bahnbearbeitung ist auch in der Lage, eine 2D-Bahnbearbeitung auszuführen. Abgesehen von der Verbesserung der Bearbeitungsqualität ist auch die Bearbeitungszeit wesentlich kleiner, weil die Zwischenstops entfallen.

Um die Bahnbearbeitung realisieren zu können, muss zuerst ein sogenanntes Geschwindigkeitsprofil über die gesamte Kontur generiert werden. Beim Bahnfahren richtet sich die Werkzeuggeschwindigkeit nach diesem Geschwindigkeitsprofil. Auf der einen Seite begrenzt das Geschwindigkeitsprofil die maximale Werkzeuggeschwindigkeit für jeden Übergang zwischen zwei beliebig aufeinander folgenden Segmenten, um die Bahnabweichungen an den Übergängen in einem zulässigen Fehlerbereich zu halten.
Auf der anderen Seite zwingt das Geschwindigkeitsprofil das Werkzeug, sich so zu bewegen, dass der Endpunkt des letzten Segments der Kontur ohne Überschwingen erreicht werden kann. Das Generieren des Geschwindigkeitsprofils ist eine sehr komplizierte Sache und erfordert, dass alle Segmente der Kontur von vornherein bekannt sein müssen.

Im Prinzip gibt es zwei Wege, um das Geschwindigkeitsprofil zu generieren. Bei der ersten Methode legt man ein kleines Fenster über die zu bearbeitende Kontur. Über den Segmenten, die in dieses Fenster passen, wird ein Geschwindigkeitsprofil erzeugt. Während der Bearbeitung wird das Fenster in Richtung des Konturendes verschoben. Das Geschwindigkeitsprofil wird jedes Mal wieder aktualisiert. Die Berechnung des Geschwindigkeitsprofils geschieht in Echtzeit.

Bei der zweiten Methode wird das Geschwindigkeitsprofil komplett fertig berechnet, bevor das Bahnfahren beginnt. Bei der ersten Methode ist eine immense Rechenkapazität notwendig. Außerdem ist das berechnete Geschwindigkeitsprofil nicht optimal in Bezug auf die gesamte Kontur, weil man das Geschwindigkeitsprofil nur über dem jeweiligen Fenster optimieren kann. Der Nachteil der zweiten Methode liegt darin, dass eine Vorberechnung notwendig ist.

Aber wir sind der Meinung, dass die Vorberechnungszeit im Vergleich zu der Bearbeitungszeit vernachlässigbar ist. Abgesehen vom optimalen Geschwindigkeitsprofil über der gesamten Kontur ist das Handhaben während des Bahnfahrens hier wesentlich einfacher. Aus diesen Gründen haben wir die Bahnbearbeitung mit der zweiten Methode realisiert.

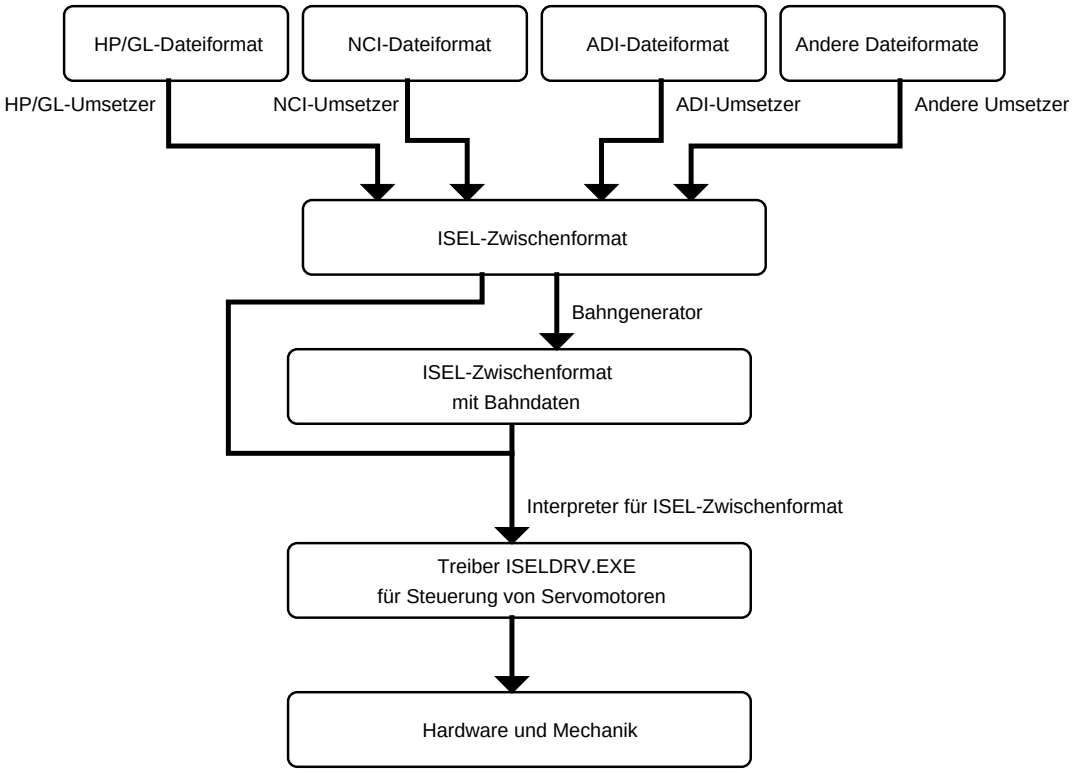


Bild 3.8: Prinzip der Maschinensteuerung mit Bahnbearbeitung

Aus diesem Bild können Sie sehen, dass die Bahnbearbeitung eine Option der Maschinensteuerung darstellt. Sie können Ihre Maschine sowohl mit einer segmentweisen Bearbeitung als auch mit einer Bahnbearbeitung steuern. Dadurch erreichen Sie eine sehr große Flexibilität.
Das Arbeitsprinzip des Bahngenerators ist ziemlich einfach. Der Bahngenerator liest und interpretiert nacheinander die Befehle aus der im *isel*-Zwischenformat gespeicherten Datendatei. Dabei werden alle aufeinander folgenden Bewegungsbefehle, die keine Eilbewegung darstellen, zu einer Kontur zusammengefasst, die mit einer Bahnbearbeitung erzeugt werden soll. Über diese Kontur wird ein Geschwindigkeitsprofil berechnet. Die Daten des Geschwindigkeitsprofils werden anstelle der Parameter aller zur Bahn gehörenden Segmente in der Datendatei gespeichert. Alle Eilbewegungsbefehle und alle Befehle, die keine Bewegungsbefehle sind, werden vom Bahngenerator als das Ende einer eventuell vorhandenen Bahn interpretiert und unverändert in die Datendatei zurückgespeichert.

Auf diese Art und Weise kann eine Datendatei mehrere Konturen enthalten, die mit der Bahnbearbeitung erzeugt werden sollen. Zwischen den Konturen können ohne weiteres andere Aktionen, wie z. B. Werkzeugwechseln, Handhaben der Ein- und Ausgabeports, Ein- und Ausschalten der Kühlmittelpumpe etc. durchgeführt werden.

Sie haben gesehen, welche Flexibilität hier erreichbar ist.

Im Bild 3.9 wollen wir Ihnen an einem Beispiel die Eingangsdatendatei und die Ausgangsdatendatei des Bahngenerators zeigen.

Eingangsdatendatei des Bahngenerators

```
;*****************************************
; Datendateibeispiel für den Bahngenerator
;*****************************************

N000001 IMF_PBL_V1.0
N000002 FASTABS X500 Y-500 Z500 A500
N000003 WPZERO
N000004 MOVEREL Y1000 Z1000
N000005 MOVEABS X1000 Y1000 Z-1000
N000006 COOLANT ON
N000007 CCWREL I0 J-1000 X-1000 Y-1000 Z1000
N000008 VEL 10000
N000009 CCWABS I0 J1000 X1000 Y1000 Z-1000
N000010 COOLANT OFF
N000011 REF XYZ
```

Ausgangsdatendatei des Bahngenerators

```
;*****************************************
; Datendateibeispiel für den Bahngenerator
;*****************************************

N000001  IMF_PBL_V1.0
N000002  FASTABS X500 Y-500 Z500 A500
N000003  WPZERO
N000004  PATH
0000000002
0000000001
07002CA22C4AD6C6B044571B8A48DED284420000000000007A44000
07A4400
000000F304353FF304353F000000000000000000000000000000000
000000000000002
0700907A084A16C10B45B35D5A480000000000007A44000000000000
FAC42E
F9E43E000000002EF964BF000000000000000000000000000000000
00000
N000005  PATHEND
N000006  COOLANT ON
N000007  PATH
0000000003
0000000001
```

```
080000002CA22C497C59C444012A86439C829542010000007A44DB0
FC93FDB0F49400000000000007AC400007AC400007AC400007A446F
12833A83F9223F0000000002
0A0000401C46000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000
000000000000000003
080000002CA22C497C59C444012A864300000000010000007A44E4C
B964000000000000000000007A4400007A4400007A4400007AC46F1
2833A83F922BF
N000008  PATHEND
N000009  COOLANT OFF
N000010  REF XYZ
```

Bild 3.9: Beispiel für die Eingangs- und die Ausgangsdatendatei des Bahngenerators

In der Datendatei fängt ein Kommentar mit dem Zeichen ";" an. Kommentare auf separaten Zeilen werden nicht bewertet.

Der erste Befehl einer Datendatei muss der Kennungscode IMF_PBL sein. Die Zeichenkette _V1.0 direkt hinter dem Kennungscode hat keine Bedeutung für den Bahngenerator und wird von ihm nicht berücksichtigt.
In diesem Beispiel bedeutet diese Zeichenkette die Versionsnummer des Bahngenerators. Der Kennungscode IMF_PBL ist unbedingt notwendig, weil der Bahngenerator mit Hilfe dieses Kennungscodes seine Eingangsdatendatei von jeder anderen Datei unterscheiden kann.

Der Befehl mit der Satznummer 2 in der Eingangsdatei ist ein Befehl für die Eilbewegung. Er wird unverändert in die Ausgangsdatendatei geschrieben.

Der nächste Befehl mit der Nummer 3 setzt den momentanen Werkzeugpunkt als neuen Werkstücknullpunkt fest. Dieser Befehl ist kein Bewegungsbefehl. Er wird ebenfalls unverändert in die Ausgangsdatendatei geschrieben.

Die Befehle mit den Nummern 4 und 5 sind aufeinander folgende Bewegungsbefehle, die vom Bahngenerator als eine Kontur zusammengefasst werden. Es ist nicht notwendig, eine Bewegungslänge gleich 0 wie bei der X-Achse im Befehl 4 anzugeben.

Weil der Befehl mit der Nummer 6 kein Bewegungsbefehl ist, endet die Kontur hier. Die Kontur, die durch die Bahnbearbeitung erzeugt werden soll, besteht aus den Segmenten 4 und 5. Aus den Parametern dieser beiden Segmente berechnet der Bahngenerator das Geschwindigkeitsprofil, dessen Daten ebenfalls in der Ausgangsdatendatei abgespeichert werden. Der Anfang des Geschwindigkeitsprofils wird durch den Kenncode PATH gekennzeichnet.

In der nächsten Zeile steht die Anzahl der zu dieser Kontur gehörenden Bewegungssegmente.

Um die Anzahl der Segmente in der Ausgangsdatendatei abzuspeichern, sind exakt 12 Byte notwendig (die Zeichen CARRIAGE RETURN und LINEFEED am Zeilenende wurden dabei mitgezählt). Danach stehen die Daten des Geschwindigkeitsprofils für diese Kontur fest.
Jedes Segment der Kontur braucht 3 Zeilen, um den Teil des Geschwindigkeitsprofils über sich zu speichern.
In der ersten Zeile steht die Segmentnummer, die lokal innerhalb der Kontur ist. Einschließlich der Zeichen CARRIAGE RETURN und LINEFEED am Zeilenende belegt die lokale Segmentnummer genau 12 Byte in der Ausgangsdatendatei.

In den nächsten beiden Zeilen stehen die eigentlichen Daten des Geschwindigkeitsprofils für das Segment. Aus Platzgründen werden diese Daten nicht in einem Klar-Text-Format wie bei der lokalen Segmentnummer, sondern in einem internen ASCII-Format gespeichert.
Einschließlich der Zeichen CARRIAGE RETURN und LINEFEED an jedem Zeilenende werden exakt 128 Byte hierfür in der Ausgangsdatendatei reserviert. Insgesamt braucht jedes Segment 140 Byte in der Ausgangsdatendatei, um den Teil des Geschwindigkeitsprofils über sich zu speichern.
Das Ende des Geschwindigkeitsprofils wird durch den Kenncode PATHEND gekennzeichnet.

Der Befehl mit der Nummer 6, der das Ende der ersten Kontur bedeutet, wird wieder unverändert zurückgespeichert.
Die Befehle 7, 8 und 9 werden vom Bahninterpreter zur zweiten Kontur zusammengefasst, die mit der Bahnbearbeitung erzeugt werden soll. Der Befehl 8 ist zwar kein Bewegungsbefehl. wird vom Bahngenerator aber als ein Bewegungsbefehl betrachtet. Auf diese Weise ist es möglich, die Bahngeschwindigkeit für den Kreis im Befehl 9 auf 10 000 zu setzen, ohne die Bewegung stoppen zu müssen.
Das Geschwindigkeitsprofil dieser Kontur wird berechnet und in die Ausgangsdatendatei gespeichert.
Falls die Befehle 10 und 11 Bewegungsbefehle wären, würden sie auch vom Bahngenerator zu einer Kontur zusammengefasst. Aber weil das nicht der Fall ist, endet die zweite Kontur ebenfalls nach drei Segmenten.
Die Befehle 10 und 11 werden unverändert in der Ausgangsdatendatei gespeichert.

Unserer Bahngenerator ist in der Lage, Geschwindigkeitsprofile für die 3D-Bahnbearbeitung zu erzeugen. Sie sollten aber beachten, dass die vierte Achse während der Bahnbearbeitung einfach ignoriert wird. Bei unserem Zwischenformat ist es zwar möglich, alle 4 Achsen bei separaten

Bewegungsbefehlen interpolierend zu fahren, aber es ist momentan nicht möglich, die vierte Achse während der Bahnbearbeitung mitzuverwalten.

Am Beispiel im Bild 3.9 können Sie sich ungefähr vorstellen, wie der Bahngenerator arbeitet. Beachten Sie, dass der Bahngenerator die Befehle neu nummeriert.
Die Kenncodes PATH und PATHEND für jede zu einer Bahn zusammengefassten Kontur werden vom Bahngenerator ebenfalls mit Befehlsnummern versehen. Die Information über die Anzahl der zu einer Kontur gehörenden Segmente sowie die lokale Segmentnummer innerhalb einer Kontur sind in vielen Fällen sehr nützlich. Hier können wir uns z. B. vorstellen, dass Sie auf Ihrer Bedieneroberfläche eine Anzeige aufbringen, um dem Bediener zeigen zu können, wie viel Prozent der Kontur schon abgearbeitet sind.

Die Befehlsnummern für PATH und PATHEND sowie die lokalen Segmentnummern können Sie in Ihrem Anwenderprogramm dazu benutzen, eine Möglichkeit zu schaffen, dass der Bediener das Werkstück nach dem Beheben eines Bearbeitungsfehlers (z. B. im Fall eines gebrochenen Fräsers) von der tatsächlichen Fehlerstelle aus weiter bearbeiten lassen kann (siehe Treiberfunktion 34).

Nachdem die Bahndaten erzeugt und in der Ausgangsdatendatei gespeichert wurden, entsteht logischerweise die Frage, wie der Interpreter mit diesen Bahndaten eine Bahn erzeugen kann. Hier können wir schon kurz sagen, dass sich die Aufgabe des Interpreters, genau wie bei anderen Befehlen, ausschließlich darauf beschränkt, die Bahndaten aus der Datei zu lesen und sie dem Treiber über die entsprechende Funktion zu übergeben (siehe Treiberfunktion 33). Es ist die alleinige Sache des Treibers, wie er aus diesen Bahndaten eine Bahn erzeugt.

3.2.5 Wie kann man den Bahngenerator benutzen?

Die Aufgabe des Bahngenerators ist das Berechnen und das Erzeugen des Geschwindigkeitsprofils. Er ist kein Bestandteil des Treibers, sondern ein Bestandteil der Anwenderprogramme.

Im Rahmen unseres Produktes bieten wir Ihnen optional die Funktion „Bahnsteuerung“ an. Wir stellen Ihnen den Bahngenerator sowohl in der Form von mit Compiler Microsoft C7.00 erzeugten Objektfiles als auch in der Form eines ausführbaren Programms zur Verfügung.
Falls Sie Ihre Anwendungsprogramme mit einer Entwicklungsumgebung von Microsoft entwickeln, können Sie ohne weiteres diese Objektfiles in Ihre Programme einbinden. Wenn Sie diese Objektfiles nicht benutzen wollen oder können, weil z. B. Ihre Entwicklungsumgebung nicht eine von Microsoft ist**,** können Sie mit dem mitgelieferten EXE-Programm die Bahndaten berechnen.

Im Lieferumfang sind für eine 3D-Bahnsteuerung folgende Dateien enthalten:

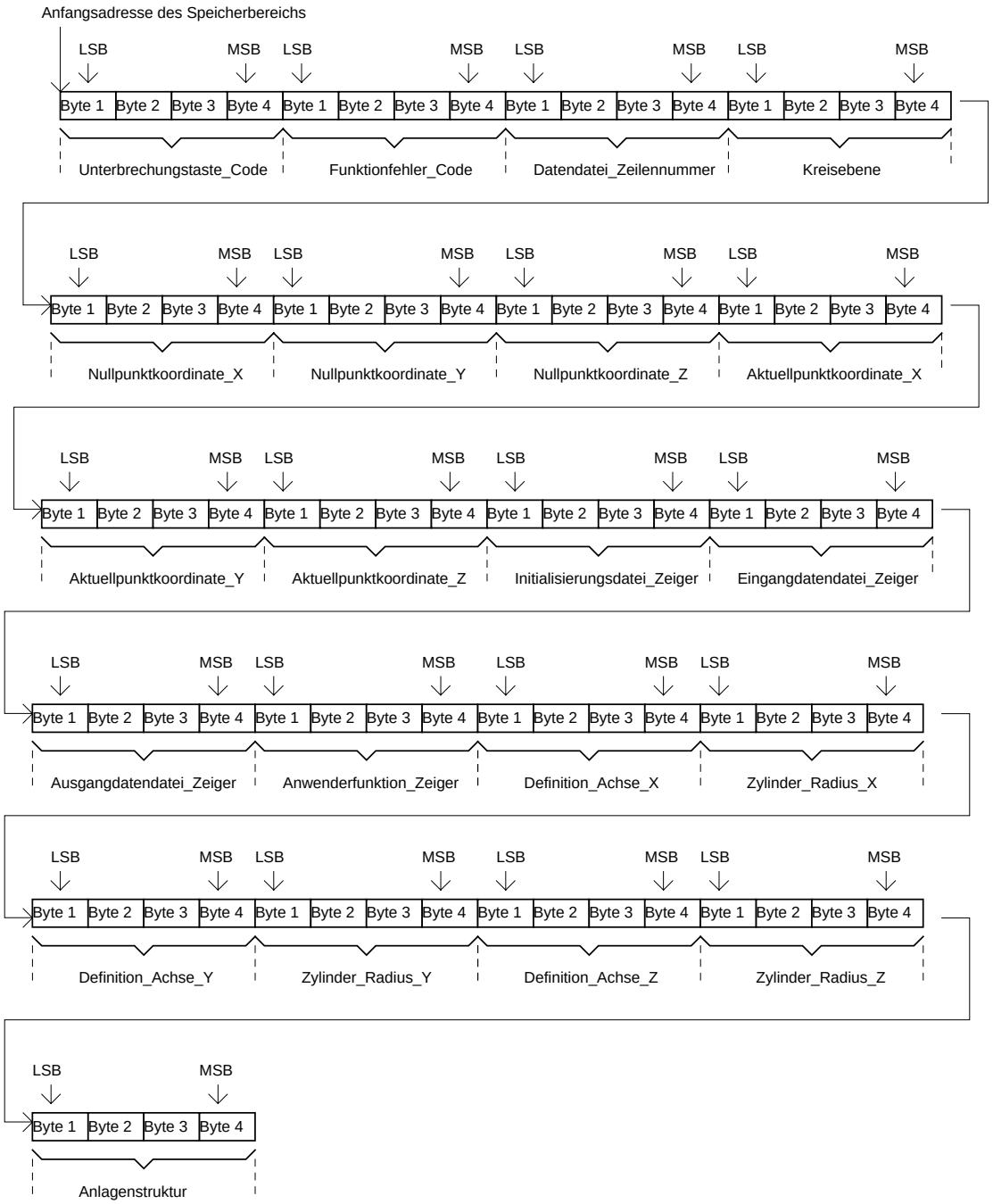
BAHN_S.OBJ	für das Speichermodell SMALL,
BAHN_M.OBJ	für das Speichermodell MEDIUM,
BAHN_C.OBJ	für das Speichermodell COMPACT,
BAHN_L.OBJ	für das Speichermodell LARGE und
BAHN.EXE	ausführbares Programm.

3.2.5.1 Benutzung der Objektfiles zur Berechnung der Bahndaten

Falls Sie die Bahnbearbeitung wünschen, müssen Sie das passende Objektfile in Ihr Programm einbinden und die entsprechende Funktion aufrufen, um das Geschwindigkeitsprofil aus einer Datendatei mit *isel*-Zwischenformat zu erzeugen. Danach ist es die Aufgabe des Interpreters, in Ihrem Anwenderprogramm unter anderem diese Bahndaten aus der mit dem Bahngenerator erzeugten Datendatei zu lesen und dem Treiber über die entsprechende Funktion zu übergeben.
Der Aufruf des Bahngenerators in Ihrem Programm kann wie folgt aussehen:

```
void BahnDatenGenerator(char far *ParameterZeiger);
/*Funktionsdeklaration*/

void main(void)
{
   char far *ParameterZeiger ;

      .
      .
      .

   BahnDatenGenerator(ParameterZeiger) ; /* Aufruf der Funktion zum
                                          Berechnen des Geschwindigkeitsprofils
*/

      .
      .
      .
}
```

In Ihrem Programm brauchen Sie nur die Funktion *BahnDatenGenerator()* aufzurufen. Der Rest wird von dieser Funktion erledigt. Beachten Sie, dass die aufrufende Funktion eine Stack-Größe von mindesten 10000 Bytes haben muss. Die Kommunikation zwischen Ihrem Programm und der Funktion *BahnDatenGenerator()* geschieht über einen in Ihrem Anwenderprogramm definierten Speicherbereich, dessen Anfangsadresse der Far-Zeiger *ParameterZeiger* ist. Dieser Speicherbereich ist genau 84 Byte groß.
Die Organisation dieses Speicherbereichs ist im Bild 3.10 zu sehen.



LSB = Last Significant Byte (Niederwertigste Byte)
MSB= Most Significant Byte (Höchstwertigste Byte)

Bild 3.10: Speicherbereich für den Datenaustausch mit der Funktion *BahnDatenGenerator()*

In dem ersten 4-Byte-Feld *Unterbrechungstaste_Code* können Sie den ASCII-Code einer beliebigen Taste der Tastatur definieren. Mit dem Bestätigen dieser Taste ist der Anwender in der Lage, die Funktion *BahnDatenGenerator()* während ihrer Abarbeitung sofort zu unterbrechen. Er kehrt dann zu der aufrufenden Funktion mit dem entsprechenden Fehlercode zurück.

Beachten Sie hier, dass ein erweiterter IBM-ASCII-Code nicht akzeptiert wird, d. h. der ASCII-Code der Unterbrechungstaste liegt im Bereich 0 ... 127.

In dem Feld *Funktionsfehler_Code* gibt die Funktion *BahnDatenGenerator()* der aufrufenden Funktion einen Fehlercode zurück. Über diesen Fehlercode kann die aufrufende Funktion feststellen, ob die Funktion *BahnDatenGenerator()* fehlerfrei ausgeführt wurde oder nicht. Anhand dieses Fehlercodes können Sie ermitteln, welcher Fehler aufgetreten ist. Die verschiedenen Fehlermöglichkeiten werden wir am Ende dieses Kapitels noch genauer behandeln.

Über das Feld *Datendatei_Zeilennummer* gibt die Funktion *BahnDatenGenerator()* der aufrufenden Funktion im Fehlerfall die Zeilennummer der Eingangsdatendatei zurück. Anhand dieser Zeilennummer können Sie feststellen, an welcher Stelle in der Eingangsdatendatei die Berechnung abgebrochen wurde. Dies ist sehr nützlich für die Fehleranalyse.

Im fehlerfreien Fall ist die Zeilennummer gleich 0. Beachten Sie hier, dass die zurückgegebene Zeilennummer der Eingangsdatendatei sich wegen zusätzlicher Kommentarzeilen sehr stark von der Befehlsnummer in der Datei unterscheiden kann.

Im Feld *Kreisebene* müssen Sie die Kreisebene definieren. Solange die Kreisebene nicht durch einen entsprechenden Befehl in der Eingangsdatendatei geändert wird, findet die Kreis- sowie die Helixinterpolation auf dieser Ebene statt. Die Zuordnung zwischen dem Wert im Feld *Kreisebene* und der Kreisebene lautet:

Wert	Kreisebene
0	XY-Ebene
1	XZ-Ebene
2	YZ-Ebene
(sonst)	YZ-Ebene.

Über den Feldern *Nullpunktkoordinate_X*, *Nullpunktkoordinate_Y* und *Nullpunktkoordinate_Z* müssen Sie die Koordinaten des Werkstücknullpunktes in Bezug auf den Referenzpunkt der Maschine angeben. d. h. Sie müssen angeben, wo der Werkstücknullpunkt definiert ist, bevor es anfängt, die Datendatei zu bearbeiten. Diese Angabe ist deswegen wichtig, weil alle Absolutkoordinaten in den Bewegungsbefehlen den Werkstücknullpunkt als Bezugspunkt nehmen. Die Angabe der Koordinaten erfolgt in Mikrometer [µm].

Über die Felder *Aktuellpunktkoordinate_X*, *Aktuellpunktkoordinate_Y* und *Aktuellpunktkoordinate_Z* müssen Sie genau angeben, wo das Werkzeug steht, direkt bevor es anfängt, die Datendatei zu bearbeiten. Die Einheit der Koordinaten ist Mikrometer [µm].

Beachten Sie, dass der Bezugspunkt für die Berechnung dieser Aktuellpunktkoordinaten nicht der Referenzpunkt, sondern der Werkstücknullpunkt ist. Die Koordinaten des Werkstücknullpunktes im Bezug auf den Referenzpunkt werden der Funktion BahnDatenGenerator() über die oben erwähnten Felder *Nullpunktkoordinate_X*, *Nullpunktkoordinate_Y* und *Nullpunktkoordinate_Z* übergeben.

Über das Feld *Initialisierungsdatei_Zeiger* bekommt die Funktion *BahnDatenGenerator()* einen Far-Zeiger, der auf eine Zeichenkette zeigt. Diese Zeichenkette ist der Name der Initialisierungsdatei der Anlage (siehe Handbuch für das Konfigurationsprogramm PARKON.EXE).
Der Dateipfad muss angegeben werden. Diese Angabe ist deswegen notwendig, weil sämtliche Anlagenparameter in dieser Initialisierungsdatei enthalten sind. Die Funktion BahnDatenGenerator() braucht viele dieser Anlagenparameter, um das Geschwindigkeitsprofil zu berechnen.
Über das Feld *Eingangdatendatei_Zeiger* müssen Sie einen Far-Zeiger auf eine Zeichenkette angeben. Diese Zeichenkette ist der Name der Eingangsdatendatei, aus der Sie Bahndaten berechnen lassen wollen. Der komplette Pfad muss angegeben werden.

Über das Feld *Ausgangdatendatei_Zeiger* definieren Sie, wie die Ausgangsdatendatei heißen soll. In dieser Ausgangsdatendatei werden die Bahndaten gespeichert. Den Pfad müssen Sie angeben. Falls eine Datei mit demselben Namen in diesem Verzeichnis bereits vorhanden ist, wird diese bereits vorhandene Datei ohne jede Warnung gelöscht.

Über das Feld *Anwenderfunktion_Zeiger* können Sie einen Far-Zeiger angeben, der auf eine von Ihnen definierte Funktion zeigt. Beachten Sie, dass der Wert in diesem Feld erst dann als ein Funktionszeiger interpretiert wird, wenn der Wert ungleich 0 ist. Sonst wird dieses Feld einfach ignoriert. Diese Funktion darf nur einen einzigen Parameter haben. Es ist eine Long-Zahl.
Die Deklaration dieser Funktion muss so aussehen:

```
void AnwenderFunktion (long Zahl);
```

Diese Funktion wird mit Hilfe des angegebenen Zeigers regelmäßig innerhalb der Funktion *BahnDatenGenerator()* aufgerufen. Über den einzigen Parameter dieser Funktion wird der Prozentsatz (0 ... 100) übergeben.
Diese Zahl sagt aus, wie viel Prozent der Eingangsdatendatei fertig abgearbeitet sind.
Diese von Ihnen selbst definierte Funktion dient als ein Informationskanal zwischen Ihrem Anwenderprogramm und der Funktion *BahnDatenGenerator()* während der Zeit, in der das Geschwindigkeitsprofil berechnet wird. So können Sie sehr einfach z. B. eine Anzeige auf den Bildschirm bringen, die anzeigt, wie viel Prozent des Geschwindigkeitsprofils schon fertig berechnet sind.

Der Treiber ab der Version 3.1 bietet die Möglichkeit, zwischen den Achsen umzuschalten (siehe Funktion 57). Nach dem Umschalten können Sie mit der neuen Achsenzuordnung ganz normal arbeiten. Die Bahnbearbeitung ist weiterhin möglich wie alle anderen Funktionen auch.

Über den Feldern *Definition_Achse_X*, *Definition_Achse_Y*, *Definition_Achse_Z* können Sie dem *BahnDatenGenerator()* mitteilen, welche Achse Sie als X-Achse bzw. als Y-Achse bzw. als Z-Achse während der Bahnbearbeitung haben wollen. Die Zuordnung zwischen den Achsnummern in diesen Feldern und der wirklichen Achsen lautet:

Wert	Achse
1	X-Achse
2	Y-Achse
3	Z-Achse
4	A-Achse

Am folgenden Beispiel wollen wir das Zusammenspiel zwischen der Achsumschaltung und der Bahnbearbeitung erläutern. Falls

Definition_Achse_X = 4
Definition_Achse_Y = 3
Definition_Achse_Z = 1

sind, betrachtet der *BahnDatenGenerator()* die A-Achse der Anlage (Achsnummer 4) als X-Achse, die Z-Achse der Anlage (Achsnummer 3) als Y-Achse und die X-Achse der Anlage (Achsnummer 1) als Z-Achse. Die achsspezifischen Parameter wie die Achsbeschleunigung, Achsgeschwindigkeit usw. werden aus der Initialisierungsdatei gelesen.
Die Koordinaten der Achsen X, Y und Z innerhalb der Bahn werden mit den gelesenen Parametern der Achsen A, Z und X kombiniert. Daraus werden die Bahndaten berechnet.
Die Übergabe der Bahndaten zum Treiber erfolgt wie normal. Aber vor der Übergabe müssen Sie mit Hilfe der Funktion 57 dem Treiber die gewünschte Achszuordnung mitteilen, um eine korrekte Ausgabe zu bekommen. In den meisten Fällen wird die Achsumschaltung gar nicht benötigt. In diesem Fall müssen Sie

Definition_Achse_X = 1
Definition_Achse_Y = 2
Definition_Achse_Z = 3

setzen.

Parallel zu der Achsenumschaltung bietet der Treiber ab der Version 3.1 noch die Möglichkeit, auf einem zylindrischen Körper zu gravieren (siehe Funktion 58). Über den Feldern *Zylinder_Radius_X*, *Zylinder_Radius_Y* und *Zylinder_Radius_Z* können Sie dem *BahnDatenGenerator()* die Zylinderradien

der entsprechenden Achsen übergeben. Daraus werden die Bahndaten für das Gravieren auf der Zylinderoberfläche berechnet.
Das Gravieren auf einer Zylinderoberfläche ist somit im Bahn-Modus ohne weiteres möglich. Die Übergabe der Bahndaten zum Treiber erfolgt wie normal. Aber vor der Übergabe der Bahndaten müssen Sie mit Hilfe der Funktion 58 dem Treiber auch den selben Zylinderradius wie hier mitteilen, um eine korrekte Bahndatenausgabe zu erreichen.
Beachten Sie, dass die Radienangaben in den Feldern *Zylinder_Radius_X*, *Zylinder_Radius_Y* und *Zylinder_Radius_Z* immer zusammen mit den Achsnummern in den Feldern *Definition_Achse_X*, *Definition_Achse_Y*, *Definition_Achse_Z* sind. Wenn z. B.

Definition_Achse_Y = 4 und
Zylinder_Radius_Y = 10000

sind, interpretiert der *BahnDatenGenerator()* die A-Achse der Anlage als eine Y-Achse. Der auf der A-Achse der Anlage befestigten Zylinderkörper hat einen Radius von 10 000 µm.
Im Zusammenhang mit den achsspezifischen Parametern der A-Achse der Anlage und dem Zylinderradius von 10000 µm werden die Koordinaten der Y-Achse in der Eingangsdatendatei in die entsprechende Bahndaten für die Y-Achse berechnet.
Beim Gravieren können Sie dem Treiber die Bahndaten für die Y-Achse übergeben, nachdem Sie mit Hilfe der Funktion 57 die A-Achse der Anlage als Y-Achse umdefiniert und mit der Funktion 58 den Zylinderradius für die A-Achse der Anlage (d. h. die momentane Y-Achse) übergeben haben.

Die Werte in den Feldern *Zylinder_Radius_X*, *Zylinder_Radius_Y* und *Zylinder_Radius_Z* müssen gleich 0 sein, falls Sie nicht die Absicht haben, auf einem Zylinder zu gravieren.
Bei einem Zylinderradius ungleich 0 wird die entsprechende Rundachse als eine Linearachse interpretiert. Durch diese Tatsache und durch die Achsumschaltung kann die Struktur der Anlage geändert werden. Deswegen müssen Sie in dem letzten Feld *Anlagenstruktur* die neue Struktur angeben. Die Zuordnung zwischen dem Wert in diesem Feld und der Anlagenstruktur lautet:

Wert	Anlagenstruktur
1	X_TTT-Struktur
2	XY_TTT-Struktur
3	XYZ_TTT-Struktur
4	KEIN_TTT-Struktur.

Mit Hilfe der Angabe über die neue Anlagenstruktur entscheidet der *BahnDatenGenerator()*, ob eine Bahnbearbeitung möglich ist.
Die Funktion *BahnDatenGenerator()* legt einen Fehlercode in das Feld *Funktionsfehler_Code* und kehrt zu der aufrufenden Funktion zurück.

Anhand dieses Fehlercodes kann die aufrufende Funktion feststellen, ob die Berechnung des Geschwindigkeitsprofils erfolgreich ist. Im Kapitel 3.2.5.3 werden diese Fehlermöglichkeiten ausführlich erläutert

3.2.5.2 Benutzung des Programms BAHN.EXE zur Berechnung der Bahndaten

Um die Bahndaten zu berechnen, können Sie anstelle von Objektfiles auch das ausführbare Programm BAHN.EXE benutzen. Über eine Kommandozeile können Sie BAHN.EXE direkt von Ihrem Anwenderprogramm heraus aufrufen.

Der Aufruf von BAHN.EXE innerhalb eines C-Programms kann wie folgt aussehen:

```
#include  <process.h>
#include  <sTdlib.h>
#include  <errno.h>

void main( void )
{
 int iFehlerCode ;
 char cKommandozeile[128+1] ;

/ * Bilden die Kommandozeile */
                                    .
                                    .
                                    .
/* Aufrufen der Kommandozeile */

 iFehlerCode = system( cKommandozeile ) ;
                                    .
                                    .
                                    .
}
```

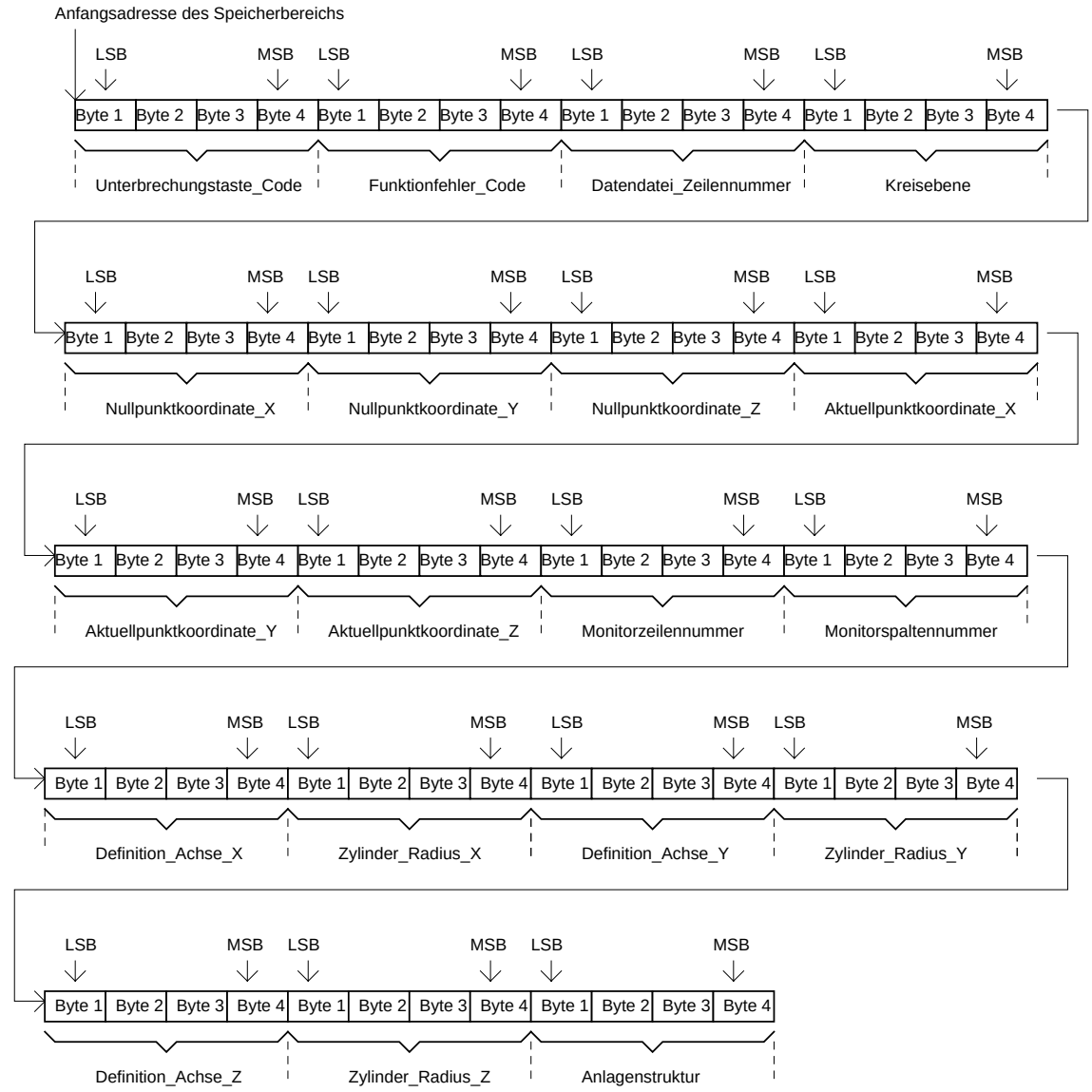
Das Format der Kommandozeile *cKommandozeile* lautet:

BAHN.EXE Par_Datei Input_Datei Output_Datei Ini_Datei

Die Bedeutung der einzelnen Parameter in der Kommandozeile ist:

BAHN.EXE
Dies ist das ausführbare Programm zur Berechnung der Bahndaten.

Par_Datei:
Es handelt sich hier um eine Parameterdatei, die die Anfangsparameter zur Berechnung der Bahndaten beinhaltet. Sie müssen eine Datei im binären Datenformat erzeugen mit folgender Struktur:



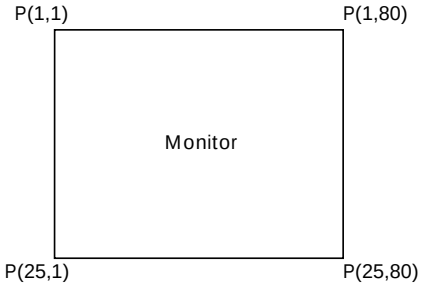
LSB = Last Significant Byte (Niederwertigste Byte)
MSB= Most Significant Byte (Höchstwertigste Byte)

Bild 3.11: Struktur der Parameterdatei zur Berechnung der Bahndaten mit BAHN.EXE

Abgesehen von dem Fehlen der 4-Bytes-Datenfelder *Initialisierungs_datei_Zeiger*, *Eingangdatendatei_Zeiger*, *Ausgangdatendatei_Zeiger* sowie *Anwenderfunktion_Zeiger* und das Hinzufügen der 4-Bytes-Datenfelder *Monitorzeilennummer* sowie *Monitorspaltennummer* ist die Struktur der Parameterdatei im Bild 3.11 weitgehend identisch zu der Struktur des Speicherbereichs im Bild 3.10 für die Berechnung der Bahndaten mit *BahnDatenGenerator()*.
Die Bedeutung der Datenfelder im Bild 3.11 können Sie im Kapitel 3.2.5.1 nachlesen.

Bei der Benutzung von BAHN.EXE ist es nicht möglich, eine Anwenderfunktion periodisch aufrufen zu lassen. Um eine prozentuale Angabe über die bereits

ausgeführten Bahndatenberechnung online machen zu können, müssen Sie die beiden Parameter *Monitorzeilennummer* und *Monitorspaltennummer* benutzen. Diese beiden Parameter legen die Anfangsposition auf dem Bildschirm fest, an der das Programm BAHN.EXE periodisch die gewünschte Prozentangabe ausgibt. Diese Prozentangabe hat die Form „xxx %“ und belegt genau 4 Textspalten einer Zeile. Diese beiden Parameter haben den Wertbereich, wie im Bild 3.12 dargestellt ist.



Monitorzeilennummer = 1, ... , 25
Monitorspaltennummer = 1, ... , 80

Bild 3.12: Wertbereich für *Monitorzeilennummer* und *Monitorspaltennummer*

Falls die Parameter *Monitorzeilennummer* und *Monitorspaltennummer* eine Position außerhalb des Monitors (z. B. 0,0) darstellen, erfolgt keine Ausgabe.
Das Erstellen der Parameterdatei im Binärformat ist eine ziemlich einfache Sache. Zuerst müssen Sie in Ihrem Programm einen Speicherbereich mit der gleichen Struktur wie im Bild 3.11 definieren. Danach erfolgt die Initialisierung der Parameter in diesem Speicherbereich. Anschließend wird dieser komplette Speicherbereich binär in eine Datei geschrieben.
Diese Datei ist dann die Parameterdatei für Ihre Berechnung.

Nach der Berechnung der Bahndaten bekommen Sie ähnlich wie bei der Benutzung vom *BahnDatenGenerator()* in dem Feld *Funktionsfehler_Code* den Fehlercode für die Berechnung.
Die Fehlermöglichkeiten und deren Bedeutungen werden ausführlich im Kapitel 3.2.5.3 behandelt.

Input_Datei
Die Input_Datei ist die Datendatei im *isel*-Zwischenformat, aus der die Bahndaten berechnet werden sollen.

Output_Datei
Die Output_Datei ist die Bahndatendatei, in der das Ergebnis der Berechnung abgespeichert werden soll.

Ini_Datei:
Die Ini_Datei ist die Initialisierungsdatei für Ihre Anlage.

3.2.5.3 Fehlermöglichkeiten bei der Berechnung der Bahndaten

Bei der Berechnung der Bahndaten können verschiedene Fehler auftreten. Sowohl bei der Benutzung von *BahnDatenGenerator()* als auch bei der Benutzung von BAHN.EXE wird der Fehlercode über dem Feld *Funktionfehler_Code* zurückgeliefert. Die Fehlermöglichkeiten und deren Bedeutungen werden im Folgenden erläutert.

Fehlercode = 0: Fehlerfrei
Die Ausgangsdatendatei mit Bahndaten konnte fehlerfrei erzeugt werden. Im Feld *Datendatei_Zeilennummer* steht die Zahl 0.

Fehlercode = 1: Unterbrechungstaste-Fehler
Der Code für die Unterbrechungstaste im Feld *Unterbrechungstaste_Code* ist fehlerhaft, d. h. der Wert in diesem Feld liegt außerhalb des Bereichs 0 ... 127.

Fehlercode = 2: Unterbrechungs-Fehler
Während ihrer Arbeit wurde die Funktion *BahnDatenGenerator()* durch das Betätigen der im Feld *Unterbrechungstaste_Code* definierten Taste unterbrochen. Im Feld *Datendatei_Zeilennummer* steht die Nummer der zuletzt bearbeiteten Zeile der Eingangsdatendatei.

Fehlercode = 3: Anlagenstruktur-Fehler
Die Bahnbearbeitung kann nur in solchen Anlagen durchgeführt werden, deren Hauptachsen ein kartesisches Koordinatensystem auf der Ebene oder im Raum bilden. Die Bahnbearbeitung ist nur dann möglich, wenn Ihre Anlage die Struktur vom Typ XY_TTT oder vom Typ XYZ_TTT hat (siehe Handbuch für das Konfigurationsprogramm PARKON.EXE).

Fehlercode = 4: Initialisierungsdateiname-Fehler
Der Name der Initialisierungsdatei, dessen Zeiger im Feld *Initialisierungsdatei_Zeiger* steht, ist offensichtlich falsch. Entweder hat der Dateiname nicht die Form *.INI oder die genannte Datei existiert gar nicht.

Fehlercode = 5: Initialisierungsdateiformat-Fehler
Die Datei, deren Name durch den Zeiger im Feld *Initialisierungsdatei_Zeiger* lokalisiert ist, hat nicht das interne Format einer Initialisierungsdatei (siehe Handbuch für das Konfigurationsprogramm PARKON.EXE).

Fehlercode = 6: Eingangsdatendateiname-Fehler
Die Eingangsdatendatei, deren Name durch den Zeiger im Feld *Eingangsdatendatei_Zeiger* lokalisiert ist und aus der die Ausgangsdatendatei mit Bahndaten erzeugt werden soll, existiert nicht oder zumindest nicht in dem angegebenen Pfad.

Fehlercode = 7: Eingangsdatendateiformat-Fehler
Die Eingangsdatendatei, aus der die Ausgangsdatendatei mit Bahndaten erzeugt werden soll, besitzt nicht das interne Format, das eine Datendatei mit *isel*-Zwischenformat haben muss.

Fehlercode = 8: Eingangsdatendateisegment-Fehler
Die Eingangsdatendatei hat einen fehlerhaften Befehl. Durch die zurückgegebene Zeilennummer im Feld *Datendatei_Zeilennummer* kann die aufrufende Funktion ermitteln, auf welcher Zeile der Eingangsdatendatei dieser fehlerhafte Befehl steht. In den meisten Fällen handelt es sich hier um Bewegungsbefehle mit der Kreis- bzw. Helixinterpolation.

Fehlercode = 9: Ausgangsdatendateinamen-Fehler
Die Ausgangdatendatei, deren Name durch den Zeiger im Feld *Ausgangdatendatei_Zeiger* lokalisiert ist, konnte nicht erzeugt werden. In den meisten Fällen ist der angegebene Pfad nicht existent oder der angegebene Name entspricht nicht der Namenskonvention von MS DOS.

Fehlercode = 10: Parameterdatei-Fehler
Dieser Fehler kann in Verbindung mit der Parameterdatei auftreten. Er deutet darauf hin, dass die übergebenen Parameter nicht korrekt sind. Die Ursache des Fehlercodes 10 kann sein:

- par_Datei ist nicht vorhanden oder
- Dateihändler ist nicht mehr vorhanden oder
- falsches Format der par-Datei.

Dieser Fehler tritt nur auf, wenn Sie für die Berechnung der Bahndaten das Programm BAHN.EXE benutzen.

Fehlercode = 11: Zeilenkommando-Fehler
Beim Aufruf des Programms BAHN.EXE zur Berechnung der Bahndaten müssen Sie eine Kommandozeile bilden, auf der notwendige Parameter stehen. Wenn die Kommandozeile fehlerhaft ist, wie z. B. ein Parameter fehlt, erscheint dieser Fehler.
Dieser Fehler tritt nur auf, wenn Sie für die Berechnung der Bahndaten das Programm BAHN.EXE benutzen.

Fehlercode = 12: Achsnummer-Fehler
Die Angabe der Achsnummer in den Feldern *Definition_Achse_X*, *Definition_Achse_Y*, *Definition_Achse_Z* ist fehlerhaft. Die Ursache dieses Fehlers kann sein:

- die Achsnummer ist kleiner als 1 oder
- die Achsnummer ist größer als die Achsanzahl der Anlage oder
- mindestens zwei Achsnummern in den Feldern sind gleich.

Dieser Fehler tritt sowohl bei der Benutzung von Objektfiles als auch bei der Benutzung von BAHN.EXE auf.

Beachten Sie, dass die Ausgangsdatendatei im Fehlerfall (d. h. Fehlercode ungleich 0) gelöscht wurde, bevor die Funktion *BahnDatenGenerator()* oder das Programm BAHN.EXE zurückkehrt, d. h. im Fehlerfall gibt es keine Ausgangsdatendatei.

3.2.6 Sicherheit der Anlage

Um die Sicherheit der Servoanlagen zu gewährleisten, werden verschiedene Maßnahmen in den Treiber integriert. Diese Maßnahmen werden in den folgenden Unterkapiteln erläutert. Alle diese Sicherheitsmaßnahmen (Watch-Dog-Signal, Überwachung der Hardware-Endschalter, der Software-Endschalter und der Nachlauffehler) laufen nebeneinander und ergänzen sich. Dadurch bekommen Sie als Anwender eine wesentlich größere Sicherheit für Ihre Servoanlage.

3.2.6.1 Watch-Dog-Signal und Kontroll-Byte

Nach der Installation im Speicher generiert der Treiber Hardware-Interrupts über die Interruptquelle IRQ10 oder IRQ11 in regelmäßigen Abständen von ein paar Millisekunden (siehe Handbuch für das Programm PARKON).
Das Auslösen dieses Interrupts hängt nicht davon ab, ob eine Bewegung im Hintergrund aktiv ist oder nicht. In der Interrupt-Serviceroutine wird ein so genanntes Watch-Dog-Signal an dem Pin 40 des 50-poligen RIBBON-Steckverbinders (siehe Kapitel 2.6) ausgegeben.

Der Zustand des Watch-Dog-Signals ändert sich periodisch zwischen HIGH-Zustand und LOW-Zustand. Dieses Watch-Dog-Signal sollten Sie benutzen, um Ihren Servocontroller zu überwachen.
Wenn das Watch-Dog-Signal in einer bestimmten Zeit seinen Zustand nicht ändert, muss der Servocontroller nach einer definierten Zeit von selbst ausschalten.
Der Sinn des Watch-Dog-Signals liegt in der Überwachung der Software. Falls die Software auf dem Steuerrechner aus einem bestimmten Grund abstürzt, ändert sich das Watch-Dog-Signal nicht mehr. Nach einer bestimmten Zeit schaltet sich der Servocontroller von selbst aus. Dadurch entsteht keine Gefahr.

In der Interrupt-Serviceroutine wird nach dem Generieren des Watch-Dog-Signals der Inhalt des sogenannten Kontroll-Bytes eingelesen. Das Kontroll-Byte ist ein Eingabeport, den Sie optional benutzen können. Mit Hilfe des Programms PARKON können Sie die Portadresse, die Benutzungsmaske sowie den Portwert für den fehlerfreien Fall definieren.

Falls der eingelesene Portwert und der fehlerfreie Portwert unterschiedlich sind, wird die Bewegung der Anlage sofort unterbrochen. Intern im Treiber wird ein

Fehlerflag gesetzt. Solange dieser Flag aktiv ist, ist eine Bewegung nicht möglich. Über dieses Kontroll-Bytes können Sie bis zu 8 Hardwaresignale zurückführen. Jedes dieser Hardwaresignale soll eine bestimmte hardwaremäßige Fehlerquelle darstellen.

Die häufigste Anwendung findet dieses Kontroll-Byte in der Überwachung des Stroms des Servocontrollers. Falls der Strom des Servocontrollers ausfällt, merkt der Treiber es durch das Lesen des Kontroll-Bytes. Die Bewegung wird sofort gesperrt.
Wenn der Strom irgendwann wieder da ist, ist eine Bewegung weiterhin unmöglich, solange die Reset-Funktion (Treiberfunktion 2) noch nicht aufgerufen wird. Eine unkontrollierte Bewegung ist also unmöglich. Das Überwachen der Encoder-Signale sind eine weitere mögliche Anwendung des Kontroll-Bytes.

3.2.6.2 Hardware-Endschalter

Um die Mechanik zu schützen, hat jede Achse einer Anlage normalerweise zwei sogenannte Hardware-Endschalter (Achsenanschläge).
Am Achsenanschlag, muss die Bewegung sofort unterbrochen werden. Sonst kann es passieren, dass der Regler mit seinem Integrationsanteil aufgrund der vorhandenen Abweichung zwischen der Soll- und der Ist-Position zu einem hohen Motorstrom führt. Der hohe Motorstrom erzeugt auf der einen Seite eine große Kraft, die zu mechanischen Schäden führen kann, und auf der anderen Seite eine starke Überhitzung, die im schlimmsten Fall den Motor beschädigen kann.

Unser Treiber bietet Ihnen die Unterstützung für bis zu zwei Hardware-Endschaltern pro Achse. Die Portadresse und die Bitnummer, an denen der jeweilige Hardware-Endschalter angeschlossen ist, sowie den aktiven Pegel (High oder Low) können Sie durch das mitgelieferte Konfigurationsprogramm PARKON einstellen.

Während einer Bewegung überwacht der Treiber ständig die Hardware-Endschalter. Falls einer der Schalter betätigt ist, wird die momentane Bewegung sofort unterbrochen. Ein eventuell noch vorhandener Rest des Bewegungssegments geht unwiderruflich verloren. Danach wird intern ein Flag gesetzt. Solange dieses Flag gesetzt ist, kann das Anwenderprogramm keine weiteren Bewegungssegmente mehr zum Treiber schicken, falls der Teach-In-Modus nicht eingeschaltet ist (siehe Kapitel 3.3.9).
Im Teach-In-Modus werden weitere Bewegungssegmente vom Treiber akzeptiert, um eine Teach-In-Bewegung realisieren zu können. Es sei betont, dass die Hardware-Endschalter im Teach-In-Modus weiterhin überwacht werden, d. h. eine Bewegung wird auch sofort unterbrochen, falls einer der Schalter betätigt ist. Im Teach-In-Modus wird das oben genannte Flag im Fehlerfall aber nicht gesetzt, d. h. das Anfahren auf einen der Schalter im Teach-In-Modus hat keine Nachwirkungen für den weiteren Ablauf. Auch den

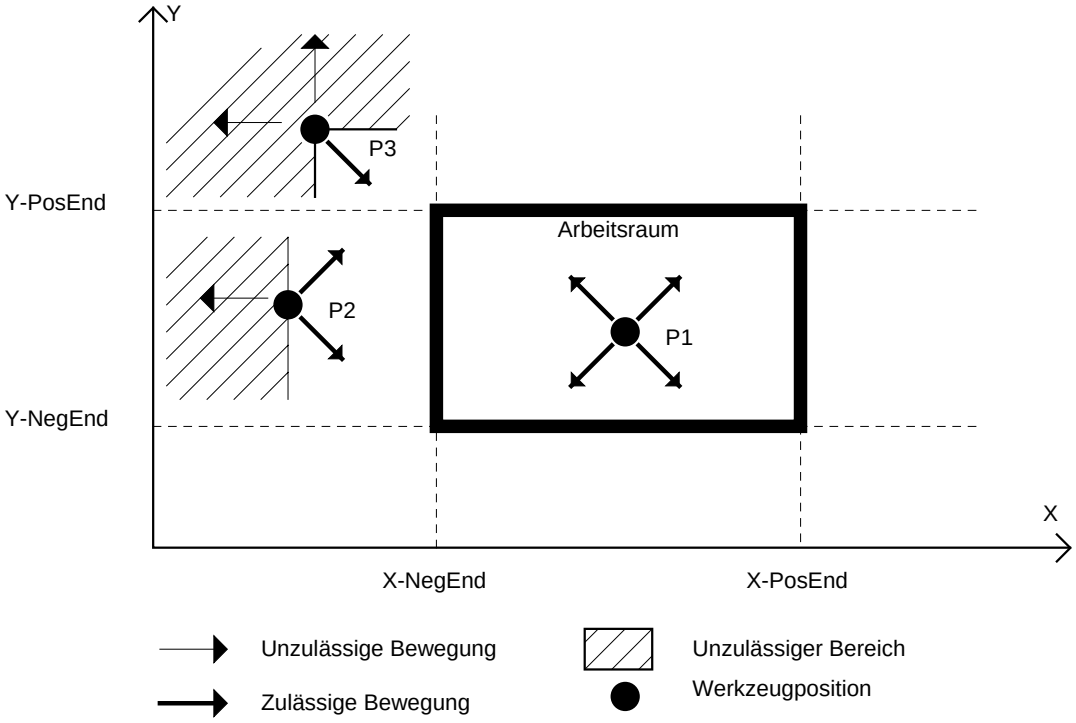
Zustand dieses Flag kann das Ein- bzw. das Ausschalten des Teach-In-Modus nicht beeinflussen.
Dieses Flag, falls einmal gesetzt, kann nur durch ein Reset (siehe Kapitel 3.3.2) zurückgesetzt werden. Dies sollte auch getan werden, weil sehr viele Treiberfunktionen bei gesetztem Flag gesperrt sind.
Die Maßnahmen, die der Treiber im Falle eines aktiven Hardware-Endschalters trifft, sind softwaremäßige Maßnahmen. Solche Maßnahmen sind nicht ausreichend für die Sicherheit einer Servosteuerung. Zusätzlich sollte der Servocontroller im Fehlerfall eines Hardware-Endschalters dafür sorgen, dass entweder die Leistungsendstufen sofort stromlos geschaltet oder die Ströme in die aktive Richtung der Hardware-Endschalter gesperrt werden.

Im ersten Lösungsweg können Sie die Achsen bei einem Hardware-Endschalter-Fehler nicht mehr bewegen. Das Ausfahren aus einem Hardware-Endschalter heraus ist somit ohne weiteres nicht mehr möglich.
Aus diesem Grund haben wir die Treiberfunktion 48 eingeführt (siehe Kapitel 3.3.48).
Beim Aufruf dieser Funktion können Sie an dem Pin 43 des RIBBON-Steckers ein HIGH- oder LOW-Signal ausgeben. Im aktiven Zustand der Hardware-Endschalter sollen Sie dieses Signals benutzen, um den Sicherheitskreis der Servosteuerung zu überbrücken. Nach dem Überbrücken ist das Einschalten der Leistungsendstufen wieder möglich. Das Ausfahren der Achsen ist kein Problem mehr. Beachten Sie, dass Sie den Sicherheitskreis der Servosteuerung nur so lange wie notwendig überbrücken. Sonst ist die Sicherheit der Servosteuerung nicht mehr gewährleistet (siehe Kapitel 2.5).

Der zweite Lösungsweg ist nur dann möglich, wenn die Leistungsendstufen mitmachen. Solche Leistungsendstufen haben zwei Eingänge, einen für die positive und einen für die negative Richtung. Hardware-Endschalter werden direkt mit diesen Eingängen verbunden. Im Fehlerfall wird der entsprechende Eingang aktiviert und der Strom in diese Richtung sofort gesperrt.
Das Herausfahren der Achsen aus den Hardware-Endschaltern ist weiterhin möglich, weil nur der Strom in die aktive Richtung gesperrt ist.

3.2.6.3 Software-Endschalter

Die Hardware-Endschalter, wie sie im Kapitel 3.2.6.2 erwähnt sind, begrenzen "hardwaremäßig" den Arbeitsraum Ihrer Anlage. Mit den Software-Endschaltern bietet unser Treiber Ihnen eine weitere Möglichkeit, um den Arbeitsraum Ihrer Anlage zu definieren.
Durch entsprechende Funktionsaufrufe können Sie die Benutzung der Software-Endschalter zulassen, sperren oder die Positionen der Software-Endschalter variieren (siehe Kapitel 3.3.14 und 3.3.15). Diese Möglichkeit gestattet es Ihnen, den Arbeitsraum Ihrer Anlage besonders flexibel zu begrenzen. Wenn die Software-Endschalter aktiviert sind, wird eine Bewegung sofort unterbrochen, falls die Anlage versucht, diese von Ihnen selbst gesetzte Grenze zu überschreiten.



X-NegEnd, X-PosEnd: Positionen der Software-Endschalter in der X-Achse
Y-NegEnd, Y-PosEnd: Positionen der Software-Endschalter in der Y-Achse

Bild 3.13: Ausführbarkeit einer Bewegung im Fall der Benutzung von Software-Endschaltern

Im Unterschied zu den Fehlern der Hardware-Endschalter geht ein eventuell noch vorhandener Rest des unterbrochenen Bewegungssegments nicht verloren. Wenn Sie jetzt die Software-Endschalter deaktivieren, oder deren Positionen neu setzen, wird der Segmentrest automatisch abgefahren. Ein weiterer Unterschied zwischen einem Hardware-Endschalter-Fehler und einem Software-Endschalter-Fehler besteht darin, dass ein neues Bewegungssegment während eines Software-Endschalter-Fehlers weiterhin vom Treiber akzeptiert wird, falls ein Segmentrest nicht mehr vorhanden ist.

Beachten Sie, dass eine Kontrolle des Software-Endschalters nur während des Abfahrens eines Segments möglich ist, d. h. der Treiber startet zuerst die Bewegung. Sofort danach wird die Ausführbarkeit der Bewegung kontrolliert. Im negativen Fall wird die Bewegung unterbrochen und Sie haben wieder den Segmentrest am Hals.
Um den Rest des Segments in diesem Fall los zu werden, ohne dass die Anlage bis zum Segmentende fahren muss, müssen Sie die Reset-Funktion aufrufen (siehe Kapitel 3.3.2). Der Rest eines Segments wird nur dann vom Treiber nicht bemerkt, wenn er sich im Teach-In-Modus befindet und wenn Sie die Bewegung durch die Stoppfunktion löschen (siehe Kapitel 3.3.10).

Genau wie bei den Hardware-Endschaltern werden die Software-Endschalter im Teach-In-Modus weiterhin vom Treiber überwacht. Das Bild 3.13 soll Ihnen zeigen, was wir unter der Ausführbarkeit einer Bewegung im Fall der Software-Endschalter verstehen.
Die Software-Endschalter bilden einen Arbeitsraum für das Werkzeug. Außerhalb dieses Raumes haben wir dann den Sperraum. Falls es momentan keine Software-Endschalter-Fehler gibt, ist jede Bewegung ausführbar (siehe Punkt P1). Wenn dies nicht der Fall ist, werden nur solche Bewegungen erlaubt, die versuchen, das Werkzeug zurück in den Arbeitsraum zu bringen (siehe Punkte P2 und P3). Die Ausführbarkeit einer Bewegung hängt damit von der momentanen Werkzeugposition ab.

3.2.6.4 Nachlauffehler und Ausschalten von Achsreglern

Eine weitere Sicherheit für Ihre Anlage bekommen Sie dadurch, dass der Treiber den sogenannten Nachlauffehler (Schleppfehler) überwacht. Jede Achse einer Anlage hat naturgemäß immer eine Totzeit ungleich Null. Die Totzeit stellt nichts anders als die Verzögerungszeit der Anlage dar.
Aufgrund dieser Totzeit können das Soll-Geschwindigkeitssignal und das Ist-Geschwindigkeitssignal niemals zeitgleich laufen. Es führt dann dazu, dass eine Abweichung ungleich Null zwischen der Soll- und der Ist-Position während einer Bewegung immer vorhanden ist (siehe Bild 3.14). Diese Abweichung ist der Nachlauffehler.

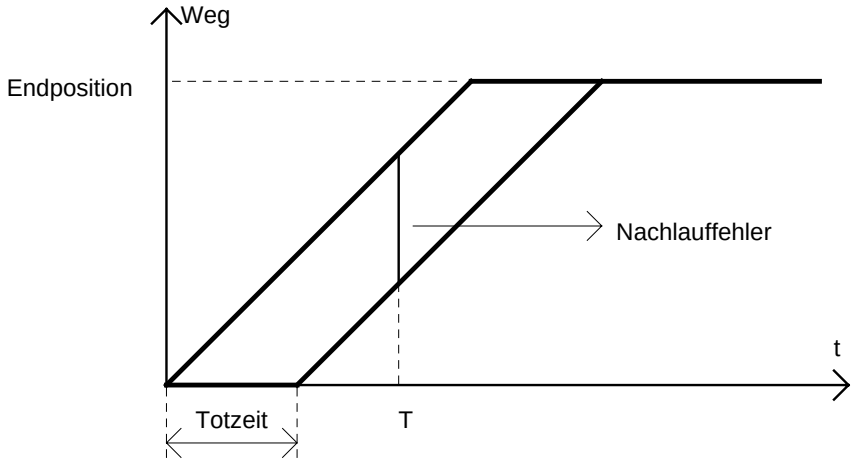


Bild 3.14: Nachlauffehler während einer Bewegung

Die obere Grenze des durch die Totzeit entstehenden Nachlauffehlers kann man in Abhängigkeit von der Geschwindigkeit wie im Folgenden grob abschätzen:

Nachlauffehler = Totzeit * Geschwindigkeit * Risikofaktor.

Die Totzeit der einzelnen Achsen Ihrer Anlage können Sie mit Hilfe des Einstellprogramms PAREIN.EXE mühelos ermitteln.

Der Risikofaktor ist deshalb wichtig, weil jede Anlage einer gewissen Lastschwankung während der Bearbeitung unterliegt. Den Risikofaktor können Sie frei definieren. Ein Wert von 200 % ist vernünftig.
Während einer Bewegung kontrolliert der Treiber ständig den Nachlauffehler. Falls der Nachlauffehler die nach der obigen Formel berechnete Obergrenze überschreitet, ist einer der folgenden Fehler die Ursache dafür:

- Die Soll-Geschwindigkeiten wurden zu hoch gesetzt. Die Achsen können diese Werte gar nicht erreichen.
- Die Encoder oder die Encoder-Signalleitungen sind nicht in Ordnung.
- Die Leistungsendstufen sind defekt oder es wurde einfach vergessen, sie einzuschalten. Diese Nachlässigkeit tritt öfter ein, als man denkt.
- Der Schlitten fährt auf einen Anschlag oder auf ein unüberwindbares Hindernis.

Wenn der Treiber einen unzulässigen Nachlauffehler feststellt, wird die Bewegung sofort unterbrochen. Ein eventuell vorhandener Segmentrest geht verloren. Intern wird ein Fehlerflag gesetzt, das Sie nur durch die Reset-Funktion zurücksetzen können (siehe Kapitel 3.3.2).
Solange dieses Fehlerflag gesetzt ist, können Sie die meisten Treiber-funktionen nicht aufrufen. Ein neues Bewegungssegment wird nicht mehr akzeptiert, wenn sich der Treiber nicht im Teach-In-Modus befindet.
In diesem Modus wird der Nachlauffehler weiterhin überwacht. Falls der Fehler auftritt, wird die Bewegung auch sofort unterbrochen. Das Fehlerflag wird dabei aber nicht gesetzt. Die Überwachung des Nachlauffehlers ist deshalb wichtig, weil ein zu großer Nachlauffehler, d. h. eine zu große Regelabweichung, zu einem hohen Motorstrom führt. Unter Umständen können Schäden beim Motor und bei der Mechanik entstehen.

Bei einem Regler mit Integrationsanteil kann der Nachlauffehler mit der Zeit zu einem hohen Motorstrom führen, obwohl der Nachlauffehler vom Treiber bemerkt ist. Um das zu vermeiden, bietet der Treiber Ihnen optional die Möglichkeit, den Regler beim Nachlauffehler auszuschalten.
Wenn der Regler einmal ausgeschaltet ist, können selbst große Regel-abweichungen nicht mehr zu einem hohen Motorstrom führen. Mit Hilfe des Konfigurationsprogramms PARKON.EXE können Sie selbst festlegen, ob der Treiber beim Nachlauffehler den Regler automatisch ausschaltet oder nicht. Bei der nächsten Bewegung wird der Regler automatisch wieder eingeschaltet. Sie als Anwender werden von dem Ein- bzw. Ausschalten des Reglers absolut nichts merken und brauchen darum auch um nichts zu kümmern.

Die Möglichkeit des Treibers zum Ausschalten des Reglers sollten Sie unbedingt benutzen, wenn Ihre Leistungsendstufen keine hardwaremäßigen Maßnahmen zur Strombegrenzung haben. Wenn die hardwaremäßige Strombegrenzung möglich ist, ist die Benutzung optional.

3.2.7 Die Referenzfahrt

Um den Maschinen-Nullpunkt zu definieren, muss die Referenzfahrt ausgeführt sein. Das ist eigentlich üblich bei jeder Anlage, die keine absoluten Positionsmesssysteme benutzen. Absolutmesssysteme sind normalerweise sehr teuer und fordern ein kompliziertes Interface. Sie arbeiten dafür aber sehr zuverlässig und haben keine Probleme mit Störungen. Unsere PC-Einsteckkarte kann nicht mit absoluten Positionsmesssystemen arbeiten. Deswegen ist die Referenzfahrt ein muss. Wir haben zwei Modi der Referenzfahrt berücksichtigt. Es sind der Standard-Modus und der Nicht-Standard-Modus.

Im Standard-Modus fährt die Achse zuerst mit einer konstanten Geschwindigkeit zum Referenzschalter, bis dieser Schalter betätigt wird. Mit einer wesentlich kleineren Geschwindigkeit fährt die Achse wieder aus dem Schalter heraus. Der Punkt, in dem der Schalter seinen Pegel ändert, wird üblicherweise als der Referenzpunkt bzw. als der Maschinen-Nullpunkt definiert.

Mit dem Nicht-Standard-Modus wollen wir Drehachsen berücksichtigen, die endlos drehen können. Bei solchen Drehachsen werden normalerweise Magnetschalter als Referenzschalter benutzt. Man kann hier ohne weiteres den Standard-Modus benutzen. Die Symmetrie der Achse nach der Referenzfahrt wird aber nicht gewährleistet (siehe Bild 3.15).
Beim Nicht-Standard-Modus werden die Positionen der beiden Umschaltpunkte des Magnetschalters ermittelt. Der Mittelpunkt dieser beiden Umschaltpunkte wird als Referenzpunkt bzw. als Maschinen-Nullpunkt definiert. Somit wird die Symmetrie der Achse nach der Referenzfahrt gewährleistet (siehe Bild 3.15).

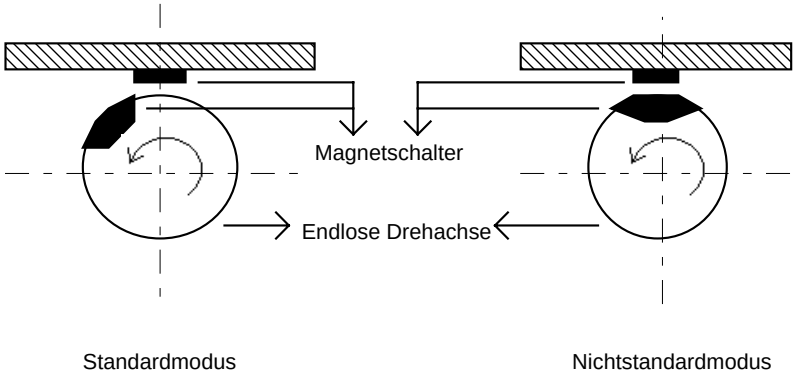


Bild 3.15: Modi der Referenzfahrt bei Drehachsen mit Magnetschaltern

Die Hardware-Endschalter, die zur Begrenzung des physikalischen Arbeitsbereichs der Anlage notwendig sind, werden in vielen Fällen aus Spargründen gleichzeitig als Referenzschalter benutzt. Softwareseitig haben wir hier schon vorgesehen, dass Sie bei jeder Achse einen separaten Schalter oder einen der beiden Hardware-Endschalter als Referenzschalter konfigurieren können (siehe das Handbuch für das Konfigurationsprogramm PARKON.EXE).

Während der Referenzfahrt wird der Hardware-Endschalter betätigt, der als Referenzschalter benutzt wird. Durch die hardwaremäßige Gegebenheit Ihrer Servosteuerung können die Leistungsendstufen beim Betätigen des Hardware-Endschalters ausgeschaltet werden, wie wir im Kapitel 3.2.6.2 schon erwähnt haben. In diesem Fall sind Hardwaremaßnahmen notwendig, um das Ausschalten der Leistungsendstufen zu vermeiden.
Eine der Möglichkeiten ist die Benutzung der Treiberfunktion 48 (siehe Kapitel 3.3.48).

Vor der Referenzfahrt müssen Sie diese Funktion aufrufen, um den Sicherheitskreis des Servocontrollers zu überbrücken. Nach der Referenzfahrt ist die Funktion noch einmal zu benutzen, um den Sicherheitskreis des Servocontrollers wieder in Gang zu bringen. Im Fall, dass die Leistungsend-stufen nicht ausgeschaltet sind sondern nur der Strom in die aktive Richtung gesperrt wird, brauchen Sie bei der Referenzfahrt gar nichts weiter zu machen.

In der Praxis ist es üblich, dass das Encoder-Indexsignal bei der Referenzfahrt benutzt wird, um einen hoch genauen Maschinen-Nullpunkt definieren zu können. Die Referenzfahrt mit dem Indexsignal ist in der Anfangphase ähnlich wie die ohne Indexsignal. Der Unterschied besteht in der Endphase. Beim Herausfahren aus dem Referenzschalter wird nicht mehr der Punkt, in dem der Referenzschalter seinen Pegel ändert, sondern der darauf folgende Punkt, in dem das Encoder-Indexsignal ankommt, als der Maschinen-Nullpunkt definiert. Die Position des Indexsignals, das der Encoder einmal pro Umdrehung liefert, ist unabhängig von der Genauigkeit des Referenzschalters. Auf diese Art und Weise wird eine sehr hohe Genauigkeit des Maschinen-Nullpunkts erreicht.

Unsere Treibersoftware benutzt diese Möglichkeit der Referenzfahrt standardmäßig nicht. Um die Referenzfahrt mit dem Encoder-Indexsignal ausführen zu können, müssen Sie den Treiber mit dem Optionsschalter /IR starten:

[Laufwerk:\][Pfad] ISELDRV.EXE *.INI [/IR].

Beim Aufruf der Funktion 6 des Treibers wird die Referenzfahrt mit dem Encoder-Indexsignal automatisch ausgeführt, falls die Achse im Referenzfahrt_Standard-Modus steht. Der Optionsschalter /IR hat keinen Einfluss auf die Achsen, die im Referenzfahrt_Nicht-Standard-Modus stehen. Es gibt keinen Unterschied beim Aufruf der Funktion 6, egal ob Sie die Referenzfahrt mit oder ohne Indexsignal ausführen wollen. Beachten Sie, dass Sie hardwaremäßig die Indexsignale der Encoder zu den entsprechenden Pins auf dem RIBBON-Steckverbinder der Einsteckkarte zurückführen müssen (siehe Kapitel 2.9). Sonst bekommen Sie beim Aufruf der Funktion 6 den Fehlercode 37 (siehe Kapitel 3.2.20). Außerdem müssen Sie beachten, dass unsere Einsteckkarte im Lieferzustand für das direkte Anschließen von Encoder-Indexsignalen nicht geeignet ist. Eine kleine Änderung des Widerstandnetzwerks ist notwendig. In diesem Fall sollten Sie mit unseren Service-Technikern sprechen.

3.2.8 Geschwindigkeitsabhängige Peripheriesteuerung

Bei vielen Bearbeitungsaufgaben besteht die Notwendigkeit, eine oder mehrere Peripherien geschwindigkeitsabhängig zu steuern, z. B. bei der Bearbeitung mit LASER ist es oft notwendig, die LASER-Intensität geschwindigkeits-abhängig zu steuern. Oder beim Kleberauftrag erreicht man dann eine wesentlich größere Bearbeitungsgeschwindigkeit, falls es möglich ist, die Öffnung des Düsenventils geschwindigkeitsabhängig zu steuern. Das sind zwei von vielen Anwendungsfällen, wo man wirklich große Vorteile durch die geschwindigkeitsabhängige Peripherie-Steuerung erreichen kann.
Während der Bewegung besteht die Möglichkeit, die Ist-Bahngeschwindigkeit der Anlage durch den Aufruf der entsprechenden Treiberfunktion abzufragen. Diese Informationen können Sie dann in Ihrem Anwenderprogramm für eine geschwindigkeitsabhängige Steuerung der Peripherien benutzen.
Solange die durch die Trägheit verursachte Nachlaufzeit vernachlässigbar klein ist, kann man ohne weiteres diese Methode benutzen. Falls dies nicht der Fall ist, kann von einer geschwindigkeitsabhängigen Steuerung keine Rede sein. Im Bild 3.16 wird diese Tatsache verdeutlicht.

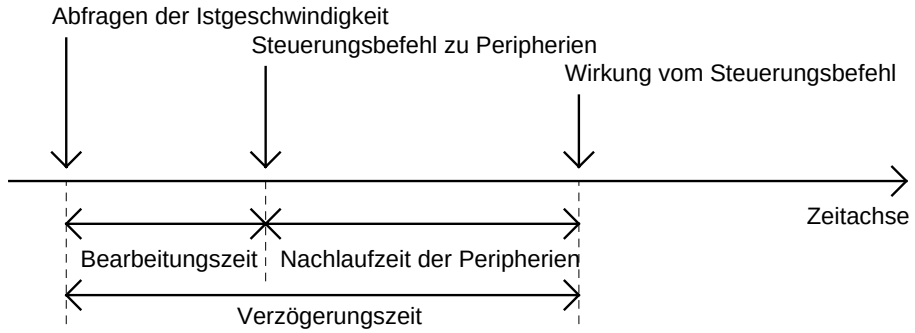


Bild 3.16: Verzögerung der Wirkung von Peripherie-Steuerungsbefehlen

Die Bearbeitungszeit, die ein Anwenderprogramm braucht, um aus der Ist-Geschwindigkeit den entsprechenden Steuerungsbefehl für die Peripherie zu erzeugen, ist in den meisten Fällen sehr klein.
Falls die Verzögerungszeit, bestehend aus dieser Bearbeitungszeit und der Nachlaufzeit der Peripherie, nicht mehr vernachlässigbar klein ist, hat diese Methode keinen Sinn.

Neben der oben genannten Methode haben Sie mit dem Treiber noch eine zweite Möglichkeit, um eine geschwindigkeitsabhängige Steuerung zu realisieren. Diese verbesserte Methode basiert auf dem Gedanken, dass auch die Bearbeitungsanlage eine Trägheit besitzt. Jede Bearbeitungsanlage braucht eine gewisse Nachlaufzeit, bis die Ist-Geschwindigkeit ihren Soll-Wert erreicht hat. Wenn Sie anstelle der Ist-Geschwindigkeit ihren Soll-Wert für die geschwindigkeitsabhängige Steuerung benutzen, kompensiert die Nachlaufzeit der Anlage ganz oder teilweise die Nachlaufzeit der Peripherie (s. Bild 3.17).

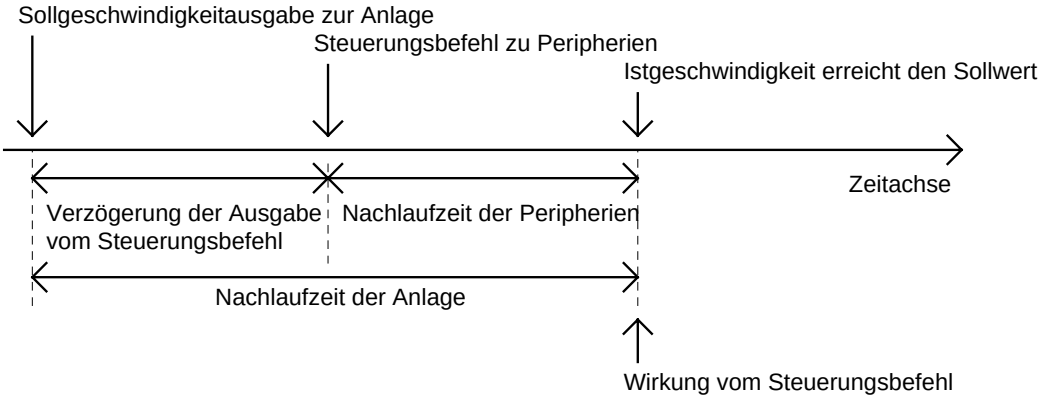


Bild 3.17: Kompensation der Peripherienachlaufzeit durch die Anlagen-Nachlaufzeit

Falls die Nachlaufzeit der Anlage größer oder gleich die der Peripherien ist, kann der Befehl zur Steuerung der Peripherien künstlich verzögert ausgegeben werden. Diese künstliche Verzögerungszeit kann so variiert werden, dass der Steuerungsbefehl seine Wirkung genau zu dem Zeitpunkt zeigt, in dem die Ist-Geschwindigkeit ihren Soll-Wert erreicht. Damit ist eine exakte geschwindigkeitsabhängige Steuerung erreicht.
Aber wie Sie selbst sehen können, funktioniert diese Methode nur in solchen Fällen, in denen die Nachlaufzeit der Anlage größer oder gleich die der Peripherien ist. Diese Methode ist eine Verbesserung im Vergleich zur ersten Methode. Auf keinen Fall ist sie aber die Lösung für alle Fälle.

Diese zweite Methode ist in unserem Treiber implementiert und unterstützt. Als Anwender können Sie diese Möglichkeit nutzen, indem Sie einen Ausgabeport mit Hilfe des Konfigurationsprogramms PARKON.EXE definieren.
An diesem Ausgabeport gibt der Treiber regelmäßig während der Bewegung einen Wert aus, der von der Soll-Geschwindigkeit abhängig ist. Dieser Ausgabewert ist für Sie eine Information über die Anlagengeschwindigkeit.
Mit Hilfe dieser Information können Sie dann eine geschwindigkeitsabhängige Peripheriesteuerung realisieren. Sie können sogar den Ausgabebereich durch die Angabe vom Maximal- und Minimalwert selbst definieren.
Vor jeder Bewegung berechnet der Treiber den Geschwindigkeitsfaktor aus diesen Grenzwerten und der zu bewegenden Geschwindigkeit. Der Geschwindigkeitsfaktor lautet:

$$\text{Geschwindigkeitsfaktor} = \frac{(\text{Maximalwert} - \text{Minimalwert})}{1{,}4 * \text{Bewegungsgeschwindigkeit}}$$

Während des Segmentfahrens ist die Bewegungsgeschwindigkeit gleich der Segment-Geschwindigkeit. Während des Bahnfahrens ist sie gleich der Bahngeschwindigkeit. Der Faktor 1,4 im Nenner soll den Override von 140 % berücksichtigen. Der Ausgabewert lässt sich so berechnen:

$$\text{Ausgabewert} = \text{Aktuellsollgeschwindigkeit} * \text{Geschwindigkeitsfaktor} + \text{Minimalwert}$$

Falls sich die Anlage nicht bewegt, wird der Minimalwert ausgegeben. Falls sich die Anlage mit der maximalen Geschwindigkeit bewegt, wird der Maximalwert ausgegeben.
Die Verzögerungszeit, mit der man eine exakte geschwindigkeitsabhängige Steuerung realisieren kann (siehe Bild 3.17), können Sie auch über das Konfigurationsprogramm frei definieren. Die Benutzung dieser geschwindigkeitsabhängigen Ausgabe muss ausdrücklich wiederum über das Konfigurationsprogramm angegeben werden. Sonst erfolgt die Ausgabe nicht. Nachdem der Treiber schon geladen ist, können Sie diese Ausgabe sperren oder freilassen, wie es nötig ist (siehe Kapitel 3.3.43).

3.2.9 Das Datenformat, die Einheiten Mikrometer, Bogensekunde und die Sonderzahl NO_MOVE_VALUE

Ein Anwenderprogramm und der Treiber müssen ständig miteinander kommunizieren und die Daten austauschen. Natürlich müssen die Daten in einem solchen Format dargestellt werden, das sowohl das Anwenderprogramm als auch der Treiber ihre Daten einfach und schnell in ein solches Format umwandeln sowie die Daten in diesem Format schnell und einfach lesen und interpretieren können.

Aus dieser Forderung ergibt sich als Konsequenz: Die Austauschdaten müssen ganze Zahlen in einer Zweier-Komplementär-Darstellung sein, weil dieses Datenformat von jeder Hochsprache automatisch unterstützt wird (siehe Kapitel 3.2.13). Als Anwender brauchen Sie sich über die Datentransformation keine Gedanken zu machen.
Es gibt zwei Bewegungsmöglichkeiten.
Einmal ist es die Linearbewegung mit Längeneinheit wie z. B. Meter, Millimeter, zum zweiten ist es die Drehbewegung mit Winkeleinheit wie z. B. Grad. Auf der einen Seite ist es sehr angenehm für die Anwender, falls sie nur mit den Ihnen bekannten Einheiten arbeiten können. Auf der anderen Seite müssen die Einheiten so gewählt sein, dass die Bewegungsparameter als ganze Zahlen mit einer ausreichenden Genauigkeit darstellbar sind. Aus diesen Gründen haben wir für die Linearbewegung die Einheit Mikrometer gewählt. Es gilt:

1 Millimeter [mm] = 1 000 Mikrometer [μm]

In fast allen Anwendungsfällen ist die Genauigkeit von einem Mikrometer völlig ausreichend. Für die Drehbewegung ist die Einheit Bogensekunde zuständig. Es gilt:

1 Umdrehung	= 360	Grad [°]
1 Grad [°]	= 60	Bogenminuten [']
1 Bogenminute [']	= 60	Bogensekunden ["]

Daraus folgt:

1 Umdrehung	= 21 600	Bogenminute [']
1 Umdrehung	= 1 296 000	Bogensekunden ["]
1 Grad [°]	= 3 600	Bogensekunden ["]

In Ihrem Anwenderprogramm müssen Sie alle Bewegungsdaten (Position, Geschwindigkeit und Beschleunigung) zuerst von der konventionellen Einheit wie z. B. Millimeter oder Grad in die Einheit Mikrometer bzw. Bogensekunde umwandeln, bevor Sie die Daten dem Treiber übergeben.
Die vom Treiber kommenden Bewegungsdaten müssen Sie natürlich wieder in die konventionellen Einheiten zurück transformieren. Diese ganze Hin- und Hertransformationen können Sie aber sehr schnell und einfach erledigen, indem Sie für jede Achse einen sogenannten Umsetzungsfaktor einführen.
Die ganze Transformation für jede Achse ist dann nur noch eine Multiplikation oder eine Division mit diesem Umsetzungsfaktor.
Um Klarheit zu schaffen, berechnen wir im Folgenden zwei Beispiele für Sie.

Beispiel 1: Berechnung des Umsetzungsfaktors für eine Linearachse

Wenn das Anwenderprogramm die Einheit Millimeter [mm] benutzt, haben wir dann den Umsetzungsfaktor = 1 000, weil ein Millimeter gleich 1 000 Mikrometer [μm] sind.
Wenn die Achse einen Weg von 500 mm bewegt werden soll, müssen Sie dem Treiber folgenden Weg übergeben:

Bewegungsweg * Umsetzungsfaktor = 500 * 1 000 = 500 000 μm

Wenn das Anwenderprogramm bei der Ist-Positionsabfrage einen Wert von 150 000 μm vom Treiber bekommt, ist der zurückgelegte Weg gleich:

Bewegungsweg in μm / Umsetzungsfaktor = 150 000 / 1 000 = 150 mm.

Beispiel 2: Berechnung des Umsetzungsfaktors für eine Drehachse

Wenn das Anwenderprogramm die Einheit Grad [°] benutzt, haben wir den Umsetzungsfaktor = 3 600, weil ein Grad gleich 3 600 Bogensekunden ["] hat.
Wenn die Achse um 10 Grad gedreht werden soll, müssen Sie dem Treiber folgenden Winkel übergeben:

Drehwinkel * Umsetzungsfaktor = 10 * 3 600 = 36 000"

Wenn das Anwenderprogramm bei der Ist-Positionsabfrage einen Wert von 72 000 vom Treiber bekommt, ist der zurückgelegte Drehwinkel gleich:

Drehwinkel / Umsetzungsfaktor = 72 000 / 3 600 = 20°

Sie haben sicherlich auch festgestellt, dass die Anzahl der Encoder-Linien und der Getriebefaktor in den obigen Berechnungen nicht vorkommen. Der Grund: Der Treiber hat intern aufgrund der von Ihnen in die Konfigurationsdatei eingegeben Parameter dies schon für Sie berücksichtigt.

Der größte Vorteil bei der Benutzung der Einheiten Mikrometer und Bogensekunde besteht in der Trennung zwischen dem Anwenderprogramm und der Mechanik. Das Anwenderprogramm braucht nicht zu beachten, welchen Encoder bzw. welchen Getriebefaktor die jeweilige Achse der Anlage hat.
Um die Achsen zu bewegen, müssen Sie durch den Aufruf einer Bewegungsfunktion dem Treiber die entsprechenden Achskoordinaten übergeben. Dabei hat eine Achskoordinate wie die folgende eine besondere Bedeutung:

NO_MOVE_VALUE = 3FFFFFFFh = 1073741823

Auf der einen Seite ist sie der größte Achskoordinatenwert, den Sie dem Treiber übergeben können, weil der Achscontroller (LM628 bzw. LM629) einen größeren Wert nicht bearbeiten kann.
Auf der anderen Seite erreichen Sie mit Ihrer Anlage nie eine so große Achskoordinate. Deswegen ordnen wir diesem Wert eine besondere Bedeutung zu.
Bei einer Relativkoordinate interpretiert der Treiber diesen Wert als eine Null.
Bei einer Absolutkoordinate wird die momentane Absolutkoordinate anstelle dieses Wertes benutzt, d. h. eine Achskoordinate von dieser Zahl bewirkt keine Bewegung der Achse.
Das Gleiche gilt auch für Koordinaten des Mittelpunktes eines Kreissegments. Deswegen wird diese Zahl NO_MOVE_VALUE genannt.
In solchen Fällen, in denen Sie in Ihrem Anwenderprogramm die Achskoordinaten nicht ständig mitschleppen wollen, ist die Benutzung dieser Zahl sehr vorteilhaft.

3.2.10 Die vielen Geschwindigkeiten und wie kann man sie berechnen?

Sie werden in diesem Handbuch noch auf verschiedene Begriffe die Geschwindigkeit betreffend stoßen. Um einer möglichen Verwirrung vorzubeugen, wollen wir Ihnen diese Begriffe hier kurz erläutern.

Achsgeschwindigkeit:
Das ist die Geschwindigkeit der *einzelnen* Achse Ihrer Anlage

Referenzgeschwindigkeiten:
Um den Maschinen-Referenzpunkt zu definieren, müssen Sie für jede Achse Ihrer Anlage eine sogenannte Referenzfahrt durchführen (siehe Kapitel 3.2.7 und 3.3.6).
Zuerst wird die Achse in Richtung des Referenzschalters gefahren. Diese Fahrt dauert solange, bis der Referenzschalter betätigt ist. Danach fährt die Achse wieder zurück. Die Rückfahrt dauert solange, bis der Referenzschalter nicht mehr betätigt ist.
Der Punkt, in dem sich die Achse nach der Rückfahrt befindet, ist der Referenzpunkt der Achse. Um eine schnelle Referenzfahrt durchführen zu

können und eine hohe Wiederholgenauigkeit des Referenzpunktes zu bekommen, wählt man die Geschwindigkeit für der Hinfahrt zum Referenzschalter wesentlich größer als die für die Rückfahrt vom Schalter. Diese beiden Geschwindigkeiten bezeichnet man als Referenzgeschwindigkeiten. Sie können beide über das Konfigurationsprogramm für jede Achse getrennt einstellen. Die Referenzgeschwindigkeiten gehören zur Kategorie der Achsgeschwindigkeiten.

Segment-Geschwindigkeit:
Diese Geschwindigkeit ist die der vektoriellen Bearbeitung. Wenn Sie eine der Funktionen 24, 25, 28 und 29 (siehe Kapitel 3.3.24, 3.3.25, 3.3.28 und 3.3.29) aufrufen, benutzt der Treiber diese Geschwindigkeit für die interne Interpolation. Aber diese Geschwindigkeit ist nicht immer gleich der Soll-Geschwindigkeit für die Bearbeitung, weil der Treiber intern die Soll-Geschwindigkeit gleich dem Produkt aus der Segment-Geschwindigkeit und dem sogenannten Geschwindigkeitsfaktor setzt. Damit haben Sie die Möglichkeit, die Bearbeitungsgeschwindigkeit durch die Änderung dieses Geschwindigkeitsfaktors zu variieren (siehe Kapitel 3.3.21).

Beachten Sie, dass sich die Segment-Geschwindigkeit immer auf das Werkzeug bezieht, d. h. diese Geschwindigkeit setzt sich aus den Achsgeschwindigkeiten zusammen. Wie man sie berechnet, können Sie weiter unten sehen.
Der Standardwert der Segment-Geschwindigkeit lässt sich über das Konfigurationsprogramm einstellen. Über die Funktion 20 (siehe Kapitel 3.3.20) können Sie dann neue Werte definieren.

Bahngeschwindigkeit:
Diese Geschwindigkeit ist die der Bahnbearbeitung. Wenn Sie die Funktion 33 (siehe Kapitel 3.3.33) aufrufen, benutzt der Treiber diese Geschwindigkeit für die interne Interpolation. Ähnlich wie bei der vektoriellen Bearbeitung ist die Soll-Geschwindigkeit für die Bahnbearbeitung gleich dem Produkt aus der Bahngeschwindigkeit und dem Geschwindigkeitsfaktor.
Damit haben Sie hier auch die Möglichkeit, die Bearbeitungsgeschwindigkeit durch die Änderung dieses Geschwindigkeitsfaktors zu variieren (siehe Kapitel 3.3.21).
Der Standardwert der Bahngeschwindigkeit lässt sich über das Konfigurationsprogramm einstellen. Über die Funktion 32 (siehe Kapitel 3.3.32) können Sie neue Werte definieren. Die Bahngeschwindigkeit bezieht sich auf das Werkzeug, d. h. diese Geschwindigkeit lässt sich aus den Achsgeschwindigkeiten berechnen. Wie man sie berechnen kann, können Sie unten sehen.

Eilgeschwindigkeit:
Diese Geschwindigkeit ist für den Eilgang gedacht. Ein Eilgang ist für ein schnelles Positionieren, wie z. B. bei einem Werkzeugwechsel interessant. Wenn Sie eine der Funktionen 26 oder 27 (siehe Kapitel 3.3.26 und 3.3.27) aufrufen, benutzt der Treiber diese Geschwindigkeit für die Interpolation. Ähnlich wie bei der Segment-Geschwindigkeit lässt sich die Soll-Geschwindigkeit beim Abfahren der Eilsegmente durch die Variation des Geschwindigkeitsfaktors ändern, weil sie das Produkt aus Eilgeschwindigkeit und Geschwindigkeitsfaktor ist.
Wie sich diese Eilgeschwindigkeit aus den einzelnen Achsgeschwindigkeiten zusammensetzt, werden wir Ihnen noch zeigen.
Neue Werte der Eilgeschwindigkeit können Sie durch die Funktion 23 einstellen (siehe Kapitel 3.3.23). Der Standardwert lässt sich wieder beim Konfigurationsprogramm einstellen.

Teach-In-Geschwindigkeit:
Diese Geschwindigkeit ist für die Teach-In-Bewegung gedacht.
Beim Aufruf einer der Funktionen 24 oder 25 (siehe Kapitel 3.3.24 und 3.3.25) wird diese Geschwindigkeit für die Interpolation nur benutzt, wenn Sie vorher den Teach-In-Modus durch die Funktion 9 (siehe Kapitel 3.3.9) eingeschaltet haben.
Beim Abfahren dieser Linearsegmente hat der Geschwindigkeitsfaktor aber keinen Einfluss darauf, ob sich die Anlage schneller oder langsamer bewegen soll. Der Treiber benutzt direkt diese Teach-In-Geschwindigkeit für die Interpolation, d. h. sie ist immer gleich der Soll-Geschwindigkeit.
Der Standardwert und die neuen Werte der Teach-In-Geschwindigkeit lassen sich durch das Konfigurationsprogramm und die Funktion 22 verändern (siehe Kapitel 3.3.22).
Weiter unten werden wir Ihnen noch zeigen, wie der Treiber die Teach-In-Geschwindigkeit aus den Achsgeschwindigkeiten berechnet.

Werkzeug-Soll-Geschwindigkeit oder Werkzeug-Geschwindigkeit:
Das ist die Geschwindigkeit, mit der sich das Werkzeug Ihrer Anlage bewegen soll. Es ist offensichtlich, dass sich diese Geschwindigkeit aus den einzelnen Achsgeschwindigkeiten zusammensetzt. Diese wiederum hängen davon ab, ob der Teach-In-Modus momentan aktiv ist, ob der Treiber einen Eilgang realisieren will oder ob es sich um einen Bearbeitungsvorgang handelt. Der Treiber entscheidet dann, ob er die Werkzeuggeschwindigkeit gleich der Teach-In-Geschwindigkeit oder gleich dem Produkt zwischen der Eilgeschwindigkeit und dem Geschwindigkeitsfaktor oder gleich dem Produkt zwischen der Segment- bzw. Bahngeschwindigkeit und dem Geschwindigkeitsfaktor setzt. Während der Referenzfahrt ist die Werkzeuggeschwindigkeit gleich der Referenzgeschwindigkeit.

Werkzeug-Ist-Geschwindigkeit oder Ist-Geschwindigkeit:
Das ist die Geschwindigkeit, mit der sich das Werkzeug tatsächlich bewegt. Es ist natürlich wünschenswert, dass Soll- und Ist-Geschwindigkeit immer

gleich sind. Aber aufgrund der Störungen in einer realen Anlage sind solche Träume nie ganz realisierbar.
Im Allgemeinen schwankt der Ist-Wert um den Soll-Wert; der Mittelwert des Ist-Wertes tendiert gegen den Soll-Wert. Über die Funktion 19 besteht für Sie die Möglichkeit, die momentane Ist-Geschwindigkeit abzufragen (siehe Kapitel 3.3.19).

Die Segment- , die Bahn- , die Eil- , die Teach-In- sowie die Ist-Geschwindigkeit gehören zur Kategorie der Werkzeug-Geschwindigkeit. Sie alle werden auf die gleiche Art und Weise aus den einzelnen Achs-Geschwindigkeiten berechnet.
Im Folgenden werden wir Ihnen zeigen, wie man sie in Abhängigkeit von dem zu fahrenden Bewegungssegment berechnen kann.

Linearsegment:
Die Werkzeug-Geschwindigkeit lässt sich in Abhängigkeit von der Anlagenstruktur und vom aktuellen Bewegungssegment berechnen. Auf der Seite der Anlagenstruktur haben wir die X_TTT-Struktur mit der X-Achse als Hauptachse, die XY_TTT-Struktur mit X- Achse und Y-Achse als Hauptachsen, die XYZ_TTT-Struktur mit den 3 Hauptachsen X, Y , Z und die Kein_TTT-Struktur (siehe Kapitel 1.2 und Handbuch für das Konfigurationsprogramm PARKON.EXE).

Alle Achsen, die keine Hauptachsen sind, nennen wir Mitschleppachsen.
Auf der Seite des Bewegungssegments haben wir das Normal- und das Mitschleppsegment. Bei einem Normalsegment ist der Bewegungsweg mindestens einer der Hauptachsen ungleich Null. Bei einem Mitschleppsegment sind die Bewegungswege aller Hauptachsen gleich Null.

Beachten Sie, dass jedes Bewegungssegment bei einer Kein_TTT-Struktur ein Mitschleppsegment ist.
Bei einem Normalsegment lässt sich die Werkzeuggeschwindigkeit aus den Geschwindigkeiten der Hauptachsen berechnen. Bei einem Mitschleppsegment ist die Werkzeuggeschwindigkeit gleich der Geschwindigkeit der ersten Mitschleppachse, bei der der Bewegungsweg ungleich Null ist. Hier werden die Mitschleppachsen in der Reihenfolge X, Y, Z, A gezählt.

Am folgenden Beispiel wollen wir das Gesagte verdeutlichen. Bei einer Anlage mit der XY_TTT-Struktur haben wir für einen Normalsegment:

$$V_{werkzeug} = \sqrt{V_x^2 + V_y^2} \quad .$$

Bei einem Mitschleppsegment haben wir:

$$V_{werkzeug} = V_z \quad ,$$

falls der Bewegungsweg der Z-Achse ungleich Null ist. Sonst haben wir:

$$V_{werkzeug} = V_a \quad ,$$

falls die A-Achse existiert.

Für ein Normalsegment entspricht die Werkzeuggeschwindigkeit wirklich der Geschwindigkeit des Werkzeugs auf der X-Achse (bei der X_TTT-Struktur) oder auf der XY-Ebene (bei der XY_TTT-Struktur) oder im Raum (bei der XYZ_TTT-Struktur).
Für ein Mitschleppsegment ist es natürlich nicht der Fall.
Um die einzelnen Achsgeschwindigkeiten aus der Werkzeuggeschwindigkeit berechnen zu können, müssen Sie zuerst die sogenannte Segmentlänge berechnen. Bei einem Normalsegment lässt sich die Segmentlänge aus den Bewegungswegen der Hauptachsen berechnen. Bei einem Mitschlepp-segment ist die Segmentlänge gleich dem Bewegungsweg der ersten Mitschleppachse, bei der der Bewegungsweg ungleich Null ist.
Hier werden die Achsen wiederum in der Reihenfolge X, Y, Z, A gezählt. Für jede Achse wird der Bewegungsfaktor berechnet. Der Bewegungsfaktor ist der Quotient aus dem Achsbewegungsweg und der Segmentlänge. Die Achs-geschwindigkeit ist gleich dem Produkt aus dem Bewegungsfaktor und der Werkzeuggeschwindigkeit. Die ganze Berechnung wird am folgenden Beispiel verdeutlicht.

Für eine Anlage mit der XY_TTT-Struktur haben wir bei einem Normalsegment:

$$Bl = \sqrt{X_{rel}^2 + Y_{rel}^2} \quad .$$

Bei einem Mitschleppsegment haben wir:

$$Bl = Z_{rel}$$

falls Z_{rel} ungleich Null ist. Sonst haben wir:

$$Bl = A_{rel}$$

falls die A-Achse existiert. Dabei ist Bl die Segmentlänge. X_{rel} , Y_{rel} , Z_{rel} und A_{rel} sind die Bewegungswege der jeweiligen Achsen.
Die Geschwindigkeit der jeweiligen Achse lässt sich so berechnen:

$$V_x = \frac{X_{rel}}{Bl} * V_{werkzeug} \quad ,$$

$$V_y = \frac{Y_{rel}}{Bl} * V_{werkzeug} \quad ,$$

$$V_z = \frac{Z_{rel}}{Bl} * V_{werkzeug} \quad \text{und}$$

$$V_a = \frac{A_{rel}}{Bl} * V_{werkzeug} \; .$$

Kreis- und Helixsegment:
Eine Kreis- bzw. eine Helixinterpolation lässt der Treiber nur zu, falls Ihre Anlage die XY_TTT- oder die XYZ_TTT-Struktur hat. Die Werkzeuggeschwindigkeit setzt sich aus den Achsgeschwindigkeiten der Achsen zusammen, die die Kreisebene bilden. Wir haben

$$V_{werkzeug} = \sqrt{V_1^2 + V_2^2} \; .$$

Dabei sind V_1 und V_2 die Achsgeschwindigkeiten der beiden Achsen, die die Kreisebene bilden. Es kann die XY- oder XZ- oder YZ-Ebene sein. $V_{Werkzeug}$ ist die tatsächliche Geschwindigkeit des Werkzeugs auf der Kreisbahn.

Aus der Werkzeuggeschwindigkeit kann man die Achsgeschwindigkeiten V_1 und V_2 leider nicht mehr so einfach berechnen wie bei einem Linearsegment, weil sie Cosinus- und Sinus-Funktionen sind. Aber für die dritte Achse und die A-Achse haben wir:

$$V_3 = \frac{Achse3_{rel}}{Bl} \quad \text{und}$$

$$V_a = \frac{A_{rel}}{Bl} \; .$$

Dabei sind Achse 3_{rel} und A_{rel} die Relativkoordinaten der dritten Achse und der A-Achse im Segment-Endpunkt.
Sowohl bei einem Linearsegment als auch bei einem Kreis- bzw. Helixsegment haben die Achsgeschwindigkeiten, abgesehen von der Zeit, dieselbe Längeneinheit wie die Achsposition (Mikrometer oder Bogensekunde).
Die Einheit der Werkzeuggeschwindigkeit ist natürlich dieselbe wie die der Achsgeschwindigkeiten, aus denen sie berechnet wird.

3.2.11 Der Regler und seine Parameter

Jede Achse Ihrer Anlage wird durch einen auf der PC-Einsteckkarte befindlichen Achscontroller vom Typ LM628 bzw. LM629 überwacht.
Unter anderem bietet dieser Chip Ihnen einen digitalen PID-Regler, dessen Parameter Sie vor der Benutzung entsprechend den spezifischen Besonderheiten Ihrer Anlage einstellen müssen.

Aber bevor der PID-Algorithmus dieses Chips unter die Lupe genommen werden soll, werden wir Sie noch ein bisschen in die Theorie der Regelungstechnik einführen.

Ein kontinuierlicher bzw. analoger PID-Regler hat die Übertragungsfunktion:

$$G(s) = kp + ki / s + kd * s \qquad \text{mit}$$

s : Operator der LAPLACE-Transformation,
kp : Koeffizient des P-Anteils,
ki : Koeffizient des I-Anteils und
kd : Koeffizient des D-Anteils.

Das Produkt zwischen der Übertragungsfunktion und dem Fehlersignal E(s) (Fehlersignal = Soll-Wert - Ist-Wert) ist die Stellgröße U(s):

$$U(s) = E(s) * kp + E(s) * ki / s + E(s) * kd * s$$

Durch die Rücktransformation in die Zeitebene haben wir die zeitliche Gleichung für das Stellsignal:

$$u(t) = kp * e(t) + ki \int_0^t e(t)\,dt + kd * \frac{d\,e(t)}{d\,t} \quad ,$$

wobei u(t) sowie e(t) das Stellsignal sowie das Fehlersignal in der Zeitebene sind.

Durch die Diskretisierung dieser Gleichung bekommen wir die Differenzengleichung eines diskontinuierlichen bzw. digitalen PID-Reglers:

$$U(nTa) = kp * E(nTa) + ki * Ta * \underbrace{\sum_{i=0}^{n} E(iTa)}_{\int_0^t e(t)\,dt} + kd * \underbrace{\frac{E(nTa) - E((n-1)Ta)}{Ta}}_{\frac{d\,e(t)}{d\,t}}$$

P-Anteil I-Anteil D-Anteil

mit Ta: Abtastzeit des digitalen PID-Reglers,
U (n Ta) : Stellsignal zum Zeitpunkt t = n * Ta
E (n Ta) : Fehlersignal zum Zeitpunkt t = n * Ta
E ((n-1) * Ta) : Fehlersignal zum Zeitpunkt t = (n-1) * Ta

Diese originale Differenzengleichung sieht zwar schön aus, ist aber in der Praxis nur sehr begrenzt einsetzbar. Man muss sie modifizieren, um sie an die

realen Bedingungen anzupassen. Diese Problematik wollen wir Ihnen im Folgenden näher erläutern.
Bei einer kleiner werdenden Abtastzeit Ta tendiert das Verhalten eines digitalen Reglers immer mehr zu dem eines analogen Reglers. Dadurch können das Fehlersignal und die Störungen sehr schnell ausgeregelt werden. Aber je kleiner die Abtastzeit Ta ist, desto ungenauer ist die Näherungsberechnung der Differentiation im D-Anteil aufgrund der Störungen und der Ungenauigkeiten der Messsysteme. Deswegen wird die Differenzengleichung abgeändert.
Die modifizierte Differenzengleichung lautet:

$$U(nTa) = kp * E(nTa) + ki * Ta * \sum_{i=0}^{n} E(iTa) + kd * \frac{E(kTs) - E((k-1)Ts)}{Ts} \quad \text{mit}$$

Ts: Abtastzeit für die Differentiation des D-Anteils.

Die Abtastzeit Ts für die Berechnung der Differentiation muss nicht unbedingt mit der Abtastzeit Ta des Reglers identisch sein. Je größer Ts ist, desto genauer ist die Differentiation. Aber gleichzeitig wird die Stabilität des geschlossenen Systems schlechter.
Eine weitere Modifikation der Differenzengleichung betrifft die Integration des I-Anteils. Unter Umständen, wie z. B. bei einem großen Soll-Wert, bei dem das Fehlersignal sehr lange positiv bzw. negativ bleibt, kann es dazu führen, dass das Stellsignal durch die Integration einen sehr großen Wert erreicht. Einerseits überschreitet das Stellsignal die Grenze des Stellglieds der realen Anlage. Auf der anderen Seite verursacht ein großer Wert des Stellsignals eine starke Überschwingung über den Soll-Wert hinaus. Das führt dazu, dass das Positionieren sehr lange dauert. Deswegen ist es in der Praxis üblich, dass man den Integrationswert begrenzt.
Wenn der Integrationswert den zugelassenen Bereich verlässt, wird der Integrationsalgorithmus außer Kraft gesetzt. Dadurch bleiben der Integrationswert und somit auch das Stellsignal immer begrenzt.

Wir wollen Ihnen den digitalen PID-Regler des Achscontrollers jetzt etwas genauer erläutern, nachdem Sie sich schon mit der Theorie eines konventionellen PID-Reglers vertraut gemacht haben.

Der PID-Regler des Achscontrollers hat die Differenzengleichung:

$$U(nTa) = Kp * E(nTa) + Ki * \sum_{i=0}^{n} E(iTa) + Kd * (E(kTs) - E((k-1)Ts)) \quad \text{mit}$$

den beiden zusätzlichen Parametern Td und Il, die aber nicht in der Differenzengleichung vorkommen. Hier haben Sie schon bemerkt, dass der Term Ta im I-Anteil sowie der Term 1/Ts im D-Anteil gar nicht mehr vorhanden sind. Das kommt daher, dass diese beiden Parameter bereits in den jeweiligen Parametern Ki bzw. Kd mit eingegangen sind.

Die Modifikation der Differentiation wird durch den Parameter Td gewährleistet. Dabei haben wir die Beziehung:

$$Ts = (Td + 1) * Ta \qquad \text{mit}$$

$$Td = 0 \ldots 255$$

Durch die Änderung von Td können Sie die Abtastzeit Ts der Differentiation variieren. Der Parameter Td kann nur einen ganzen und positiven Wert zwischen 0 und 255 annehmen, d. h. die Abtastzeit Ts für die Differentiation ist immer ein ganzes Vielfache der Abtastzeit Ta.
Im Allgemein sollte man während der Inbetriebnahme Td zuerst gleich 0 setzen. Nachdem alle Reglerparameter fertig eingestellt wurden, kann Td dann schrittweise erhöht werden, um das Optimum des Regelverhaltens zu finden. Dabei muss der Faktor Kd entsprechend verkleinert werden (siehe unten).
Die Begrenzung der Integration kann man durch den Parameter Il problemlos realisieren.

Es gilt: $Il = 0 \ldots 32767 = 0 \ldots (2^{15} - 1)$

Der Parameter Il kann nur einen ganzen und positiven Wert in diesem Bereich annehmen. Nachdem Il eingestellt wurde, sind +Il bzw. -Il die obere bzw. die untere Grenze für die Integration. Es ist selbstverständlich, dass der I-Anteil des Reglers keine Wirkung hat, falls Sie Il gleich 0 setzen. Der Wertbereich 0 ... 32767 für Il ist mehr als groß genug, wenn man bedenkt, dass das Stellsignal des Achscontrollers eine 12-Bit-Auflösung hat. Bei einer 12-Bit-Auflösung liegt das Stellsignal im Bereich $-(2^{12} - 1) \ldots +(2^{12} - 1)$.

Neben den beiden oben genannten Modifikationen im Bezug auf den I-Anteil und den D-Anteil gibt es noch weitere Modifikationen für den PID-Algorithmus der Achscontroller. Diese Modifikationen haben einen rein rechnerischen Hintergrund. Sie ist notwendig, um den Rundungsfehler beim Berechnen zu vermeiden. Aufgrund dieser Modifikationen haben wir dann die folgenden Umrechnungsformeln zwischen den Parametern des PID-Reglers im Controller und den Parametern eines normalen PID-Reglers:

$$Kp = 256 * kp$$

$$Ki = 65536 * ki * Ta$$

$$Kd = 256 * kd / Ts = 256 * kd / (Td * Ta)$$

Ähnlich wie bei Td und Il können Kp, Ki und Kd nur positive und ganze Werte annehmen. In Bezug auf den Wertbereich gilt:

$$Kp = 0 \ldots 32767 = 0 \ldots (2^{15} - 1)$$
$$Ki = 0 \ldots 32767 = 0 \ldots (2^{15} - 1)$$
$$Kd = 0 \ldots 32767 = 0 \ldots (2^{15} - 1)$$

Die oben genannten Formeln bilden die Grundlagen für die Berechnung der Reglerparameter im Achscontroller. Zuerst können Sie die Reglerparameter kp, ki und kd mit einer beliebigen, Ihnen vertrauten, Reglerentwurfsmethode berechnen. Aus diesen Parametern können Sie dann Kp, Ki und Kd berechnen. Sie müssen Kp, Ki und Kd natürlich noch abrunden.
Die Werte Kp, Ki, Kd, Il und Td sind die Werte, die Sie am Ende entweder über das Konfigurationsprogramm als Standardwerte oder über die Funktion 4 (siehe Kapitel 3.3.4) dem Achscontroller übergeben müssen.
Für alle Berechnungen brauchen Sie noch den Wert für die Abtastzeit Ta des digitalen Reglers im Achscontroller. Es gilt:

$$Ta = 2048 / 6\ MHz$$

Dabei ist 6 MHz die Taktfrequenz, mit der der Chip LM628 oder der Chip LM629 auf unserer PC-Einsteckkarte arbeiten. Der Wert 2048 ist eine interne Konstante des Chips.

3.2.12 Wie können Sie den Treiber benutzen?

Der Treiber stellt Ihnen verschiedene Funktionen zur Verfügung. Diese Funktionen können Sie nutzen, in dem Sie einen sogenannten Software-Interrupt aufrufen. Die Nummer dieses Software-Interrupts können Sie durch das Konfigurationsprogramm selbst wählen (siehe Handbuch für das Konfigurationsprogramm PARKON.EXE).

Anhand der Daten, die Sie vor dem Interrupt-Aufruf in den einzelnen Prozessorregistern abgelegt haben, weiß der Treiber dann, welche Funktion Sie benutzen wollen und wie er diese Funktion ausführen soll.
Nachdem die Funktion ausgeführt wurde, legt der Treiber seine Daten in die entsprechenden Prozessorregister und kehrt zurück.
Aufgrund der zurückgegebenen Daten wissen Sie dann, wie die Funktion ausgeführt wurde.

An dieser Stelle sagen wir Ihnen gleich, dass nur die 4 Prozessorregister ax, bx, cx und dx für den Datenaustausch zwischen einem Anwenderprogramm und dem Treiber benutzt werden und dass die Nummer der von Ihnen gewünschten Funktion immer in das Prozessorregister ax geladen werden muss und dass immer das Prozessorregister al bei Rückkehr vom Treiber den Fehlercode für die Ausführung der gewünschten Funktion erhält.

Bis jetzt haben wir über einen Interrupt-Aufruf gesprochen, aber wie man einen Interrupt-Aufruf realisieren kann, haben wir Ihnen noch nicht verraten. Falls Sie diese Programmiertechnik schon kennen, können Sie im nächsten Kapitel weiterlesen. Falls es nicht der Fall ist, geben Sie sich ein bisschen Mühe, bis zum Ende dieses Kapitels zu lesen. Danach wissen auch Sie Bescheid, wie man das macht.

Vor dem Aufruf eines Interrupts müssen Sie zuerst Ihre Daten in den Prozessorregistern ablegen. Danach teilen Sie dem Prozessor mit, welchen Interrupt Sie haben möchten. Der Prozessor ruft dann die entsprechende Interrupt-Routine auf. Die aufgerufene Interrupt-Routine beendet ihre Arbeit und kommt zurück. In den Prozessorregistern liegen die Daten, die die Interrupt-Routine Ihnen übergeben will. Damit haben Sie einen Interrupt-Aufruf ausgeführt.

Im Folgenden wollen wir Ihnen an einem kleinen Beispiel in der Programmiersprache C (Compiler Microsoft C 6.00A) zeigen, wie Sie einen Interrupt aufrufen können.

```
#include <dos.h>

void main( void )
{
int IntNr ;                    /* Deklaration für die Interrupt-Nummer */
union REGS ProReg ;    /* Deklaration für die Prozessorregister */
      .
      .
IntNr = ... ;             /* Hier initialisieren Sie die Variable IntNr mit der Nummer
                             des Interrupts, den Sie gern aufrufen möchten */
                          /* Hier laden Sie die Prozessorregister mit Ihren Daten */
ProReg.x.ax = ... ;   /* Hier laden den Prozessorregister ax */
ProReg.h.bl  = ... ;   /* Hier laden den Low-Byte vom Prozessorregister bx */
ProReg.h.bh = ... ;   /* Hier laden den High-Byte vom Prozessorregister bx */

/*Auf einer ähnlichen Weise können Sie alle anderen Prozessorregister laden*/

int86(IntNr., &ProReg, &ProReg);     /* Hier rufen Sie Ihren Interrupt auf */

/* Ab dieser Stelle stehen die Daten, die die Interrupt-Routine Ihnen mitteilen
will, in ProReg. Sie können auf diese Daten zugreifen, indem Sie die einzelnen
Variablen, wie z. B. ProgReg.x.ax oder ProgReg.x.bx ... benutzen. */
      .
      .
}
```

Auf die gleiche Weise können Sie in anderen Programmiersprachen Ihren Interrupt aufrufen. Natürlich hat jede Programmiersprache ihre eigenen Befehle, um den Interrupt aufzurufen.

3.2.13 Die Zweier-Komplementär-Darstellung der Zahlen

An verschiedenen Stellen haben wir schon erwähnt, dass die Austauschdaten zwischen einem Anwenderprogramm und dem Treiber im Zweier-Komplementär-Format dargestellt werden müssen. An dieser Stelle wollen wir Ihnen erläutern, wie Sie ganze Zahlen in das Zweier-Komplementär-Format transformieren können.
Um die positiven Zahlen brauchen Sie Sich nicht zu kümmern, weil die Transformation nur die negativen Zahlen betrifft. Sie müssen zuerst den Betrag der negativen Zahl bilden. Dieser Betrag wird bitweise negiert und danach um 1 inkrementiert.

Das Ergebnis ist dann die negative Zahl in der Zweier-Komplementär-Darstellung. Am Beispiel von +3 und -3 wollen wir die Vorgehensweise der Transformation verdeutlichen:

00000011B ---> Die positive Zahl 3 bzw. Betrag von der negativen Zahl -3
11111100B ---> Bitweise Negation
11111101B ---> Um 1 inkrementiert

00000011B = 03H ---> Zweier-Komplementär-Darstellung von +3
11111101B = FDH ---> Zweier-Komplementär-Darstellung von -3

Dieses Datenformat wird von jeder Hochprogrammiersprache unterstützt. Das ist der entscheidende Vorteil von diesem Datenformat.
Der Austausch der Daten zwischen einem Anwenderprogramm und dem Treiber geschieht durch die Prozessorregister. Im Falle großer Datenmengen wird noch ein Speicherbereich zusätzlich benutzt, dessen Anfangsadresse über die Prozessorregister übergeben wird.
Abgesehen von der Adresse müssen alle sowohl über die Prozessorregister als auch die über den zusätzlichen Speicherbereich übergebenen Daten im Zweier-Komplementär-Format stehen. Für die über einen zusätzlichen Speicherbereich übergebenen Daten gilt außerdem, dass das niederwertigste Byte einer Zahl immer an der niedrigsten Adresse, d. h. an der Anfangsadresse des für diese Zahl reservierten Speicherbereiches, und dass das höchstwertigste Byte immer auf der höchsten Adresse liegen muss.
Diese Speicherzuordnung wird auch von allen Hochprogrammiersprachen unterstützt.

Zusammenfassend kann man sagen, dass Sie sich über die Datentransformation und die Belegung eines Speicherbereichs mit den zu übergebenen Daten absolut keine Gedanken zu machen brauchen, wenn Sie Ihr Anwenderprogramm in einer Hochprogrammiersprache schreiben. Es ist nur dann interessant, wenn Sie in Assembler programmieren.
Anschließend wollen wir ein kleines Programm in C zeigen, um zu verdeutlichen, was wir Ihnen oben erzählt haben.

Aus diesem Beispiel sollten Sie entnehmen, wie Sie die verschiedenen Variablentypen und Speicherbereiche für den Datenaustausch zwischen Ihrem Programm und dem Treiber deklarieren können.

```
void main( void )
{
short KurzZahl;
long LangZahl;

/* Der Compiler legt die beiden Variablen KurzZahl und LangZahl als Zweier-
Komplementär-Zahlen im Speicher ab. Die Variable KurzZahl belegt 2 Byte und
die Variable LangZahl belegt 4 Byte. Dabei sind (void far*)&KurzZahl und (void
far*)&LangZahl die Anfangsadressen des jeweiligen Speicherbereichs. Für die
beiden Variablen gilt, dass das niederwertigste Byte an der Anfangsadresse
des für sie reservierten Speicherbereiches und das höchstwertigste Byte an der
höchsten Adresse liegen. */

struct
   {
     long X ; long Y ; long Z ; long A ;
   } Position ;

/* Die Struktur Position belegt genau 16 Byte im Speicher. (void far*)&Position
ist die Anfangsadresse dieses Speicherbereichs. Jede der Variablen Position.X,
Position.Y, Position.Z und Position.A wird als eine Zweier-Komplementär-Zahl
dargestellt und belegt 4 Byte im Speicher.
Für jede Variable gilt, dass das niederwertigste Byte an der Anfangsadresse
des für sie reservierten Speicherbereiches und das höchstwertigste Byte an der
höchsten Adresse liegen. */

long Array[6];

/* Jedes Element von Array wird als eine Zweier-Komplementär-Zahl dargestellt
und belegt 4 Byte im Speicher. Für jede Variable gilt, dass das niederwertigste
Byte an der Anfangsadresse des für sie reservierten Speicherbereiches und
das höchstwertigste Byte an der höchsten Adresse liegen. Das Feld Array
belegt genau 24 Byte im Speicher. (void far*)&Array[1] bzw. (void far*)Array ist
die Anfangsadresse dieses Speicherbereichs. */
  .
  .
  .
}
```

3.2.14 Ist der Treiber schon installiert?

Es ist natürlich sehr peinlich, wenn ein Anwenderprogramm den Software-Interrupt aufruft, um sich mit dem Treiber zu unterhalten, obwohl der Treiber noch gar nicht installiert ist. In solchen Fällen bleibt meistens nichts anders übrig, als die Reset-Taste des PC zu drücken.
Deswegen ist es schon empfehlenswert, dass ein Anwenderprogramm beim Start kontrollieren sollte, ob der Treiber schon da ist oder nicht.

Auf welche Weise können Sie das kontrollieren?
Über einen Interrupt-Aufruf geht das natürlich nicht, weil der Aufruf ins Leere gehen würde, falls der Treiber noch nicht installiert ist. Dann hat die Reset-Taste darunter zu leiden. Im Rahmen unseres Produkts bieten wir Ihnen zwei kleinen Assemblerroutinen, *CheckInstall* und *KonInstall*, die Sie in Ihr Programm einbinden sollten. (Bevor Sie weiterlesen, sollten Sie die mitgelieferten Assemblerroutinen in der Datei *KON_INS.ASM* ausdrucken).
Mit Hilfe dieser Routinen sind Sie in der Lage, das Vorhandensein des Treibers zu kontrollieren und die Nummer des für die Kommunikation zwischen dem Anwenderprogramm und dem Treiber zuständigen Software-Interrupts zu ermitteln. Um mögliche Fehlerquellen im Zusammenhang mit der Benutzung einer falschen Nummer des Software-Interrupts zu vermeiden, ist die Benutzung dieser Assemblerroutinen sehr empfehlenswert, obwohl die Software-Interruptnummer Ihnen bekannt ist, weil *Sie selbst* sie definieren müssen (siehe Handbuch für das Konfigurationsprogramm PARKON.EXE).

Diese Routinen benutzen die Tatsache, dass der Treiber bei seiner Installation den DOS-Multiplex-Interrupt 2F umbiegt. Zusätzlich legt der Treiber dabei noch eine eigene Identifikationszeichenkette an.
Das Umbiegen des Multiplex-Interrupts 2F ist eine sehr häufig verwendete Technik beim Programmieren von residenten Programmen. Jedes residente Programm biegt diesen Interrupt auf seine eigene Routine und bekommt eine eigene Identifikationsnummer, die im Bereich 0C0h ... 0FFh liegt. Alle Nummern kleiner als 0C0h sind bereits von DOS-Treibern belegt. Falls der Treiber schon installiert ist, bekommt er auch seine eigene Identifikationsnummer, die im Bereich 0Ch ... 0FFh liegt.
In der Routine *CheckInstall* wird der Interrupt 2F mit dem Register ah gleich der Identifikationsnummer und dem Register al gleich 0 aufgerufen. Wenn das Register al nach dem Aufruf gleich 0FFh ist, bedeutet dies, dass die Identifikationsnummer von *irgendeinem* residenten Programm belegt ist.

Um feststellen zu können, dass es sich dabei um unseren Treiber handelt, muss noch weiter kontrolliert werden. Die neue von unserem Treiber angelegte Routine des Interrupts 2F legt vor der Rückkehr in das aufrufende Programm in den Registern ds:si die Anfangsadresse der oben erwähnten Identifikationszeichenkette und in das Register ah die Nummer des Software-Interrupts ab. *CheckInstall* kontrolliert anhand ds:si, ob es sich dabei um die

vom Treiber angelegte Identifikationszeichenkette handelt. Wenn es der Fall ist, wird das Register ax gleich dem Register ah gesetzt.
Danach kehrt *CheckInstall* mit dem gesetzten Carry-Flag in die aufrufende Routine *KonInstall* zurück.

Anhand des gesetzten Carry-Flags und des Registers ax weiß *KonInstall*, dass der Treiber schon installiert ist und welche Nummer der Software-Interrupt hat. Wenn die Register ds:si nicht auf die Identifikationszeichenkette zeigen oder wenn das Register al nach dem Aufruf des Interrupts 2F nicht gleich 0FFh ist, wird die Identifikationsnummer in ah um 1 inkrementiert und das Ganze wiederholt sich bis ah gleich 0FFh. Falls die Identifikationszeichenkette nicht gefunden werden kann, kehrt _CheckInstall mit dem zurückgesetzten Carry-Flag zu _KonInstall zurück. Anhand dieses zurückgesetzten Carry-Flags weiß _KonInstall, dass der Treiber noch nicht installiert ist.

Vor der Rückkehr der Routine _KonInstall in das aufrufende Anwender-programm lädt sie das Register ax mit 0, wenn der Treiber noch nicht installiert ist. Sonst ist das Register ax gleich der Nummer des Software-Interrupts.
Sie als Anwender brauchen sich um die Routine _CheckInstall nicht zu kümmern, weil diese von der Routine _KonInstall aufgerufen wird. Neben dem Quellcode bekommen Sie von uns noch das Object-File dieser beiden Assemblerroutinen. Das Object-File wurde mit Microsoft Macro Assembler Version 6.1 erzeugt. Die Routine _KonInstall wurde so geschrieben, dass sie als ein Unterprogramm aufgerufen werden kann, wenn Sie Ihr Programm in der Programmiersprache C schreiben. Der Integer-Rückkehrwert von der C-Routine KonInstall ist der Wert im Register ax.

Im Folgenden zeigen wir Ihnen an einem Beispiel, wie Sie diese Routine in Ihrem Programm benutzen können.

```
short KonInstall( void )                    /* Prototype der Routine in C */

void main( void )
{
int SoftwareInterruptNummer ;
        .
        .
        .
/* Anhand des Rückkehrwerts vom KonInstall( ) können Sie feststellen, ob der
Treiber schon installiert ist oder nicht und welche Nummer der Software-
Interrupt hat.*/

if( (SoftwareInterruptNummer = KonInstall( ) ) == 0 )
  {
       ...                          /* Der Treiber ist noch nicht installiert */
   }
else
```

```
    {
        ...                     /* Der Treiber ist bereits installiert */
    }
.
.
.
}
```

Wenn Sie in einer anderen Sprache als C programmieren, müssen Sie diese beiden Assembler-Routinen anhand des mitgelieferten Quellcodes entsprechend modifizieren und neu übersetzen. Aber das dürfte eigentlich kein Problem für Sie sein.

3.2.15 Treiberversion 3.10 und Software-Unterschiede zu der Version 3.00

Abgesehen von der Hardwareänderung (siehe Kapitel 2.7) weist die Treiberversion 3.10 eine Menge von Änderungen und Erweiterungen auf.

Die größte Software-Änderung ist mit der Hardware-Änderung verbunden. Es betrifft die Referenzfahrt und den Schlüsselschalter.
Bei der Referenzfahrt wird nicht mehr ein Watch-Dog-Signal am PIN 43 ausgelöst wie bei der Version 3.00. Das Überbrücken des Sicherheitskreises muss separat über das Ausgabeport an PIN 40 erfolgen. Benutzen Sie dazu die Treiberfunktion 48.
Der Schlüsselschalter, der bei der Treiberversion 3.00 das manuelle Überbrücken des Sicherheitskreises ermöglicht, fällt bei der Version 3.10 ganz weg, d. h. das Herausfahren aus einem aktiven Hardware-Endschalter muss auch mit Hilfe der Funktion 48 realisiert werden.

Bei der Treiberversion 3.10 legen wir einen großen Wert auf die Sicherheit der Anlagen. Eine der Maßnahmen ist das ständige Auslösen von Hardware-Interrupt IRQ10 oder IRQ11. In der Version 3.00 wird der Hardware-Interrupt nur im Fall eines aktiven Bewegungssegments ausgelöst, d.h. der Hardware-Interrupt dient ausschließlich zur Generierung des Interpolatiostaktes.
Dies führt dann zu dem Nachteil, dass die Servoanlage nicht überwacht werden kann, falls kein Bewegungssegment im Hintergrund aktiv ist.
In machen Situationen, wie z. B. beim Einschalten des Servocontrollers, kann es zu einer ruckartigen Bewegung führen.

Bei der Version 3.10 wird der Hardware-Interrupt ständig ausgelöst, egal ob ein Bewegungssegment aktiv ist oder nicht. Dadurch ist es möglich, die Servoanlage ständig zu überwachen.
Durch das Einführen des Kontroll-Bytes (siehe Treiberfunktion 49) ist der Treiber in der Lage, auf solche Hardwarefehler wie z. B. Stromausfall, Bruch der Encoder-Signalleitungen etc. sofort zu reagieren. Eine unkontrollierte Bewegung ist nicht mehr möglich. Selbst der Softwareabsturz ist durch das

Watch-Dog-Signal abgefangen. In diesem Fall wird der Servocontroller sofort stromlos geschaltet. Eine Gefahr kann gar nicht mehr entstehen.

Falls Sie eine genaue Zeitbasis brauchen, können Sie die neue Treiberfunktion 52 nehmen. Mit dieser Funktion können Sie jederzeit die abgelaufene Zeit seit der Treiberinstallation abfragen. Sie bekommen die Zeitangabe zwar in Millisekunden zurück, die Genauigkeit der Zeitangabe ist aber in Mikrosekunden. Diese Funktion steht nicht in Konkurrenz zu der Funktion 7, mit der Sie eine Zeitverzögerung realisieren können.

Bei den Servoanlagen gibt es immer eine Abweichung zwischen der Ist- und Soll-Position der Achsen. Mit der Funktion 18 sind Sie in der Lage, jederzeit die Ist-Position der Achsen abzufragen. Die Abfrage der aktuellen Soll-Positionen ist in vielen Fällen sehr sinnvoll aber in der Treiberversion 3.00 leider noch nicht möglich. In der Treiberversion 3.10 können Sie mit den beiden Funktionen 50 und 51 sehr bequem und einfach die aktuellen Soll-Positionen abfragen.

Eine weitere Änderung der Treiberversion 3.10 umfasst die Ein- und Ausgaben. Die Funktionen 16 und 17 zum direkten Lesen und Schreiben von Ein- und Ausgabeports werden nicht mehr unterstützt. Alle Ein- und Ausgabeoperationen erfolgen nur noch über den Treiber. Damit wollen wir verhindern, dass Sie wahllos auf jeden Ein- und Ausgabeport zugreifen. In solchen Fällen könnte es gefährlich werden.

Neben den Funktionen 36 bis 42 für die Ein- und Ausgabeoperationen über Kanalnummern haben wir noch die Funktionen 53 bis 56 eingeführt. Mit Hilfe dieser neuen Funktionen können Sie auf zwei Ein- und zwei Ausgabeports zugreifen. Diese Ports sind frei definierbar.
Mit dem Einführen dieser Ports wollen wir eine klare Trennung zwischen Ein- und Ausgabeports für die Anwender (Funktionen 36 bis 42) und Ein- und Ausgabeports für steuerungseigene Zwecke (Funktionen 53 bis 56) erreichen. Bei allen als benutzt definierten Ausgabeports werden die Ausgabewerte intern im Treiber abgespeichert. Diese Werte können Sie jederzeit zurücklesen (Funktionen 40, 41 und 55). Außerdem werden diese Werte periodisch vom Treiber in seiner Hardware-Interruptroutine an den Ausgabeports ausgegeben.

Auf diese Art und Weise wollen wir Sie zwingen, die Portausgabe mit den Treiberfunktionen zu machen. Falls Sie ohne die Benutzung der Treiberfunktionen direkt ausgeben, werden die Ports in der nächsten Hardware-Interruptroutine wieder mit den alten Werten beschrieben. Somit bleibt Ihre direkte Ausgabe wirkungslos.

Mit Hilfe der Funktion 57 können Sie die Achsen umschalten, d. h. Sie sind jederzeit in der Lage, die Achsnummerierung X, Y, Z, A intern im Treiber zu ändern. Das Achsenumschalten ist sehr sinnvoll für viele Anwendungsfälle, wie z. B. bei Maschinen für die Serienfertigung.

Im Zusammenhang mit dieser Funktion können Sie mit Hilfe der Funktion 58 das Gravieren auf einer Zylinderoberfläche sehr einfach realisieren.
Zusätzlich zu den Achsreglerparameter Kp, Ki, Kd, Il und Td haben wir noch den Achsgeschwindigkeits-Verstärkungsfaktor Kv eingeführt. Dieser Faktor ist der dynamische Kennwert der betreffenden Achse. Aufgrund dieses Faktors müssen Sie bei der Funktion 4 den Speicherbereich für die Parameterübergabe genau um 4 Byte erweitern, um den Faktor Kv übergeben zu können.

3.2.16 Keine oder ruckartige Bewegung, was haben Sie falsch gemacht?

Der Datenaustausch mit dem Treiber geschieht über den Software-Interrupt, dessen Nummer Sie frei definieren können (siehe Kapitel 3.2.12).
Beim Hardware-Interrupt haben Sie die Auswahl zwischen IRQ10 oder IRQ11 (siehe Handbuch für das Konfigurationsprogramm PARKON.EXE). Der Hardware-Interrupt ist zuständig für das Generieren vom Interpolationstakt, in dem die Stützpunkte der Bewegung berechnet werden. Wenn dieser Interrupt nicht ausgelöst werden kann, kann der Treiber auch keine Stützpunkte berechnen. Deswegen wird es auch keine Bewegung geben.
Wenn der Interrupt nur unregelmäßig ausgelöst wird, werden die Stützpunkte der Bewegung auch nur unregelmäßig berechnet. Dadurch ist die Bewegung ruckartig. Im Folgenden werden die verschiedenen Gründe genannt, die dazu führen können, dass Sie keine oder nur eine ruckartige Bewegung bekommen.

Ein Grund dafür, warum der Hardware-Interrupt nicht ausgelöst wird, besteht darin, dass der von Ihnen gewählte Interrupt (IRQ10 oder IRQ11) bereits von einer anderen Hardware benutzt ist. Sie haben zwei Wahlmöglichkeiten. Wenn es mit der einen nicht geht, müssen Sie die andere Möglichkeit nehmen. Falls es trotzdem nicht weiterhilft, müssen Sie einen der Hardware-Interrupts freimachen, in dem Sie die entsprechende Soft- und Hardware entfernen, die den von Ihnen gewünschten Hardware-Interrupt weggenommen haben. Bei Pentium-Rechnern mit PCI-Bus kann es sein, dass Sie im BIOS Ihres Rechners den gewünschten Interrupt erst frei lassen, bevor Sie ihn benutzen können (siehe Kapitel 2.4).

Die Hardware-Interrupte können nicht nur hardwaremäßig sondern auch softwaremäßig gesperrt werden. Auf der Assembler-Ebene ist der Befehl „CLI“ zuständig für das Sperren aller Interrupt-Anforderungen. Das Gegenstück ist der Befehl „STI“ für das Freilassen der Hardware-Interrupts.
Sie sollten mit der höchsten Sparsamkeit den Befehl „CLI“ benutzen. Wenn es anders nicht geht, sollen die Interrupts so schnell wie möglich durch den Befehl „STI“ wieder freigelassen werden. Wenn Sie mit einer Hochsprache wie „C“ oder „PASCAL“, ... programmieren, werden Sie wahrscheinlich häufig die Bibliothekroutinen benutzen. Weil viele Bibliothekfunktionen den Befehl „CLI“ intern aufrufen, werden die Interrupts unbewusst von Ihnen gesperrt. Erfahrungsgemäß wird der Befehl „CLI“ sehr häufig von den Bibliothekroutinen

benutzt, die zuständig für die direkte Eingabe von der Tastatur oder die direkte Ausgabe an dem Monitor sind. Einige Routinen der Programmiersprache „C“ wollen wir hier nennen. Es sind „printf()“, „scanf()“, „putch()“, „getch()“, ... Wenn eine Bewegung im Hintergrund aktiv ist, sollten Sie so wenig wie möglich solche Routinen benutzen.

Während einer Bewegung besteht oft der Wunsch, verschiedene Informationen vom Treiber durch Aufrufe von entsprechenden Funktionen zu holen. Die häufig benutzten Funktionen sind die Funktion 5 zum Abfragen des Treiberstatus' sowie die Funktionen 18 und 19 zum Abfragen der Ist-Positionen und Ist-Geschwindigkeit. Wegen des Koordinierens der Daten innerhalb des Treibers führt ein Aufruf von Treiberfunktionen zwar nicht immer aber öfter zu einem kurzzeitigen Sperren von Hardware-Interrupts. Deswegen ist es schon sinnvoll, dass Sie eine kurze Zeitverzögerung zwischen den Aufrufen der Treiberfunktionen einfügen. Besonders kritisch ist das Abfragen der Ist-Positionen, weil die Controllerchips LM628 dabei besonders belastet werden. Eine Zeitverzögerung von etwa 5 ms ist hier sehr sinnvoll.

3.2.17 Satzzykluszeit

Die Satzzykluszeit ist die Zeit, die der Treiber für die Interpolationsvorbereitung benötigt, d. h. jedes Mal wenn Ihre Applikationssoftware eine Treiber-bewegungsfunktion aufruft, benötigt der Treiber diese Zeit, um die Vorberechnung für die Interpolation durchzuführen.
Nach der Vorberechnungszeit fängt die Bewegung erst an.
Die Vorberechnungszeit hängt sehr stark davon ab, was für einen PC Sie haben. Bei einem PC 486DX (66 MHz) haben wir folgende Satzzykluszeit gemessen:

Interpolationsmethode	Satzzykluszeit
Linearinterpolation (Funktionsnummer 24, 25, 26, 27)	12,90 ms
Kreis- und Helixinterpolation (Funktionsnummer 28, 29, 30, 31)	14,24 ms

Diese Satzzykluszeit hat nur Bedeutung für die vektorielle Bearbeitung. Für die Bahnbearbeitung hat diese Satzzykluszeit keinen Sinn, weil die Interpolations-daten schon vorher mit Hilfe des Bahndatengenerators berechnet werden. Man kann sagen, dass die Satzzykluszeit für die Bahnbearbeitung gleich 0 ist.

3.2.18 Treiberstart mit dem Optionsschalter /DSM für die Steuerung der Arbeitsspindel

Dieser Start-Modus wurde eingeführt, um die Besonderheiten der Bedienoberfläche der Firma ***isel****automation* im Zusammenhang mit der Spindelsteuerung zu berücksichtigen. Der Treiber mit dem Optionsschalter /DSM wird folgendermaßen gestartet:

[Laufwerk:\][Pfad] ISELDRV.EXE *.INI [/DSM]

Der Sinn und der Zweck des Optionsschalters /DSM ist die Benutzung eines freien LM628-Chips auf der PC-Einsteckkarte UPMV4/12, um die Arbeitsspindel zu steuern. Falls der Treiber mit diesem Optionsschalter gestartet wurde und Sie die Funktion 59 aufrufen, um den Spindel-Modus der letzten Achse zu aktivieren, wird nicht die letzte Achse sondern die Achse dahinter in den Spindel-Modus eingeschaltet. Die Anzahl der Achsen im Interpolationsverbund ändert sich nicht.

Das folgende Beispiel soll es Ihnen verdeutlichen.
Sie definieren mit dem Programm PARKON.EXE die Achsenanzahl gleich 3, d. h. Ihre Anlage hat drei Achsen X, Y und Z. Falls der Optionsschalter /DSM aktiv ist und Sie die Funktion 59 erfolgreich aufgerufen haben, steht nicht die Achse Z sondern die Achse A im Spindel-Modus, obwohl Ihre Anlage laut Definition nur die drei Achsen X, Y und Z hat. Diese drei Achsen stehen weiterhin im Verbund der Interpolationsachsen.
Wenn der Spindel-Modus auf diese Art aktiviert wurde, bleibt dieser Modus für immer erhalten. Ein Aufruf der Funktion 59, um den Spindel-Modus auszuschalten, kehrt mit dem Fehlercode 102 sofort zurück. Selbst die Reset-Funktion kann den Spindel-Modus nicht mehr deaktivieren.
Im Unterschied zu dem Normalfall funktioniert die Achsenumschaltung mit der Funktion 57 selbst bei einem aktiven Spindel-Modus. Über die Funktionen 60 bis 66 kann die im Spindel-Modus stehende Achse ganz normal angesprochen werden. Wenn Sie mit dem Programm PARKON.EXE die Achsenanzahl gleich 4 definiert haben, wird der Optionsschalter /DSM schon während der Treiberinitialisierung deaktiviert. Mit der Funktion 65 haben Sie jederzeit die Möglichkeit abzufragen, ob der Treiber mit dem Optionsschalter /DSM gestartet wurde oder nicht.
Sie können diesen Start-Modus für Ihre eigene Applikation benutzen. Aber falls Sie wollen, können Sie für die Spindelsteuerung auch unsere Software benutzen. Das Programm SP_DIREC.EXE mit der entsprechenden Dokumentation ist auf der Installationsdiskette mitgeliefert.

3.2.19 Was müssen Sie machen, um unsere Steuerung in WINDOWS NT/2000 zu benutzen?

Unser Treiber ist ein MS-DOS-Programm, das Sie aber ohne weiteres im DOS-Fenster von WINDOWS 3.xx, 9x oder OS/2 benutzen können (s. Kapitel 1.4).
In WINDOWS NT/2000 können Sie weiterhin DOS-Fenster starten.
Die Sicherheitsmechanismen von WINDOWS NT/2000 verbieten aber direkte Hardware-Zugriffe, d. h. Sie können unsere Steuerung im DOS-Fenster von WINDOWS NT/2000 nicht ohne weiteres laufen lassen, weil unser Treiber direkt auf die Hardware zugreift.
Um die Sicherheitsmechanismen von WINDOWS NT/2000 umzugehen, benötigen Sie noch eine zusätzliche Software, die diese Mechanismen ausschaltet. Es gibt hier verschiedene Anbieter. Wir haben bisher nur die Software "DOS Enabler" der Firma "Kithara Software" (http://www.kithara.de) getestet. Aufgrund der guten Funktionalität und des günstigen Preis ist die

Benutzung dieser Software zu empfehlen. In der mitgelieferten Dokumentation wurde schon sehr ausführlich beschrieben, wie die Software konfiguriert werden soll.
Im Zusammenhang mit unserer Steuerung wollen wir hier noch etwas dazu ergänzen. Um unseren Treiber zu benutzen, brauchen Sie eine Bedienoberfläche, d.h. es laufen zwei DOS-Programme gleichzeitig in einem DOS-Fenster. Hier ist es notwendig, ein Batch-Programm (*.bat) zu erzeugen. Das Batch-Programm soll zuerst den Treiber starten und dann die Bedienoberfläche. Es reicht aber nicht aus, nur das Batch-Programm in das "Kithara - Control Center" einzugeben. Sie müssen sowohl das Batch-Programm als auch den Treiber sowie die Bedienoberfläche in das "Kithara - Control Center" eingeben. Für das Batch-Programm brauchen Sie keine Hardware-Resourcen zu konfigurieren. Es müssen aber die benutzten Hardware-Resourcen sowohl für den Treiber als auch für die Bedienoberfläche eingegeben werden. Nach der Konfiguration ist ein Neustart des Rechners notwendig, um die Einstellungen wirksam werden zu lassen.

3.2.20 Fehlermöglichkeiten beim Treiberaufruf

Bei jedem Aufruf einer der Treiberfunktionen bekommt das Anwenderprogramm unter anderem im Register al einen Fehlercode zurück. Anhand dieses Fehlercodes können Sie feststellen, ob die Funktion vom Treiber ordnungsgemäß ausgeführt wurde. Wenn das nicht der Fall ist, können Sie durch diesen Fehlercode feststellen, warum die Ausführung der Funktion gescheitert ist. Im Folgenden wollen wir Ihnen die einzelnen Fehlercodes und ihre Bedeutungen erläutern.

Fehlercode = 0: Fehlerfrei
Die Funktion ist vom Treiber akzeptiert und fehlerfrei ausgeführt worden.

Fehlercode = 1: Falsche Funktionsnummer
Sie wollten eine Funktion aufrufen, die gar nicht vorhanden ist, d. h. Sie haben vor dem Funktionsaufruf eine falsche Funktionsnummer in das Register ax geladen.

Fehlercode = 2: Der Software-Interrupt ist bereits aktiv
Sie wollten eine Funktion aufrufen, während ein anderer Funktionsaufruf noch nicht beendet ist, d. h. es liegt ein verschachtelter Aufruf der Treiberfunktionen vor. Dieser Fehler kann normalerweise nicht auftreten, wenn Sie die Treiberfunktionen nur aus einem Anwenderprogramm heraus aufrufen. Falls Sie aber versuchen, die Treiberfunktionen von mehreren voneinander unabhängigen Prozessen, wie z. B. von Ihrem Anwenderprogramm oder auch von einer Hardware-Interruptroutine aufzurufen, kann dieser Fehler auftreten. Wir raten Ihnen aber, diese Methode in Ihrem Programm nicht zu verwenden, weil Sie dann nicht mehr in der Lage sind, die Funktionsaufrufe zu kontrollieren und auf einen eventuell vorhandenen Fehler entsprechend zu reagieren. Sie würden sich die Haare raufen.

Fehlercode = 3: Break-Fehler
Die Funktion kann nicht ausgeführt werden, weil die Breakfunktion aufgerufen wurde (siehe Kapitel 3.3.10). Nachdem Sie die Breakfunktion aufgerufen haben, wird ein sogenanntes Break-Flag intern im Treiber gesetzt. Solange dieses Flag gesetzt ist, können Sie nur noch die folgenden Funktionen aufrufen:

- Reset-Funktion (Funktion 2)
- Änderung der Reglerparameter (Funktion 4)
- Abfragen des Status' (Funktionen 5 und 49)
- Zeitfunktionen (Funktionen 7 und 52)
- Ein- oder Ausschalten des Hand-Modus (Funktion 8)
- Ein- oder Ausschalten des Teach-In-Modus (Funktion 9)
- Breakfunktion (Funktion 10)
- Einlesen von einem Port (Funktionen 36, 37, 40, 41, 53, 55 und 56)
- Ausgeben an einem Port (Funktionen 38, 39, 42, 48 und 54)
- Abfragen der Ist-Positionen (Funktion 18)
- Abfragen der Ist-Geschwindigkeit (Funktion 19)
- Abfragen der Soll-Positionen (Funktionen 50 und 51)
- Relative oder absolute Linearnormalbewegung (Funktionen 24 und 25) im Teach-In-Modus
- Operationen mit den reservierten Datenbytes (Funktionen 44, 45 und 46)
- Setzen des Sicherheitskreis-Ausgabeport (Funktion 48)

Das Break-Flag kann nur durch einen Aufruf der Reset-Funktion (Funktion 2) zurückgesetzt werden.

Fehlercode = 4: In-Bewegung-Fehler
Die Funktion ist nicht ausführbar, weil der Treiber momentan ein Bewegungssegment im Hintergrund ausführt. Falls das Abfahren eines Bewegungssegments im Hintergrund noch nicht fertig ist, können Sie die folgenden Funktionen nicht aufrufen:

- Ein- oder Ausschalten des Test-Modus (Funktion 3)
- Änderung der Reglerparameter (Funktion 4)
- Referenzfahrt (Funktion 6)
- Einschalten des Hand-Modus (Funktion 8)
- Einschalten des Teach-In-Modus (Funktion 9), wenn die Stop-Funktion vorher noch nicht aufgerufen wurde (siehe Kapitel 3.3.10)
- Setzen oder Löschen von Werkstücknullpunkt (Funktionen 11, 12 und 13)
- Bewegungsfunktionen (Funktionen 24 bis 31 und 33), wenn der Teach-In-Modus nicht eingeschaltet ist (siehe Kapitel 3.3.9)
- Achsen umschalten (Funktion 57)
- Definition des Zylinderradius' (Funktion 58)

Alle anderen Funktionen sind aufrufbar. Besonders interessant für Sie sind die Funktionen 5, 18 und 19 zum Abfragen des Treiberstatus', der Ist-Positionen und der Werkzeug-Ist-Geschwindigkeit. Damit haben Sie die Möglichkeit, diese Informationen in Ihrem Anwenderprogramm fortlaufend zu kontrollieren und anzuzeigen. Bei der Bearbeitung einer aus mehreren Bewegungssegmenten bestehenden Kontur müssen Sie ständig den Treiberstatus abfragen und kontrollieren, ob das Abfahren eines Segments schon fertig ist, um ein weiteres Bewegungssegment dem Treiber übergeben zu können.
Um die Werkzeuggeschwindigkeit während der Bearbeitung zu ändern, können Sie die Funktion 21 benutzen.

Fehlercode = 5: Software-Endschalter-Fehler
Dieser Fehler kann nur auftreten, wenn Sie die Funktion 14 oder 15 aufrufen. Mit diesem Fehlercode will der Treiber Ihnen mitteilen, dass das Werkzeug momentan außerhalb des durch die Software-Endschalter definierten Arbeitsbereichs liegt.
Durch das Abfragen des Treiberstatus (Funktion 5) können Sie immer bestimmen, welche Software-Endschalter momentan aktiv sind.
Beachten Sie, dass ein Software-Endschalter-Fehler nur dann auftreten kann, wenn Sie die Benutzung der Software-Endschalter durch die Funktion 15 zulassen. Wenn die Benutzung der Software-Endschalter nicht zugelassen ist, kann dieser Fehler auch nicht auftreten, unabhängig davon ob sich das Werkzeug momentan innerhalb oder außerhalb des durch die Software-Endschalter definierten Arbeitsbereich befindet.
Das Auftreten dieses Fehlers bedeutet aber nicht, dass die aufgerufene Funktion (Funktion 14 oder 15) nicht ausgeführt worden ist. Wenn Sie die Funktion 14 aufrufen, werden die neuen Software-Endschalter auf jeden Fall vom Treiber angenommen, falls der Software-Endschalter-Setzen-Fehler (Fehlercode = 6) nicht vorkommt. Das Gleiche gilt auch, wenn Sie die Benutzung der Software-Endschalter zulassen wollen (Funktion 15).
Die Benutzung der Software-Endschalter wird auf jeden Fall zugelassen, unabhängig davon ob der Software-Endschalter-Fehler vorkommt oder nicht.

Fehlercode = 6: Software-Endschalter-Setzen-Fehler
Dieser Fehler kann nur auftreten, wenn Sie die Funktion 14 aufrufen, um neue Software-Endschalter zu definieren.
Er besagt, dass mindestens auf einer Achse der Positionswert des positiven Software-Endschalters kleiner oder gleich dem Positionswert des negativen Software-Endschalters ist.
Die neuen Software-Endschalter werden vom Treiber nicht akzeptiert, unabhängig davon, ob die Benutzung der Software-Endschalter momentan zugelassen ist oder nicht. Der Treiber behält weiterhin die alten Software-Endschalter bei, wenn dieser Fehler auftritt.

Fehlercode = 7: Momentaner Hardware-Endschalter-Fehler
Er besagt, dass mindestens ein Hardware-Endschalter *momentan* aktiv ist. Die aufgerufene Funktion wurde nicht ausgeführt. Durch das Abfragen des Treiberstatus (Funktion 5) sind Sie immer in der Lage, zu wissen, welche Hardware-Endschalter momentan aktiv sind.

Im Fall eines Hardware-Endschalter-Fehlers können Sie nur noch folgende Funktionen aufrufen:

- Reset (Funktion 2)
- Ein- oder Ausschalten des Test-Modus (Funktion 3)
- Änderung der Reglerparameter (Funktion 4)
- Abfragen des Status' (Funktionen 5 und 49)
- Referenzfahrt (Funktion 6)
- Zeitfunktionen (Funktionen 7 und 52)
- Ein- oder Ausschalten des Hand-Modus (Funktion 8)
- Ein- oder Ausschalten des Teach-In-Modus (Funktion 9)
- StartStopBreak-Funktion (Funktion 10), falls der Teach-In-Modus eingeschaltet ist (siehe Kapitel 3.3.9)
- Einlesen von einem Port (Funktionen 36, 37, 40, 41, 53, 55 und 56)
- Ausgeben an einem Port (Funktionen 38, 39, 42, 48 und 54)
- Abfragen der Ist-Positionen (Funktion 18)
- Abfragen der Soll-Positionen (Funktionen 50 und 51)
- Abfragen der Ist-Geschwindigkeit (Funktion 19)
- Bewegungsfunktionen (Funktionen 24 und 25), falls der Teach-In-Modus eingeschaltet ist (siehe Kapitel 3.3.9)
- Operationen mit den reservierten Datenbytes (Funktionen 44, 45 und 46).

Es sei betont, dass sich die momentanen Zustände der Hardware-Endschalter durch die Reset-Funktion nicht ändern.

Fehlercode = 8: In-Bearbeitung-Hardware-Endschalter-Fehler
Die aufgerufene Funktion kann nicht ausgeführt werden, weil ein Hardware-Endschalter irgendwann einmal während der Bearbeitung aktiv war.
Sie dürfen diesen Fehler mit dem momentanen Hardware-Endschalter-Fehler (Fehlercode 7) nicht verwechseln. Während des Abfahrens eines Bewegungssegments werden die Hardware-Endschalter ständig überwacht. Falls einer der Hardware-Endschalter aktiv ist, wird ein Flag intern gesetzt. Dieses Flag bleibt solange gesetzt, bis Sie die Reset-Funktion aufrufen. Beachten Sie, dass dieses Flag bei einem aktiven Hardware-Endschalter nicht gesetzt wird, wenn die Bewegung im Teach-In-Modus stattfindet. Dieses Flag ist ein Indikator dafür, ob ein Hardware-Endschalter-Fehler während eines Bearbeitungsvorgangs aufgetreten ist oder nicht. Den Zustand dieses Flags können Sie jederzeit durch das Lesen des Treiberstatus' (Funktion 5) abfragen.

Wenn dieses Flag gesetzt ist, können Sie nur noch eine der folgenden Funktionen aufrufen:

- Reset-Funktion (Funktion 2)
- Änderung der Reglerparameter (Funktion 4)
- Abfragen des Status' (Funktionen 5 und 49)
- Referenzfahrt (Funktion 6)
- Zeitfunktionen (Funktionen 7 und 52)
- Ein- oder Ausschalten des Hand-Modus (Funktion 8)
- Ein- oder Ausschalten des Teach-In-Modus (Funktion 9)
- StartStopBreak-Funktion (Funktion 10), falls der Teach-In-Modus eingeschaltet ist (siehe Kapitel 3.9.9)
- Einlesen von einem Port (Funktionen 36, 37, 40, 41, 53, 55 und 56)
- Ausgeben an einem Port (Funktionen 38, 39, 42, 48 und 54)
- Abfragen der Ist-Positionen (Funktion 18)
- Abfragen der Ist-Geschwindigkeit (Funktion 19)
- Abfragen der Soll-Positionen (Funktionen 50 und 51)
- Bewegungsfunktionen (Funktionen 24 und 25), falls der Teach-In-Modus eingeschaltet ist (siehe Kapitel 3.9.9)
- Operationen mit den reservierten Datenbytes (Funktionen 44, 45 und 46)

Das Aufrufen der Reset-Funktion ist die einzige Möglichkeit, diesen Fehler zu beheben.

Fehlercode = 9: Referenzfahrt-Fehler
Die Funktion kann vom Treiber nicht ausgeführt werden, weil Sie bei mindestens einer Achse Ihrer Anlage noch keine Referenzfahrt gemacht haben. Solange der Treiber nicht aus dem Speicher entfernt wird, brauchen Sie die Referenzfahrt nur einmal zu machen. Intern reserviert der Treiber für jede Achse ein sogenanntes Referenz-Flag. Nachdem die Referenzfahrt durchgeführt wurde, wird dieser Flag gesetzt.
Abgesehen von der Referenzfahrt-Funktion kann keine der Treiberfunktionen den Zustand dieses Flags ändern. Selbst die Reset-Funktion kann die Zustände dieses Flags nicht verändern. Solange der Referenzfehler vorliegt, können Sie nur noch eine der folgenden Funktionen aufrufen:

- Ein- oder Ausschalten des Test-Modus (Funktion 3)
- Reglerparameteränderung (Funktion 4)
- Abfragen des Status' (Funktionen 5 und 49)
- Referenzfahrt (Funktion 6)
- Zeitschleife (Funktion 7)
- Einlesen von einem Port (Funktionen 36, 37, 40, 41, 53, 55 und 56)
- Ausgeben an einem Port (Funktionen 38, 39, 42, 48 und 54)
- Operationen mit den reservierten Datenbytes (Funktionen 44, 45 und 46)

Durch das Lesen des Treiberstatus' können Sie zwar jederzeit ermitteln, ob ein Referenzfehler vorliegt oder nicht. Sie können aber nicht bestimmen, bei welchen Achsen die Referenzfahrt noch nicht durchgeführt worden ist.

Fehlercode = 10: Handbewegung-Fehler
Die Funktion kann nicht ausgeführt werden, weil der Treiber momentan im Hand-Modus steht (siehe Kapitel 3.3.8). Falls der Hand-Modus aktiv ist, können Sie die folgenden Funktionen nicht aufrufen:

- Referenzfahrt (Funktion 6)
- Abfragen der Ist-Geschwindigkeit (Funktion 19)

Durch das Lesen des Treiberstatus' können Sie immer ermitteln, ob der Hand-Modus momentan aktiv ist oder nicht.

Fehlercode = 11: Stop-Fehler
Dieser Fehler tritt nur auf, wenn Sie die Referenzfahrt (Funktion 6) ausführen wollen, während sich der Treiber noch im Stop-Modus befindet (siehe Kapitel 3.3.10).

Fehlercode = 12: Teach-In-Fehler
Die von Ihnen verlangte Funktion kann vom Treiber nicht durchgeführt werden, weil der Teach-In-Modus momentan aktiv ist (siehe Kapitel 3.3.9).
Wenn der Treiber momentan im Teach-In-Modus steht, können Sie die folgenden Funktionen nicht aufrufen:

- Einschalten des Hand-Modus (Funktion 8)
- Ausschalten des Teach-In-Modus (Funktion 9), während ein Bewegungssegment noch aktiv im Hintergrund ist
- Start- oder Breakfunktion (Funktion 10)
- Änderung des Geschwindigkeitsfaktors (Funktion 21)
- Eilbewegungsfunktionen (Funktionen 26 und 27)
- Kreisbewegungsfunktionen (Funktionen 28 und 29)

Durch das Lesen des Treiberstatus' können Sie jederzeit ermitteln, ob der Teach-In-Modus momentan aktiv ist oder nicht.

Fehlercode = 13: Kreis-Fehler
Dieser Fehler kann nur im Zusammenhang mit einem Aufruf der Funktionen 28 oder 29 auftreten. Er besagt, dass die dem Treiber übergebenen Kreisparameter nicht korrekt sind, d. h. aus diesen Parametern kann der Treiber keinen Kreis bzw. Kreisbogen machen.

Weiterhin kann dieser Fehler auch auftreten, wenn Sie vom Treiber das Abfahren eines Kreises oder Kreisbogens verlangen, obwohl Ihre Anlage nur eine Achse oder keine TTT-Struktur hat. Wenn Ihre Anlage zwei Achsen hat, tritt der Fehler dann auf, wenn Sie einen Kreis bzw. einen Kreisbogen auf der XZ- oder YZ-Ebene wünschen.

Fehlercode = 14: Achsen-Fehler
Dieser Fehler kann nur im Zusammenhang mit dem Aufruf der Funktion 4 auftreten. Die von Ihnen dem Treiber übergebene Achsennummer ist nicht korrekt, d. h. diese Nummer liegt außerhalb des Bereichs 1 ... 4.

Fehlercode = 15: Nachlauf-Fehler
Die gewünschte Funktion kann nicht ausgeführt werden, weil ein Nachlauf-Fehler vorliegt. Während der Ausführung eines Bewegungssegments überwacht der Treiber ständig den Nachlauffehler (siehe Kapitel 3.2.6).
Falls dieser Fehler auftritt, wird ein Flag intern gesetzt. Solange dieses Flag gesetzt ist, können Sie nur noch eine der folgenden Funktionen aufrufen:

- Reset-Funktion (Funktion 2)
- Änderung der Reglerparamter (Funktion 4)
- Abfragen des Status' (Funktionen 5 und 49)
- Zeitfunktionen (Funktionen 7 und 52)
- Ein- oder Ausschalten des Hand-Modus (Funktion 8)
- Ein- oder Ausschalten des Teach-In-Modus (Funktion 9)
- Stop- und Breakfunktion (Funktion 10), falls der Teach-In-Modus eingeschaltet ist
- Einlesen von einem Port (Funktionen 36, 37, 40, 41, 53, 55 und 56)
- Ausgeben an einem Port (Funktionen 38, 39, 42, 48 und 54)
- Abfragen der Ist-Positionen (Funktion 18)
- Abfragen der Soll-Positionen (Funktionen 50 und 51)
- Abfragen der Ist-Geschwindigkeit (Funktion 19)
- Bewegungsfunktionen (Funktionen 24 und 25), falls der Teach-In-Modus eingeschaltet ist
- Operationen mit den reservierten Datenbytes (Funktionen 44, 45 und 46)

Dieses Flag können Sie nur durch die Reset-Funktion zurücksetzen.

Fehlercode = 16: Referenz-Nachlauf-Fehler
Dieser Fehler kann nur im Zusammenhang mit dem Aufruf der Referenzfahrtfunktion (Funktion 6) auftreten. Er besagt, dass ein Nachlauffehler während der Referenzfahrt auftritt.
Der Unterschied zum Nachlauf-Fehler mit dem Fehlercode 15 besteht darin, dass dieser Fehler nicht intern im Treiber, z. B. durch das Setzen eines Flags, registriert wird.

Fehlercode = 17: Geschwindigkeits-Fehler
Dieser Fehler kann nur im Zusammenhang mit dem Aufruf einer der Funktionen 20, 22, 23, 32 auftreten. Er besagt, dass Sie einen nicht-positiven Geschwindigkeitswert (≤ 0) dem Treiber übergeben wollen. Dieser Geschwindigkeitswert wird vom Treiber nicht akzeptiert, d. h. der alte Geschwindigkeitswert bleibt erhalten.

Fehlercode = 18: Helix-Fehler
Dieser Fehler kann nur im Zusammenhang mit dem Aufruf der Funktionen 30 und 31 auftreten. Er besagt, dass die dem Treiber übergebenen Parameter nicht korrekt sind, d. h. aus diesen Parametern kann der Treiber keinen Kreis bzw. Kreisbogen auf der Hauptebene nachbilden.
Es kann auch sein, dass der übergebene Bewegungswinkel negativ ist.
Weiterhin kann dieser Fehler auch auftreten, wenn Ihre Anlage nur eine Achse oder eine Kein-TTT-Struktur hat. Wenn Ihre Anlage zwei Achsen hat, tritt der Fehler auch dann auf, wenn die Kreisebene der Helixinterpolation die XZ- oder YZ-Ebene ist.

Fehlercode = 19: Bahndaten-Nachspeichern-Fehler
Dieser Fehler kann nur im Zusammenhang mit dem Bahnfahren (siehe Funktion 33) auftreten. Er besagt, dass das Anwenderprogramm während der Bahnbearbeitung die Bahndaten nicht rechtzeitig in den beiden Datenbereichen nachgeladen hat. Falls dieser Fehler auftritt, wird die Bahn sofort abgebrochen. Der Fehler wird intern im Treiber bemerkt. Solange dieses Flag aktiv ist, können Sie nur noch folgende Funktionen aufrufen:

- Reset-Funktion (Funktion 2)
- Änderung der Reglerparameter (Funktion 4)
- Abfragen des Status' (Funktionen 5 und 49)
- Zeitfunktionen (Funktionen 7 und 52)
- Ein- oder Ausschalten des Hand-Modus (Funktion 8)
- Ein- oder Ausschalten des Teach-In-Modus (Funktion 9)
- Stop- und Breakfunktion (Funktion 10), falls der Teach-In-Modus eingeschaltet ist
- Einlesen von einem Port (Funktionen 36, 37, 40, 41, 53, 55 und 56)
- Ausgeben an einem Port (Funktionen 38, 39, 42, 48 und 54)
- Abfragen der Ist-Positionen (Funktion 18)
- Abfragen der Soll-Positionen (Funktionen 50 und 51)
- Abfragen der Ist-Geschwindigkeit (Funktion 19)
- Bewegungsfunktionen (Funktionen 24 und 25), falls der Teach-In-Modus eingeschaltet ist
- Abfragen der Bahnfehlerparameter (Funktion 34)
- Operationen mit den reservierten Datenbytes (Funktionen 44, 45 und 46)

Dieses Fehlerflag lässt sich nur durch einen Reset löschen.

Fehlercode = 20: Bahn-Anlagenstruktur-Fehler
Dieser Fehler kann nur im Zusammenhang mit dem Aufruf der Funktion 33 auftreten. Es bedeutet, dass das Bahnfahren bei Ihrer Anlage nicht möglich ist, d. h. Ihre Anlage hat nur eine Achse oder eine Kein-TTT-Struktur.

Fehlercode = 21: Momentan-Kontrollebyte-Fehler
Über ein frei definiertes Eingabeport können Sie bis zu 8 hardwaremäßige Fehlersignale dem Treiber zurückführen (siehe Kapitel 3.2.6.1).
Dieser Fehler signalisiert Ihnen, dass mindestens ein Fehlersignal momentan aktiv ist. Mit der Funktion 49 können Sie jederzeit abfragen, welche Signale des Kontrollbytes aktiv sind. Der Fehler lässt sich dadurch lokalisieren. Dieser Fehler wird intern im Treiber gemerkt. Solange dieses Flag aktiv ist, können Sie nur noch folgende Funktionen aufrufen:

- Reset (Funktion 2)
- Abfragen des Status' (Funktionen 5 und 49)
- Zeitfunktionen (Funktionen 7 und 52)
- Einlesen von einem Port (Funktionen 36, 37, 40, 41, 53, 55 und 56)
- Ausgeben an einem Port (Funktionen 38, 39, 42, 48 und 54)
- Abfragen der Ist-Positionen (Funktion 18)
- Abfragen der Soll-Positionen (Funktionen 50 und 51)
- Abfragen der Ist-Geschwindigkeit (Funktion 19)
- Operationen mit den reservierten Datenbytes (Funktionen 44, 45 und 46)

Dieses Fehlerflag können Sie nur durch das Beseitigen der Fehlerquelle und danach durch die Reset-Funktion zurücksetzen.

Fehlercode = 22: Kontroll-Byte-Fehler
Über ein frei definiertes Eingabeport können Sie bis zu 8 hardwaremäßige Fehlersignale dem Treiber zurückführen (siehe Kapitel 3.2.6.1).
Dieser Fehler signalisiert Ihnen, dass mindestens ein Fehlersignal irgendwann mal aktiv war. Es gibt keine Möglichkeit, abzufragen, welches Fehlersignal aktiv war. Dieser Fehler wird intern im Treiber bemerkt. Solange dieses Flag aktiv ist, können Sie nur noch folgende Funktionen aufrufen:

- Reset (Funktion 2)
- Abfragen des Status' (Funktionen 5 und 49)
- Zeitfunktionen (Funktionen 7 und 52)
- Einlesen von einem Port (Funktionen 36, 37, 40, 41, 53, 55 und 56)
- Ausgeben an einem Port (Funktionen 38, 39, 42, 48 und 54)
- Abfragen der Ist-Positionen (Funktion 18)
- Abfragen der Soll-Positionen (Funktionen 50 und 51)
- Abfragen der Ist-Geschwindigkeit (Funktion 19)
- Operationen mit den reservierten Datenbytes (Funktionen 44, 45 und 46)

Dieses Fehlerflag können Sie nur durch die Reset-Funktion zurücksetzen.

Fehlercode = 23: Bitnummer-Fehler
Dieser Fehler kann nur beim Aufruf einer der Funktionen 36, 38 und 40 auftreten. Er bedeutet, dass die Bitnummer fehlerhaft ist (siehe Kapitel 3.3.36, 3.3.38 und 3.3.40).

Fehlercode = 24: Portnummer-Fehler
Dieser Fehler kann nur beim Aufruf der Funktionen 37, 39 und 41 auftreten. Er bedeutet, dass die Portnummer fehlerhaft ist (siehe Kapitel 3.3.37, 3.3.39 und 3.3.41).

Fehlercode = 25: In-Bewegung-Ausgabeport-Fehler
Dieser Fehler kann nur beim Aufruf der Funktion 43 auftreten. Er bedeutet, dass die Benutzung der 8-Bit-Ausgabe während der Bewegung nicht zugelassen ist (siehe Kapitel 3.2. ... und 3.3.40). Die Ausgabe während der Bewegung ist optional. Die Einstellung erfolgt mit dem Konfigurationsprogramm PARKON.EXE.

Fehlercode = 26: Geheimcode-Fehler
Dieser Fehler kann nur im Zusammenhang mit dem Aufruf einer der Funktionen 44, 45 und 46 auftreten. Er bedeutet, dass Sie auf ein Reserviert-Byte über einen fehlerhaften Geheimcode zugreifen wollten.

Fehlercode = 27: Sicherheitskreis-Überbrückung-Fehler
Falls einer der Hardware-Endschalter aktiv ist, muss die Funktion 48 benutzt werden, um den Servocontroller einschalten zu können (siehe Kapitel 3.2.6.2). Das Gleiche passiert auch bei der Referenzfahrt (siehe Kapitel 3.2.7). Aber solange der Sicherheitskreis des Servocontrollers überbrückt ist, ist die Sicherheit der Steuerung nicht gewährleistet. In diesem Zustand werden aufgrund der Sicherheit nur noch folgende Funktionen ausführbar:

- Reset-Funktion (Funktion 2)
- Änderung der Reglerparameter (Funktion 4)
- Abfragen des Status' (Funktionen 5 und 49)
- Zeitfunktionen (Funktionen 7 und 52)
- Ein- oder Ausschalten des Hand-Modus (Funktion 8)
- Ein- oder Ausschalten des Teach-In-Modus (Funktion 9)
- Breakfunktion (Funktion 10)
- Einlesen von einem Port (Funktionen 36, 37, 40, 41, 53, 55 und 56)
- Ausgeben an einem Port (Funktionen 38, 39, 42, 48 und 54)
- Abfragen der Ist-Positionen (Funktion 18)
- Abfragen der Ist-Geschwindigkeit (Funktion 19)
- Abfragen der Soll-Positionen (Funktionen 50 und 51)
- Relative oder absolute Linearnormalbewegung (Funktionen 24 und 25) im Teach-In-Modus
- Operationen mit den reservierten Datenbytes (Funktionen 44, 45 und 46)
- Setzen den Sicherheitskreis-Ausgabeport (Funktion 48)

Der Überbrückungszustand wird intern in einem Flag gespeichert.
Das Rücksetzen dieses Flags ist nur durch die Funktion 48 möglich.
Die Reset-Funktion hat keinen Einfluss auf dieses Flag.

Fehlercode = 28: Referenzfahrt-Unterbrechung-Fehler
Während der Referenzfahrt wird die ganze Rechenzeit des Rechners von der Steuerung beansprucht. Eine Unterbrechung der Referenzfahrt ist nur durch eine von Ihnen selbst definierte Taste möglich. Die Festlegung der Unterbrechungstaste erfolgt durch das Programm PARKON.EXE. Mit dem Programm PAREIN.EXE können Sie den Code der Unterbrechungstaste ermitteln.
Dieser Fehler entsteht nur im Zusammenhang mit dem Aufruf der Funktion 6 (siehe Kapitel 3.3.6) und wird intern nicht abgespeichert.

Fehlercode = 29: Anlagenstruktur-Fehler
Bei der Achsenumschaltung (Funktion 57) und bei dem Setzen von Zylinderradius einer Drehachse (Funktion 58) müssen Sie die neue Anlagenstruktur angeben. Beim Aufruf dieser beiden Funktionen kontrolliert der Treiber, ob die neue Anlagenstruktur zu den Achsenarten (Linear- oder Drehachse) passt oder nicht. Dieser Fehler wird nicht intern abgespeichert.

Fehlercode = 30: Zylinderradius-Fehler
Dieser Fehler entsteht nur beim Aufruf der Funktion 58. Bei der Umwandlung einer Drehachse zu einer Linearachse zwecks Zylindergravierens müssen Sie den Zylinderradius definieren.
Der Zylinderradius darf keinen negativen Wert haben. Ein Wert gleich Null wird nur bei der Umwandlung zurück in eine Drehachse akzeptiert. Dieser Fehler wird intern nicht abgespeichert.

Fehlercode = 31: Spindel-Modus-Fehler
Falls die letzte Achse noch nicht im Spindel-Modus steht, können Sie folgende Funktionen nicht benutzen:

- Festlegen des Benutzungsortes der Spindelachse (Funktion 60)
- Definieren des Änderungsfaktors der Spindelgeschwindigkeit (Funktion 61)
- Änderung der Geschwindigkeit der Spindelachse (Funktion 62)
- Änderung der absoluten Position der Spindelachse (Funktion 63)
- Einschalten des Hand-Modus der Spindelachse (Funktion 64)
- Abfragen der Bewegungsparameter der Spindelachse (Funktion 66)

Falls die letzte Achse im Spindel-Modus steht, können Sie die folgende Funktion nicht aufrufen:

- Umschaltung der Achsen (Funktion 57)

Fehlercode = 32: Spindelachse-in-Bewegung-Fehler
Falls die Spindelachse in Bewegung ist, können Sie folgende Funktionen nicht ausführen:

- Ausschalten des Spindel-Modus (Funktion 59)

- Festlegen des Benutzungsortes der Spindelachse (Funktion 60)
- Änderung der Spindelgeschwindigkeit (Funktion 62), falls sich die Spindelachse im Positions-Modus befindet
- Änderung der absoluten Position der Spindelachse (Funktion 63)
- Einschalten des Hand-Modus der Spindelachse (Funktion 64)

Fehlercode = 33: Spindel-Benutzungsort-Fehler
Falls Sie mit Hilfe der Funktion 60 die Spindelachse für die Benutzung in der Bahn festgelegt haben, können Sie folgende Funktionen nicht mehr benutzen:

- Änderung der Spindelgeschwindigkeit (Funktion 62).
  Die Spindelgeschwindigkeit lässt sich nur noch durch einen entsprechenden Befehl in der Bahn ändern (siehe Anhang B).
- Änderung der absoluten Position der Spindelachse (Funktion 63)
- Einschalten des Hand-Modus der Spindelachse (Funktion 64)

Fehlercode = 34: Spindel-Handbewegung-Fehler
Falls der Hand-Modus der Spindelachse aktiv ist, sind folgende Funktionen nicht verfügbar:

- Bahnbewegung (Funktion 33)
- Änderung der Geschwindigkeit der Spindelachse (Funktion 62)
- Änderung der absoluten Position der Spindelachse (Funktion 63)

Fehlercode = 35: Handrad-Modus-Fehler
Falls die Achsen im Handrad-Modus stehen, können Sie folgende Funktionen nicht benutzen:

- Einschalten des Hand-Modus (Funktion 8)
- Abfragen der Soll-Positionen (Funktion 50 und Funktion 51)
- Bewegungsfunktionen (Funktion 25, 27, ... 31 und 33)

Falls die Achsen nicht im Handrad-Modus stehen, können Sie die folgende Funktion nicht aufrufen:

- Anhalten der Handrad-Bewegung (Funktion 68)

Fehlercode = 36: Handrad-Bewegung-Fehler
Falls die im Handrad-Modus stehenden Achsen noch in Bewegung sind, können Sie den Handrad-Modus nicht ausschalten (Funktion 67).

Fehlercode = 37: Encoder-Indexsignal-Fehler
Der Treiber ist mit dem Optionsschalter /IR gestartet. Hardwaremäßig ist das Indexsignal von mindestens einem Encoder nicht vorhanden. Beim Aufruf der Funktion 6, um die Referenzfahrt auszuführen, meldet der Treiber mit diesem Fehlercode zurück.

3.3 Treiberfunktionen

3.3.1 Funktion 1: Abfragen der Treiberversion

Zweck	Abfragen der Treiberversion	
Aufrufparameter	ax = 1	Funktionsnummer
	bx, cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bh; bl	In diesem Register stehen
	ch; cl	6 ASCII-Zeichen für die
	dh; dl	Treiberversion

Kommentar:
Aufgrund der ständigen Verbesserung sowie Erweiterung hat jedes Softwareprodukt bis zu seinem Lebensende verschiedene Versionen. Mit Hilfe dieser Funktion können Sie die momentan von Ihnen benutzte Treiberversion abfragen. In jedem der Register bh, bl, ch, cl, dh und dl steht ein ASCII-Zeichen. Wenn Sie diese 6 ASCII-Zeichen in der Reihenfolge bh, bl, ch, cl, dh, dl zusammensetzen, bekommen Sie die Versionsnummer, z. B. wenn die Versionsnummer 1.00 ist, ergibt sich die folgende Zuordnung:

bh = 49	(ASCII-Zeichen: 1)
bl = 46	(ASCII-Zeichen: .)
ch = 48	(ASCII-Zeichen: 0)
cl = 48	(ASCII-Zeichen: 0)
dh = 0	(ASCII-Zeichen: NUL)
dl = 0	(ASCII-Zeichen: NUL)

Falls die Zeichenanzahl der Versionsnummer kleiner als 6 ist, haben die restlichen Prozessorregister den Wert 0 (ASCII-Zeichen: NUL).

Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 2
Siehe auch: Kapitel 3.2.17

3.3.2 Funktion 2: Reset

Zweck	Treiber-Reset	
Aufrufparameter	ax = 2	Funktionsnummer
	bx, cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Diese Funktion setzt den Treiber auf seinen Anfangszustand - wie bei einem Neustart.

Falls sich die Anlage momentan in einer Bewegung befindet, wird diese Bewegung sofort unterbrochen. Die Teach-In-, die Bahn- und die Eilgeschwindigkeit werden auf ihre Standardwerte zurückgesetzt. Der Änderungsfaktor Kv für die Werkzeuggeschwindigkeit wird gleich 100 (=100 %) gesetzt (siehe Kapitel 3.3.21). Die Software-Endschalter nehmen ebenfalls ihre Standardwerte an, werden aber gesperrt. Ein evtl. vorhandener Software-Endschalter-Fehler wird natürlich gelöscht. Der Referenzpunkt wird als Werkstück-Nullpunkt definiert. Der Treiber wird in die Normal-Betriebsart zurückgeschaltet, falls er sich vorher im Teach-In-Modus, im Test-Modus oder im Hand-Modus befand. Außerdem werden alle Achsregler mit den Standardwerten ihrer Parameter initialisiert.
Das Break-Flag und das In-Bewegung-Hardware-Endschalter-Status-Flag, die einen Hardware-Endschalter-Fehler während der Bearbeitung zeigen, werden wieder auf 0 zurückgesetzt (siehe Kapitel 3.3.5).

Es gibt einige Dinge, auf die das Reset keinen Einfluss hat. Das sind die Referenzflags der Achsen, die momentanen Zustände der Hardware-Endschalter, die Werte der über logische Nummern ansprechbaren Ausgabeports und die Inhalte der im Treiber für Anwender reservierten Datenbytes.
Wenn Sie die Referenzfahrt einmal durchgeführt haben, gilt auch nach dem Reset, dass die Referenzfahrt schon ausgeführt worden ist. Sie brauchen die Referenzfahrt nicht noch einmal auszuführen. Bei den Hardware-Endschaltern gilt, dass sich ihre momentanen Zustände durch die Reset-Funktion nicht ändern, d. h. direkt vor und nach dem Reset sind die Zustände der Hardware-Endschalter identisch. Durch einen Aufruf der Funktion 5 können Sie sie jederzeit abfragen.

Die Werte, die an den über logischen Nummern ansprechbaren Ausgabeports und Überwachungsports liegen, ändern sich beim Reset nicht (siehe Kapitel 3.3.38 bis 3.3.41 und 3.3.55). Durch einen Aufruf der Funktion 42 bzw. der Funktion 56 können Sie aber jederzeit diese Ausgabeports auf ihren über das Konfigurationsprogramm einstellbaren Initialisierungswerten zurücksetzen (siehe Kapitel 3.3.42, 3.3.56). Die Inhalte der Datenbytes, die Sie im Treiber für Ihre eigenen Zwecke reservieren, bleiben bei einem Aufruf der Reset-Funktion auch unverändert (siehe Kapitel 3.3.44 bis 3.3.46).
Das Überbrücken des Sicherheitskreis (siehe Kapitel 3.3.48) wird von der Reset-Funktion auch nicht beeinflusst.
Wir empfehlen Ihnen, diese Funktion aufzurufen, bevor Sie Ihr Anwender-programm beenden. Dadurch setzen Sie den Treiber in seinen originalen Zustand, in dem er sich nach einem Neustart befindet.

Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 2
Siehe auch: Kapitel 3.2.17; 3.3.5

3.3.3 Funktion 3: Ein- oder Ausschalten des Test-Modus

Zweck	Ein- oder Ausschalten den Test-Modus	
Aufrufparameter	ax = 3	Funktionsnummer
	bx	Unterfunktionen
	cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
In der Phase der Inbetriebnahme kommen sehr häufig Fehler vor, die nicht durch Ihre Programme selbst sondern durch falsche Konfiguration auftreten, wie z. B. falsche Adressen oder falsche Aktivpegel der Schalter. Deswegen ist es sehr angenehm für den Anwender, wenn er die Möglichkeit hat, die Abfrage der Hardware-Endschalter ausschalten zu können. Außerdem ist es sicher wünschenswert, dass die meist sehr langwierige Referenzfahrt nicht immer wiederholt werden muss. Aufgrund dieser Wünsche haben wir den sogenannten Test-Modus eingeführt.
In Abhängigkeit vom Wert im Register bx können Sie den Test-Modus ein- oder ausschalten.
Zuordnung zwischen dem Wert in bx und den einzelnen Unterfunktionen:

bx	Unterfunktionen
0	Ausschalten den Test-Modus
1	Einschalten den Test-Modus
sonst	Einschalten den Test-Modus

Im Test-Modus behandelt der Treiber die Referenzfahrt und die Hardware-Endschalter anders als im normalen Betrieb. Alle anderen Treiberfunktionen können Sie aber weiterhin wie im Normalbetrieb benutzen.
Wenn Sie die Funktion 6 für das Ausführen der Referenzfahrt im Test-Modus aufrufen, führt der Treiber keine Referenzfahrt im eigentlichen Sinne aus, sondern er setzt den Punkt als Referenzpunkt, in dem die Achse direkt vor dem Aufruf steht. Danach kehrt der Treiber wieder zu Ihrem Programm zurück. Beachten Sie, dass Sie trotz des eingeschalteten Test-Modus die Referenzfahrtfunktion aufrufen müssen, da Sie sonst den Referenzfehler beim Aufruf vieler Treiberfunktionen bekommen (siehe Kapitel 3.2.10).
Im Test-Modus überwacht der Treiber weiterhin die Hardware-Endschalter, d. h. Sie können jederzeit die Zustände der Hardware-Endschalter durch einen Aufruf der Funktion 5 abfragen.

Der Unterschied zum Normalbetrieb besteht darin, dass der Treiber im Test-Modus eine Bewegung nicht unterbricht, falls ein Hardware-Endschalter-Fehler auftritt. Die Anlage fährt weiter, als ob der Fehler gar nicht da wäre, obwohl der Treiber die momentanen Zustände dieser Schalter weiterhin überwacht.
Der Nachlauffehler wird im Test-Modus ganz normal überwacht. Ein evtl. auftretender Nachlauffehler unterbricht die aktuelle Bewegung aber nicht.

Wie bei Hardware-Endschalter-Fehlern können Sie jederzeit den Zustand des Nachlauffehlers durch einen Aufruf der Funktion 5 abfragen.

Ausführbarkeit während der Bewegung: Nein
Mögliche Fehlercodes: 0; 2; 3; 4; 8; 15; 19
Siehe auch: Kapitel 3.2.17

3.3.4 Funktion 4: Änderung der Reglerparameter

Zweck	Änderung der Reglerparameter einer Achse	
Aufrufparameter	ax = 4	Funktionsnummer
	bx	Segmentadresse der Reglerparameter
	cx	Offsetadresse der Reglerparameter
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Mit dieser Funktion sind Sie in der Lage, die Achsreglerparameter Ihrer Anlage zu verändern. Pro Aufruf können Sie sie aber nur bei einer Achse ändern. Die Information darüber, an welcher Achse die Änderung stattfinden soll sowie die neuen Reglerparameter müssen Sie in einem Speicherbereich ablegen, dessen Anfangsadresse beim Aufruf dieser Funktion in bx:cx steht.
Bild 3.18 zeigt Ihnen die Speicherzuordnung für jeden Parameter. Aus Platzgründen haben wir im Bild 3.18 diesen Speicherbereich in zwei Teile geteilt. In Wirklichkeit gehören diese beiden Teile zusammen.

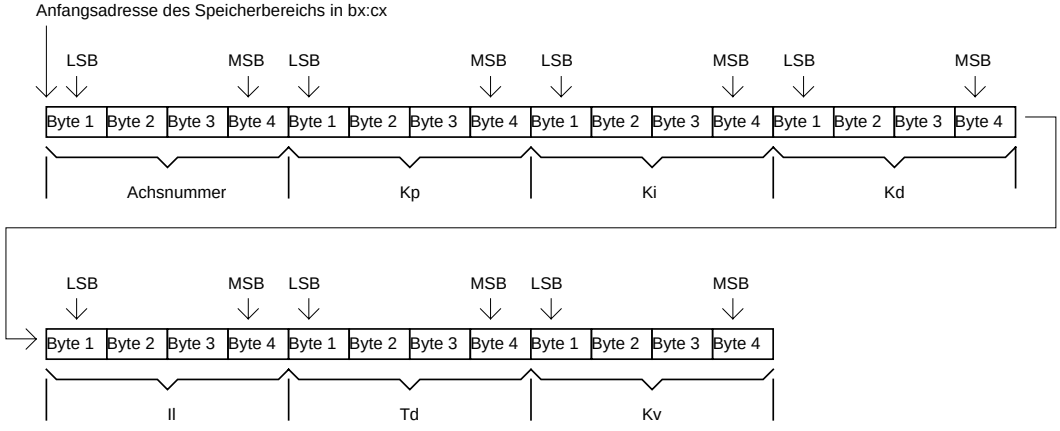


Bild 3.18: Belegung des Speichers für die Achsreglerparameter

Jeder Parameter, den Sie dem Treiber übergeben wollen, belegt genau 4 Byte im Speicher und wird als eine ganze Zahl in Zweier-Komplementär-Darstellung interpretiert. Aus den ersten 4 Bytes entnimmt der Treiber, bei welcher Achse die Änderung vorgenommen werden soll.

Zuordnung zwischen dem Wert in diesen 4 Bytes und der Anlagenachse:

Achsnummer	Anlagenachse
1	X-Achse
2	Y-Achse
3	Z-Achse
4	A-Achse.

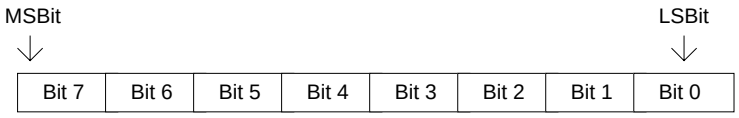
In den folgenden Bytes liegen die neuen Werte für Kp, Ki, Kd, Il, Td und Kv. Jeder der Parameter Kp, Ki, Kd und Il werden intern vom Treiber auf den Wertebereich von 0 ... 32767 (0 ... 2^{15}-1) begrenzt. Wenn einer der Parameter kleiner als 0 ist, wird er gleich 0 gesetzt und wenn er größer als 32767 ist, wird er gleich 32767 gesetzt. Ähnlich wie diese Parameter wird Td auch begrenzt. Der Wertebereich für Td liegt aber zwischen 0 ... 255 (0 ... 2^{8}-1).
Der Wertebereich von Kv liegt zwischen 1 ... 10^{7} und ist damit wesentlich größer als der Wertebereich der anderen Parameter. Diese Begrenzungen sind notwendig, weil die Parameter der Achsregler technisch bedingt nur in den genannten Wertebereichen liegen dürfen (siehe Kapitel 3.2.6). Nachdem der Treiber die neuen Werte zum Achscontroller der gewünschten Achse geschickt hat, kehrt er zum Anwenderprogramm zurück.

Ausführbarkeit während der Bewegung: Nein
Mögliche Fehlercodes: 0; 2; 4; 14
Siehe auch: Kapitel 3.2.11, 3.2.13, 3.2.15, 3.2.17

3.3.5 Funktion 5: Abfragen des Treiberstatus'

Zweck	Abfragen des momentanen Status des Treibers	
Aufrufparameter	ax = 5	Funktionsnummer
	bx, cx, dx	undefiniert
Ergebnis	al	Fehlercode
	ah	Allgemeiner Status des Treibers
	bx	Fehlerstatus
	cx	Status des Hardware-Endschalters
	dx	Status des Software-Endschalters

Kommentar:
Dies ist die einzige Funktion, bei der der Treiber alle Register ax, bx, cx und dx für die Datenübergabe nutzt. Der Fehlercode des Aufrufs dieser Funktion steht wie immer in al. In den restlichen Registern können Sie als Anwender die einzelnen Bits bewerten, um den momentanen Zustand des Treibers zu erkennen. Das Bild 3.19 soll Ihnen zeigen, wie wir die Bits eines Bytes nummerieren.



LSBit = Last Significant Bit (Niederwertigstes Bit)
MSBit = Most Significant Bit (Höchstwertigstes Bit)

Bild 3.19: Nummerierung der Bits in einem Byte

Der allgemeine Status des Treibers steht in ah. Dieses Statusbyte soll Ihnen mitteilen, ob ein Segment momentan abgefahren wird und ob der Treiber im Stop-Modus, im Teach-In-Modus, im Test-Modus bzw. im Hand-Modus steht. Die Bedeutung der einzelnen Bits wird Im Folgenden beschrieben.

Register ah:

Bit 0: In-Segment-Bewegung-Status
- = 0 Momentan ist kein Bewegungssegment im Hintergrund aktiv.
- = 1 Momentan ist ein Bewegungssegment im Hintergrund aktiv.

Bit 1: Stop-Status
- = 0 Momentan ist der Stop-Modus nicht aktiv.
- = 1 Momentan befindet sich der Treiber im Stop-Modus. (siehe Kapitel 3.3.10)

Bit 2: Teach-In-Status
- = 0 Momentan steht der Treiber nicht im Teach-In-Modus.
- = 1 Momentan steht der Treiber im Teach-In-Modus. (siehe Kapitel 3.3.9)

Bit 3: Test-Status
- = 0 Momentan steht der Treiber nicht im Test-Modus.
- = 1 Momentan steht der Treiber im Test-Modus. (siehe Kapitel 3.3.3)

Bit 4: Hand-Status
- = 0 Momentan steht der Treiber nicht im Hand-Modus.
- = 1 Momentan steht der Treiber im Hand-Modus. (siehe Kapitel 3.3.8)

Bit 5: Bahnfahren-Status
- = 0 Momentan keine 3D-Bahnbewegung im Hintergrund ausgeführt.
- = 1 Momentan wird 3D-Bahnbewegung im Hintergrund ausgeführt. (siehe Kapitel 3.3.33)

Bit 6: In-Bewegung-Status
- = 0 Momentan befindet sich die Anlage in Ruhe.
- = 1 Momentan befindet sich die Anlage in Bewegung.

Bit 7: Sicherheitskreis-Überbrückung-Status
= 0 Momentan ist der Sicherheitskreis nicht überbrückt.
= 1 Momentan ist der Sicherheitskreis überbrückt.

Mit der Funktion 48 können Sie auf dem PIN 43 des RIBBON-Steckers einen LOW- oder HIGH-Wert ausgeben. Ein LOW-Wert am PIN 43 bedeutet, dass der Sicherheitskreis nicht überbrückt ist und ein HIGH-Wert am PIN 43 bedeutet, dass der Sicherheitskreis überbrückt ist.

Beachten Sie, dass Bit 0 und Bit 6 unterschiedliche Bedeutungen haben.
Bit 0 sagt nur aus, ob ein Segment momentan im Hintergrund des Treibers bearbeitet wird. Es ist durchaus möglich, dass die Anlage sich nicht bewegt, während ein Segment im Hintergrund aktiv ist. So etwas ist möglich, wenn Sie z. B. während einer Segmentbearbeitung die Stop-Funktion aufrufen (siehe Kapitel 3.3.10) oder wenn Sie den Änderungsfaktor der Werkzeug-geschwindigkeit gleich 0 setzen (siehe Kapitel 3.3.21).
Über das Bit 0 können Sie feststellen, ob eine Bearbeitung momentan im Hintergrund aktiv ist. Falls nicht, können Sie eine weitere Bearbeitung zum Treiber delegieren. Es gilt sowohl für eine Segment- als auch für eine Bahnbearbeitung. Vor jedem Aufruf einer der Bewegungsfunktionen (siehe Kapitel 3.3.24 bis 3.3.31 und 3.3.33) müssen Sie dieses Bit kontrollieren.
Über das Bit 6 können Sie kontrollieren, ob sich die Anlage momentan bewegt. Diese Kontrolle ist notwendig, wenn Sie z. B. nach einem Aufruf der Stop-Funktion während der Bearbeitung den Teach-In-Modus einschalten wollen (siehe Kapitel 3.3.9). Der Teach-In-Modus lässt sich nur aktivieren, wenn sich die Anlage nicht bewegt. Das Bit 6 ist immer gleich 0, falls keine Bearbeitung im Hintergrund aktiv ist, d. h. falls das Bit 0 gleich 0 ist.

Der Fehlerstatus des Treibers steht in bx. Durch diesen Fehlerstatus können Sie erkennen, welche Fehler der Treiber momentan hat. Die Bedeutung der einzelnen Bits wird Im Folgenden erläutert.

Register bl:
Bit 0: Referenzfahrt-Status
= 0 Bei allen Achsen wurde die Referenzfahrt bereits durchgeführt.
= 1 Bei mindestens einer der Achsen wurde die Referenzfahrt noch nicht durchgeführt. Solange dieses Bit gesetzt ist, können Sie viele Treiberfunktionen nicht aufrufen.
(siehe Kapitel 3.3.6)

Bit 1: Break-Status
= 0 Die Break-Funktion wurde seit dem Start bzw. dem letzten Reset des Treibers noch nicht aufgerufen.
= 1 Die Break-Funktion wurde schon einmal aufgerufen. Bei einem Aufruf dieser Unterfunktion, die man als eine Not-Aus-Funktion verstehen kann, wird dieses Bit gleich 1 gesetzt. Solange dieses Bit gesetzt ist, stehen die meisten Treiberfunktionen nicht zur

Verfügung. Nur durch Reset können Sie dieses Bit auf 0 zurücksetzen und damit die gesperrten Funktionen wieder zulassen. (siehe Kapitel 3.3.10)

Bit 2: Hardware-Endschalter-Status

= 0 Momentan ist kein der Hardware-Endschalter aktiv.

= 1 Momentan ist mindestens ein Hardware-Endschalter aktiv.

Dieses Bit zeigt Ihnen den momentanen Zustand der Hardware-Endschalter an.
Um genau zu wissen, welche Hardware-Endschalter momentan aktiv sind, müssen Sie das Register cx bewerten. Während des aktiven Zustands eines der Hardware-Endschalter sind viele Treiberfunktionen gesperrt.

Bit 3: Software-Endschalter-Status

= 0 Momentan ist kein Software-Endschalter aktiv.

= 1 Momentan ist mindestens ein Software-Endschalter aktiv.

Nachdem Sie die Software-Endschalter gesetzt und freigegeben haben, überwacht der Treiber ständig die Software-Endschalter und dieses Bit zeigt Ihnen den momentanen Zustand der Software-Endschalter an.
Um genau zu wissen, welche Software-Endschalter momentan aktiv sind, müssen Sie das Register dx bewerten.
(siehe Kapitel 3.3.14 und 3.3.15)

Bit 4: In-Bewegung-Hardware-Endschalter-Status

= 0 Während der Bearbeitung war bzw. ist noch nie ein Hardware-Endschalter aktiv geworden.

= 1 Während der Bearbeitung war bzw. ist mindestens ein Hardware-Endschalter aktiv geworden. Während der Abarbeitung einer der Bewegungsfunktionen im Hintergrund überwacht der Treiber ständig die Hardware-Endschalter. Falls einer dieser Hardware-Endschalter aktiv wird, wird die Bearbeitung sofort unterbrochen und dieses Bit gleich 1 gesetzt. Dadurch werden viele Treiberfunktionen gesperrt. Nur durch ein Reset können Sie dieses Bit zurück auf 0 setzen. Sie müssen zwei Besonderheiten dieses Bits beachten. Der Treiber überwacht im Teach-In-Modus während der Abarbeitung einer der Bewegungsfunktion weiterhin die Hardware-Endschalter und unterbricht die Bewegung sofort, falls einer der Hardware-Endschalter aktiv wird. Das Bit wird in diesem Fall aber nicht auf 1 gesetzt, nur während der Bearbeitung, nicht aber während eines Teach-In-Fahrens, wird dieses Bit gleich 1 gesetzt, falls einer der Hardware-Endschalter aktiv wird.
Die zweite Besonderheit dieses Bits ergibt sich im Test-Modus (siehe Kapitel 3.3.3).

Beim Abarbeiten einer der Bewegungsfunktionen im Test-Modus werden die Hardware-Endschalter wie normal überwacht. Aber beim Aktiv werden einer der Hardware-Endschalter wird die Abarbeitung nicht unterbrochen und dieses Bit nicht auf 1 gesetzt. Die momentanen Zustände der Hardware-Endschalter können Sie im Test-Modus weiterhin durch einen Aufruf dieser Funktion und durch das anschließende Bewerten des Registers cx erkennen.

Bit 5: Nachlauffehler-Status

= 0 Seit dem Laden bzw. dem letztem Reset des Treibers haben die Nachlauffehler von allen Achsen während der Abbearbeitung der Bewegungsfunktionen noch nie die zulässigen Grenzen überschritten.

= 1 Der Nachlauffehler von mindestens einer Achse hat während der Abarbeitung einer Bewegungsfunktion die zulässige Grenze überschritten. Während der Abarbeitung der Bewegungs-funktionen überwacht der Treiber ständig die Nachlauffehler von allen Achsen. Falls der Nachlauffehler von einer beliebigen Anlagenachse die zulässige Grenze überschreitet, wird die Bearbeitung sofort unterbrochen und dieses Bit gleich 1 gesetzt. Dadurch werden viele Treiberfunktionen gesperrt. Nur durch ein Reset können Sie dieses Bit auf 0 zurücksetzen.
Sie müssen aber beachten, dass der Treiber im Teach-In-Modus die Nachlauffehler zwar weiterhin überwacht und eine Bewegung sofort unterbricht, falls die zulässige Grenze bei einer Achse überschritten wurde. Dieses Bit wird hier aber nicht auf 1 gesetzt, d. h. nur während der Bearbeitung, nicht aber während eines Teach-In-Fahrens, wird dieses Bit gleich 1 gesetzt, falls die zulässigen Grenzen überschritten wurden.

Bit 6: Bahndaten-Nachspeichern-Fehler-Status

= 0 Seit dem Laden bzw. dem letzten Reset des Treibers ist noch nie ein Nachspeichern-Fehler während der 3D-Bahnbearbeitung aufgetreten.

= 1 Während einer 3D-Bahnbearbeitung liest der Treiber die Bahndaten aus zwei im Anwenderprogramm reservierten Speicherbereichen. Das Anwenderprogramm hat die Aufgabe, die Bahndaten aus einer Datei zu lesen und diese beiden Speicherbereiche rechtzeitig nachzuladen.
Falls die Bahndaten aus irgendeinem Grund nicht rechtzeitig nachgeladen werden können, tritt der so genannte Nachspeichern-Fehler auf. Die Bahnbewegung wird sofort unterbrochen, egal ob der Test-Modus momentan aktiv ist oder nicht. Dieses Bit wird gleich 1 gesetzt. Das Aktivieren des Test-Modus kann diesen Fehler nicht ignorieren. Dadurch bleiben

viele Treiberfunktionen gesperrt. Nur durch ein Reset können Sie dieses Bit auf 0 zurücksetzen (siehe Kapitel 3.3.33).

Bit 7: Momentan-Kontrollbyte-Status

= 0 Momentan sind keine Hardware-Fehlersignale am Kontrollbyte festzustellen.

= 1 Momentan ist mindestens ein Hardware-Fehlersignal am Kontrollbyte aktiv. Über das frei definierte Kontrollbyte können Sie bis zu 8 Hardware-Fehlersignale zurückführen. Der Treiber vergleicht ständig den aktuellen Wert dieses Eingabeports mit einem von Ihnen als fehlerfrei definierten Wert.
Eine Abweichung zwischen diesen beiden Werten interpretiert der Treiber als ein Hardwarefehler. Mit der Funktion 49 können Sie die momentane Hardware-Fehlersignale lokalisieren.

Register bh:

Bit 0: Kontrollbyte-Fehler-Status

= 0 Seit dem Laden bzw. seit dem letzten Reset des Treibers waren die Hardware-Fehlersignale am Kontrollbyte noch nie aktiv.

= 1 Mindestens ein der Hardware-Fehlersignale war sei dem Laden bzw. seit dem letzten Reset des Treibers aktiv oder mindestens ein der Hardware-Fehlersignale ist momentan aktiv.
Der Unterschied zwischen diesem Bit und Bit 7 des Registers bl besteht darin, dass Bit 7 den momentanen Fehlerzustand des Kontrollbyte zeigt und dieses Bit den Fehlerzustand des Kontrollbytes abspeichert. Es ist durchaus möglich, dass dieses Bit aktiv ist während das Bit 7 einen fehlerfreien Momentanzustand anzeigt, falls einer der Hardware-Fehlersignale irgendwann mal aktiv ist. Es ist nicht möglich, den Treiber abzufragen, welches der Hardware-Fehlersignale aktiv war.
Mit Hilfe der Funktion 49 können Sie nur die momentanen Zustände der Hardware-Fehlersignale abfragen.

Bit 1 ... 7: Reservierte Bits

= 0 Diese Bits sind momentan noch nicht benutzt und immer = 0.

Um genau zu wissen, welche Hardware-Endschalter momentan aktiv sind und welche nicht, müssen Sie das Register cx bewerten. Dabei ist das Register cl zuständig für alle negativen Hardware-Endschalter und ch für alle positiven. Jeder Achse werden 2 Bits (eines in cl und eines in ch) zugeordnet. Wenn ein Bit gleich 0 ist, ist der entsprechende Hardware-Endschalter momentan nicht aktiv. Wenn ein Bit gleich 1 ist, ist der entsprechende Hardware-Endschalter momentan aktiv. Durch das mitgelieferte Konfigurationsprogramm können Sie die Anzahl der Achsen für Ihre Anlage festlegen. Weiterhin können Sie auch selbst bestimmen, ob keiner oder nur einer oder ob beide Hardware-Endschalter einer Achsen benutzt werden sollen (siehe Kapitel 4.3). Die Bits für nicht benutzte Hardware-Endschalter sowie für Hardware-Endschalter der nicht

benutzten Achsen sind immer gleich 0, d. h. diese Hardware-Endschalter sind für ein Anwenderprogramm nicht aktiv.

Register cl:

Bit 0: Zuständig für den negativen Hardware-Endschalter der X-Achse
- = 0 Der Hardware-Endschalter ist momentan nicht aktiv.
- = 1 Der Hardware-Endschalter ist momentan aktiv.

Bit 1: Zuständig für den negativen Hardware-Endschalter der Y-Achse
- = 0 Der Hardware-Endschalter ist momentan nicht aktiv.
- = 1 Der Hardware-Endschalter ist momentan aktiv.

Bit 2: Zuständig für den negativen Hardware-Endschalter der Z-Achse
- = 0 Der Hardware-Endschalter ist momentan nicht aktiv.
- = 1 Der Hardware-Endschalter ist momentan aktiv.

Bit 3: Zuständig für den negativen Hardware-Endschalter der A-Achse
- = 0 Der Hardware-Endschalter ist momentan nicht aktiv.
- = 1 Der Hardware-Endschalter ist momentan aktiv.

Bit 4 ... 7: Reservierte Bits
- = 0 Diese Bits sind momentan noch nicht benutzt und immer gleich 0.

Register ch:

Bit 0: Zuständig für den positiven Hardware-Endschalter der X-Achse
- = 0 Der Hardware-Endschalter ist momentan nicht aktiv.
- = 1 Der Hardware-Endschalter ist momentan aktiv.

Bit 1: Zuständig für den positiven Hardware-Endschalter der Y-Achse
- = 0 Der Hardware-Endschalter ist momentan nicht aktiv.
- = 1 Der Hardware-Endschalter ist momentan aktiv.

Bit 2: Zuständig für den positiven Hardware-Endschalter der Z-Achse
- = 0 Der Hardware-Endschalter ist momentan nicht aktiv.
- = 1 Der Hardware-Endschalter ist momentan aktiv.

Bit 3: Zuständig für den positiven Hardware-Endschalter der A-Achse
- = 0 Der Hardware-Endschalter ist momentan nicht aktiv.
- = 1 Der Hardware-Endschalter ist momentan aktiv.

Bit 4 ... 7: Reservierte Bits
- = 0 Diese Bits sind momentan noch nicht benutzt und immer gleich 0 gesetzt.

Durch das Bewerten des Registers dx können Sie genau wissen, welche Software-Endschalter momentan aktiv sind und welche nicht. Ähnlich wie bei den Hardware-Endschaltern ist je ein Bit des Registers dl zuständig für einen negativen Software-Endschalter und je ein Bit des Registers dh zuständig für einen positiven. Die Bits für die Software-Endschalter einer nicht benutzten Achse erscheinen einem Anwenderprogramm immer als nicht aktiv. Falls die Benutzung der Software-Endschalter nicht zugelassen ist, sind alle Bits gleich 0 (siehe Kapitel 3.3.15).
Die Bedeutung der einzelnen Bits von dh und dl:

Register dl:

Bit 0: Zuständig für den negativen Software-Endschalter der X-Achse
- = 0 Der Software-Endschalter ist momentan nicht aktiv.
- = 1 Der Software-Endschalter ist momentan aktiv.

Bit 1: Zuständig für den negativen Software-Endschalter der Y-Achse
- = 0 Der Software-Endschalter ist momentan nicht aktiv.
- = 1 Der Software-Endschalter ist momentan aktiv.

Bit 2: Zuständig für den negativen Software-Endschalter der Z-Achse
- = 0 Der Software-Endschalter ist momentan nicht aktiv.
- = 1 Der Software-Endschalter ist momentan aktiv.

Bit 3: Zuständig für den negativen Software-Endschalter der A-Achse
- = 0 Der Software-Endschalter ist momentan nicht aktiv.
- = 1 Der Software-Endschalter ist momentan aktiv.

Bit 4 ... 7: Reservierte Bits
- = 0 Diese Bits sind momentan noch nicht benutzt und immer gleich 0 gesetzt.

Register dh:

Bit 0: Zuständig für den positiven Software-Endschalter der X-Achse
- = 0 Der Software-Endschalter ist momentan nicht aktiv.
- = 1 Der Software-Endschalter ist momentan aktiv.

Bit 1: Zuständig für den positiven Software-Endschalter der Y-Achse
- = 0 Der Software-Endschalter ist momentan nicht aktiv.
- = 1 Der Software-Endschalter ist momentan aktiv.

Bit 2: Zuständig für den positiven Software-Endschalter der Z-Achse
- = 0 Der Software-Endschalter ist momentan nicht aktiv.
- = 1 Der Software-Endschalter ist momentan aktiv.

Bit 3: Zuständig für den positiven Software-Endschalter der A-Achse
= 0 Der Software-Endschalter ist momentan nicht aktiv.
= 1 Der Software-Endschalter ist momentan aktiv.

Bit 4 ... 7: Reservierte Bits
= 0 Diese Bits sind momentan noch nicht benutzt und immer gleich 0 gesetzt.

Ausführbarkeit während einer Bewegung: Ja
Mögliche Fehlercodes: 0; 2
Siehe auch: Kapitel 3.2.15; 3.2.17

3.3.6 Funktion 6: Referenzfahrt

Zweck	Ausführen die Referenzfahrt	
Aufrufparameter	ax = 6	Funktionsnummer
	bx	Achscode
	cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Mit dieser Funktion ermöglicht der Treiber Ihnen, die Referenzfahrt mühelos auszuführen. Alle notwendigen Parameter für die Referenzfahrt können Sie mit dem Konfigurationsprogramm PARKON.EXE einstellen (siehe Handbuch PARKON). Jede Achse Ihrer Anlage hat ein korrespondierendes Bit im Register bl. Anhand der Werte (0 oder 1) dieses Bits entscheidet der Treiber, ob die Referenzfahrt durchgeführt wird oder nicht. Das Register bh ist momentan noch unbenutzt und sollte beim Aufruf dieser Funktion immer gleich 0 gesetzt sein.

Im Folgenden können Sie die Zuordnung zwischen den Bits von bl und den einzelnen Achsen sehen (siehe Bild 3.19 für die Nummerierung der einzelnen Bits in einem Byte).

Register bl:
Bit 0: Für X-Achse
= 0 Keine Referenzfahrt
= 1 Referenzfahrt

Bit 1: Für Y-Achse
= 0 Keine Referenzfahrt
= 1 Referenzfahrt

Bit 2: Für Z-Achse
= 0 Keine Referenzfahrt
= 1 Referenzfahrt

Bit 3: Für A-Achse
= 0 Keine Referenzfahrt
= 1 Referenzfahrt

Bit 4 ... 7: Reservierte Bits
Diese Bits sollen beim Aufruf der Funktion immer gleich 0 gesetzt werden.

Mit dem Setzen bzw. Rücksetzen der einzelnen Bits sind Sie in der Lage, die Referenzfahrt sowohl für eine Achse als auch für mehrere Achsen durch einen einzigen Funktionsaufruf zu realisieren. Falls Sie die Referenzfahrt für mehr als eine Achse durchführen wollen, müssen Sie beachten, dass der Treiber die Referenzfahrt in der Reihenfolge Z, Y, X, A ausführt.
Wenn Sie z. B. die Referenzfahrt für die X-; Z- und A-Achse durchführen wollen, müssen Sie die Bits 0, 2, 3 gleich 1 und das Bit 1 gleich 0 setzen. Der Treiber führt die Referenzfahrt zuerst in der Z-Achse, dann in der X-Achse und anschließend in der A-Achse aus. Falls Sie die Hardware-Endschalter als Referenzschalter benutzen und falls es bei Ihrem Servocontroller notwendig ist, müssen Sie direkt vor der Referenzfahrt noch die Funktion 48 aufrufen, um den Sicherheitskreis Ihres Servocontrollers zu überbrücken (siehe Kapitel 2.5 und 3.2.6.2). Nach der Referenzfahrt ist es empfehlenswert, den Sicherheitskreis sofort wieder zu aktivieren.

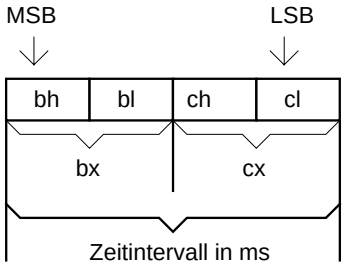
Viele Treiberfunktionen bleiben so lange gesperrt, bis Sie die Referenzfahrt für alle Achsen Ihrer Anlage ausgeführt haben. Nach jeder Referenzfahrt definiert der Treiber den Punkt als den Referenzpunkt Ihrer Anlage, in dem sich das Werkzeug direkt nach der Referenzfahrt befindet. Dieser Punkt ist gleichzeitig als der momentane Maschinen-Nullpunkt und der momentane Werkstück-Nullpunkt zu verstehen.
Daraus ergibt sich die Konsequenz, dass Sie die Referenzfahrt für alle Achsen Ihrer Anlage direkt nacheinander durchführen müssen. Bei den Achsen, bei denen keine Referenzfahrt durchgeführt wurde, kann es sonst passieren, dass die Positionen des Referenzpunktes nicht identisch mit den Positionen der Referenzschalter sind.

Ausführbarkeit während der Bewegung: Nein
Mögliche Fehlercodes: 0; 2; 3; 4; 10; 11; 15; 16; 19; 22
Siehe auch: Kapitel 2.5; 3.2.6.2; 3.2.15; 3.2.17

3.3.7 Funktion 7: Definition einer Zeitverzögerung

Zweck	Definition einer Zeitverzögerung	
Aufrufparameter	ax = 7	Funktionsnummer
	bx:cx	Zeitintervall (1 ms ... 60 000 ms)
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Diese Funktion realisiert für Sie eine Zeitverzögerung zwischen 1 ms ... 60 000 ms. Beim Aufruf dieser Funktion legen Sie das gewünschte Zeitintervall (in Millisekunde) in bx:cx fest (siehe Bild 3.20).



LSB = Last Significant Byte (Niederwertigste Byte)
MSB= Most Significant Byte (Höchstwertigste Byte)

Bild 3.20: Das Zeitintervall in bx:cx (in Millisekunden)

Der Treiber interpretiert den Wert in bx:cx als eine Zweier-Komplementär-Zahl. Wenn die Zahl in bx:cx größer als 60 000 ist, begrenzt der Treiber sie auf 60 000. Wenn sie kleiner als 1 ist, wird sie gleich 1 gesetzt, d. h. die Zeitverzögerung liegt immer im Bereich zwischen 1 ... 60 000 ms. Bei dieser Funktion müssen Sie beachten, dass der Treiber erst zu Ihnen zurückkehrt, wenn die gewünschte Verzögerungszeit abgelaufen ist. Während dieser Verzögerungszeit hat Ihr Programm keine Kontrolle über den PC.

Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 2
Siehe auch: Kapitel 3.2.17

3.3.8 Funktion 8: Ein- oder Ausschalten des Hand-Modus

Zweck	Die Anlage mit der Hand zu bewegen	
Aufrufparameter	ax = 8	Funktionsnummer
	bx	Unterfunktionen
	cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
normalerweise können Sie die einzelnen Achsen Ihrer Anlage nicht mit der Hand hin und her bewegen, weil jede Achse einen geschlossenen Regelkreis hat. Dieser Regelkreis wirkt gegen jede Bewegung, die die Achse aus ihrer Soll-Position herausführen soll. Diese Funktion ermöglicht es Ihnen, mit dem entsprechenden Wert im Register bx, den sogenannten Hand-Modus ein- bzw. auszuschalten. In diesem Hand-Modus schaltet der Treiber den Regelkreis der einzelnen Achse aus. Damit können Sie die Achsen mit der Hand bewegen.
Zuordnung zwischen dem Wert im bx und den einzelnen Unterfunktionen:

bx	Unterfunktionen
0	Ausschalten den Hand-Modus
1	Einschalten den Hand-Modus
sonst	Einschalten den Hand-Modus

Im Hand-Modus werden die Achspositionen vom Treiber weiterhin überwacht, d. h. wenn Sie die Achspositionen abfragen wollen, können Sie die Funktion 20 ohne weiteres benutzen. Die Werkzeug-Ist-Geschwindigkeit wird aber nicht mehr kontrolliert.
Wenn Sie den Hand-Modus ausschalten, schaltet der Treiber die Regelkreise wieder ein, nachdem er die aktuellen Achspositionen als die neuen Soll-Positionen der einzelnen Achsen definiert hat. Dadurch bleibt die Anlage in dem Punkt stehen, in dem sie direkt vor dem Ausschalten des Hand-Modus gestanden hat. Der Hand-Modus verursacht keinen Positionsverlust. Falls Ihre Anlage keine unterlagerten Geschwindigkeitsregelkreise hat, funktioniert der Hand-Modus immer. Falls die unterlagerten Geschwindigkeitsregelkreise vorhanden sind, funktioniert der Hand-Modus nur, wenn Sie das Datenbit 0 vom Ausgabeport 1 für das Enable/Disable der Leistungsendstufen benutzen (siehe Kapitel 1.1 und 2.5 bis 2.8).

Ausführbarkeit während der Bewegung: Nein
Mögliche Fehlercodes: 0, 2; 4; 9; 12
Siehe auch: Kapitel 3.2.17

3.3.9 Funktion 9: Ein- oder Ausschalten des Teach-In-Modus

Zweck	Realisieren der Teach-In-Bewegung	
Aufrufparameter	ax = 9	Funktionsnummer
	bx	Unterfunktionen
	cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Diese Funktion ermöglicht es Ihnen, die Teach-In-Bewegungen mühelos zu realisieren.

Die Zuordnung zwischen dem Wert im Register bx und den einzelnen Unterfunktionen ist folgende:

bx	Unterfunktionen
0	Ausschalten des Teach-In-Modus
1	Einschalten des Teach-In-Modus
sonst	Einschalten des Teach-In-Modus

Sie können den Teach-In-Modus nur einschalten, wenn sich Ihre Anlage nicht bewegt, d. h. der Treiber darf keine Bewegung im Hintergrund durchführen. Wenn eine Bewegung im Hintergrund aktiv ist, muss zuerst die Stop-Funktion aufgerufen werden, um die Achsen zum Stillstand zu bringen (siehe Kapitel 3.3.10). Falls der Treiber noch beim Abfahren eines Bewegungssegments ist, muss der Stop-Modus eingeschaltet werden. Sonst können Sie den Teach-In-Modus nicht einschalten. Die Kontrolle, ob sich die Anlage momentan bewegt oder nicht, können Sie durch einen Aufruf der Funktion 5 und das abschließende Bewerten des Bits 6 vom Register ah (siehe Kapitel 3.3.5) realisieren.
Im Teach-In-Modus ergeben sich folgende Besonderheiten, die Sie besonders beachten müssen:

- Bezüglich der Bewegungsfunktionen können Sie im Teach-In-Modus nur noch die beiden Funktionen 24 und 25 aufrufen (siehe Kapitel 3.3.24 und 3.3.25). Wenn Sie eine dieser Funktionen aufrufen, schaltet der Treiber seinen Arbeitsspeicher um, d. h. er arbeitet mit einem anderen Speicherbereich als im Normalfall. Dadurch bleiben die Daten eines Bewegungssegments erhalten, falls dieses Segment vor dem Einschalten in den Teach-In-Modus noch nicht fertig abgearbeitet wurde.
- Im Teach-In-Modus interpoliert der Treiber das Bewegungssegment mit der Teach-In-Geschwindigkeit, deren Standardwert Sie mit dem Konfigurationsprogramm einstellen können. Diese Geschwindigkeit können Sie aber auch jederzeit durch einen Aufruf der Funktion 22 ändern (siehe Kapitel 3.3.22).
- Eine Geschwindigkeitsänderung mit der Funktion 21 ist im Teach-In-Modus nicht möglich (siehe Kapitel 3.2.10 und 3.3.21).
- Wenn Sie die Stop-Funktion während der Abarbeitung eines Segments aufrufen, wird die Bewegung sofort unterbrochen (siehe Kapitel 3.3.10). Der Rest des Segments wird "vergessen", d. h. für den Treiber und für Sie ist das Segment schon fertig abgearbeitet. Das ist der entscheidende Unterschied zum Normalfall.
  Im Normalfall geht der Rest des Segments nach einer Stop-Funktion nicht verloren. Dieser Segmentrest wird nach einer Start-Funktion abgearbeitet (siehe Kapitel 3.3.10).
- Die Start-Funktion und die Stop-Funktion haben nur lokale Wirkungen innerhalb des Teach-In-Modus, wenn sie im Teach-In-Modus aufgerufen werden, d. h. diese beiden Funktionen haben keine Nachwirkungen auf den Treiber, nach dem der Teach-In-Modus ausgeschaltet ist.

- Die Break-Funktion können Sie im Teach-In-Modus nicht aufrufen (siehe Kapitel 3.3.10).
- Die Behandlung der Hard- und Software-Endschalterfehler ist anders als im Normalfall (siehe Kapitel 3.2.3 und 3.3.5).

Ähnlich wie beim Einschalten können Sie den Teach-In-Modus nur ausschalten, wenn sich Ihre Anlage nicht mehr bewegt. Nach dem Ausschalten des Teach-In-Modus benutzt der Treiber wieder seinen alten Speicherbereich, den er schon vor dem Einschalten in den Teach-In-Modus gehabt hatte. Einen evtl. vorhandenen Segmentrest können Sie normal mit der Start-Funktion abarbeiten (siehe 3.3.10).

Das Werkzeug bewegt sich von dem Punkt weiter, in dem er sich direkt nach dem Ausschalten des Teach-In-Modus befand. Er kehrt nicht zu dem Punkt zurück, in dem er direkt vor dem Einschalten des Teach-In-Modus stand. Dadurch ergibt sich eine Verschiebung des Segmentrestes (siehe Bild 3.21).

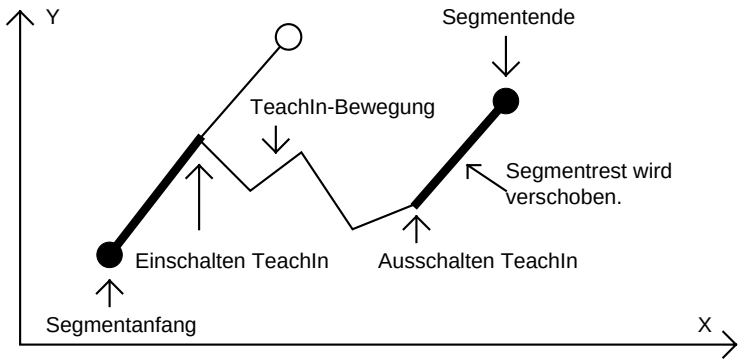


Bild 3.21: Verschiebung des Bewegungssegments nach einem Teach-In am Beispiel eines Linearsegmentes

Auszuführen während der Bewegung: nur nach einem Aufruf der Stop-Funktion
Mögliche Fehlercodes: 0; 2; 4; 9; 15
Siehe auch: Kapitel 3.2.17

3.3.10 Funktion 10: Start-Stop-Break-Abort

Zweck	Damit können Sie Ihre Anlage stoppen (Stop) oder weiterfahren (Starten) nach einem Stop oder den Bearbeitungsvorgang unterbrechen (Break und Abort).	
Aufrufparameter	ax = 10	Funktionsnummer
	bx	Unterfunktionen
	cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Durch das Setzen von entsprechenden Werten in bx sind Sie in der Lage, Ihre Maschine während des Fahrens eines Segments anzuhalten und danach weiterzufahren oder den ganzen Verarbeitungsvorgang zu unterbrechen.
Zuordnung zwischen dem Wert in bx und den einzelnen Unterfunktionen:

bx	Unterfunktion
0	Starten (Weiterfahren nach einem Stop)
1	Stop
2	Break (Not-Aus-Unterbrechen)
3	Abort (Unterbrechen)
sonst	Abort (Unterbrechen).

Während des Fahrens eines Segments können Sie diese Funktion mit bx = 1 aufrufen, um die Anlage anzuhalten. Damit schalten Sie den sogenannten Stop-Modus des Treibers ein. Ein Aufruf dieser Funktion mit bx = 0 lässt die Anlage danach weiterfahren, d. h. Sie schalten den Stop-Modus aus. Durch einen Aufruf der Funktion 5 können Sie sich jederzeit vergewissern, ob sich der Treiber momentan im Stop-Modus befindet oder nicht (siehe Kapitel 3.3.5).
Falls Sie die Start-Unterfunktion aufrufen, ohne die Stop-Unterfunktion vorher aufgerufen zu haben, hat dieser Aufruf keine Wirkung. Falls die Stop-Unterfunktion aufgerufen wird, während Ihre Anlage nicht beim Fahren eines Segments ist, bleibt die Wirkung dieses Aufrufes erhalten, bis Sie eine Start-Unterfunktion aufrufen. Bevor Sie nicht die Start-Funktion aufgerufen haben, bewegt sich die Anlage nicht, falls Sie eine der Bewegungsfunktionen nutzen (siehe Kapitel 3.3.24 bis 3.3.29).

Der größte Nutzen dieser beiden Unterfunktionen besteht darin, dass Sie während des Fahrens eines Bewegungssegmentes die Teach-In-Bewegungen einfach realisieren können (siehe Kapitel 3.3.9).
Die Break-Unterfunktion mit bx = 2 können Sie jederzeit aufrufen. Beim Aufruf dieser Funktion bleibt Ihre Anlage sofort stehen. Es werden dabei keine Bremsrampen wie bei der Stop-Unterfunktion generiert. Intern wird ein sogenanntes Break-Flag gleich 1 gesetzt.
Den Zustand dieses Break-Flags können Sie jederzeit durch einen Aufruf der Funktion 5 ermitteln (siehe Kapitel 3.3.5). Diese Unterfunktion können Sie als die Not-Aus-Funktion verstehen. Solange das Break-Flag gesetzt ist, stehen die meisten Treiberfunktionen, darunter auch alle Bewegungsfunktionen, nicht zur Verfügung. Eine Ausnahme bildet der Teach-In-Modus. Wenn Sie den Teach-In-Modus einschalten, können Sie weiterhin die beiden Bewegungsfunktionen 24 und 25 aufrufen (siehe Kapitel 3.3.24 und 3.3.25).
Somit ist die Teach-In-Bewegung selbst bei einem Break-Fehler weiterhin möglich. Das Aufrufen der Reset-Funktion ist die einzige Möglichkeit, dieses Break-Flag zurückzusetzen.
Die Abort-Unterfunktion mit bx = 3 können Sie jederzeit aufrufen. Der Aufruf hat aber keine Wirkung, falls es momentan keine Bearbeitung im Hintergrund gibt.

Während einer Segmentbearbeitung führt ein Aufruf dieser Unterfunktion, ähnlich wie bei der Stop-Unterfunktion, zu einem rampenmäßigen Anhalten der Anlage. Aber im Unterschied zur Stop-Unterfunktion geht ein evtl. vorhandener Segmentrest hier unwiderruflich verloren. Dies ist unabhängig davon, ob der Teach-In-Modus aktiv ist oder nicht. Dabei wird kein Fehlerflag gesetzt. Falls der Teach-In-Modus aktiv ist, hat diese Unterfunktion die gleiche Wirkung wie die Stop-Unterfunktion.
Der größte Nutzen dieser Unterfunktion liegt im Normal-Modus, falls Sie eine Bewegung unterbrechen wollen, ohne die Break-Unterfunktion zu benutzen. Dabei wird kein Fehlerflag gesetzt.

Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 2; 3; 7; 8; 9; 12; 15; 19; 22; 27
Siehe auch: Kapitel 3.2.17

3.3.11 Funktion 11: Setzen des aktuellen Punktes als Werkstück-Nullpunkt

Zweck	Setzen des momentanen Punktes des Werkzeugs als neuen Werkstück-Nullpunkt	
Aufrufparameter	ax = 11	Funktionsnummer
	bx, cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Der Punkt, in dem das Werkzeug momentan steht, wird durch einen Aufruf dieser Funktion als neuer Werkstück-Nullpunkt definiert. Der alte Werkstück-Nullpunkt wird gelöscht. Dieser neue Werkstück-Nullpunkt ist solange der Bezugspunkt für alle nachfolgenden Absolut-Koordinatenangaben, bis wieder ein neuer Werkstück-Nullpunkt definiert wird.

Ausführbarkeit während der Bewegung: Nein
Mögliche Fehlercodes: 0; 2; 3; 4; 7; 8; 9; 15; 19; 22; 27
Siehe auch: Kapitel 3.2.1; 3.2.17; 3.3.12

3.3.12 Funktion 12: Setzen Werkstück-Nullpunkt

Zweck	Setzen des neuen Werkstück-Nullpunktes	
Aufrufparameter	ax = 12	Funktionsnummer
	bx	Segmentadresse des neuen Werkstück-Nullpunktes
	cx	Offsetadresse des neuen Werkstück-Nullpunktes
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Ähnlich wie bei Funktion 11 können Sie mit einem Aufruf dieser Funktion einen neuen Werkstück-Nullpunkt definieren. In bx:cx übergeben Sie dem Treiber die Anfangsadresse eines Speicherbereichs, in dem die Absolutkoordinaten des neuen Werkstück-Nullpunktes stehen. Der Bezugspunkt für diese Koordinaten ist der zuletzt definierte Werkstück-Nullpunkt. Falls Sie noch keinen Werkstück-Nullpunkt definiert haben, ist der Referenzpunkt der Bezugspunkt. Jede Achskoordinate belegt 4 Byte im Speicher und wird vom Treiber als Zweier-Komplementär-Zahl interpretiert. In Abhängigkeit von der Anlagenstruktur ist die Einheit der Koordinaten entweder Mikrometer [μm] oder Bogensekunde ["]. Im Bild 3.22 können Sie die Speicherzuordnung für die einzelnen Absolutkoodinaten des neuen Werkstück-Nullpunktes sehen. Sie sollten beachten, dass es zwar für Ihre zukünftigen Anwendungen sinnvoll ist, beim Aufruf dieser Funktion 16 Byte Speicher für den Datenaustausch zu reservieren. Dies muss aber nicht sein; wenn Ihre Anlage z. B. nur 3 Achsen hat, brauchen Sie nur 12 Byte im Speicher für die 3 Achsen X, Y und Z zu reservieren.

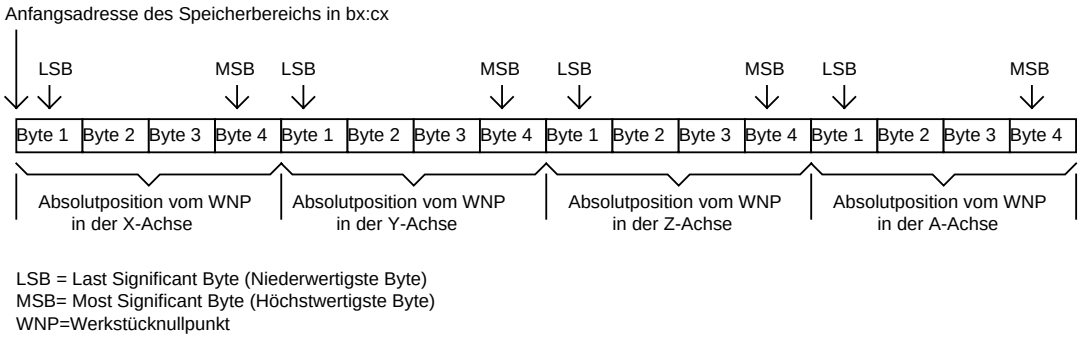


Bild 3.22: Belegung des Speichers für die Absolutkoordinaten des neuen Werkstück-Nullpunktes

Dieser neue Werkstück-Nullpunkt ist dann der Bezugspunkt für alle nachfolgenden Absolutkoordinatenangaben, bis ein neuer Werkstück-Nullpunkt definiert wird. Der alte Werkstück-Nullpunkt wird beim Aufruf dieser Funktion gelöscht.

Ausführbarkeit während der Bewegung: Nein
Mögliche Fehlercodes: 0; 2; 3; 4; 7; 8; 9; 15; 19; 22; 27
Siehe auch: Kapitel 3.2.1; 3.2.9; 3.2.13; 3.2.17; 3.3.11

3.3.13 Funktion 13: Löschen Werkstück-Nullpunkt

Zweck	Löschen des Werkstück-Nullpunktes	
Aufrufparameter	ax = 13	Funktionsnummer
	bx, cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Mit dieser Funktion können Sie den Werkstück-Nullpunkt löschen. Nach einem Aufruf dieser Funktion ist der Referenzpunkt der neue Werkstück-Nullpunkt, d. h. der Referenzpunkt dient als Bezugspunkt für alle nachfolgenden Angaben von Absolutkoordinaten.

Ausführbarkeit während der Bewegung: Nein
Mögliche Fehlercodes: 0; 2; 3; 4; 7; 8; 9; 15; 19; 22
Siehe auch: Kapitel 3.2.1; 3.2.17; 3.3.11; 3.3.12

3.3.14 Funktion 14: Setzen die Software-Endschalter

Zweck	Setzen der Software-Endschalter	
Aufrufparameter	ax = 14	Funktionsnummer
	bx	Segmentadresse des Software-Endschalters
	cx	Offsetadresse des Software-Endschalters
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Mit dieser Funktion definieren Sie sowohl die negativen als auch die positiven Software-Endschalter aller Achsen Ihrer Anlage neu.
Im Speicherbereich, dessen Anfangsadresse Sie dem Treiber über bx:cx übergeben, stehen die neuen Positionen der Software-Endschalter der einzelnen Achsen (siehe Bild 3.23). Diese neuen Positionen der Software-Endschalter sind Absolutkoordinaten und haben den Werkstück-Nullpunkt als den Bezugspunkt.

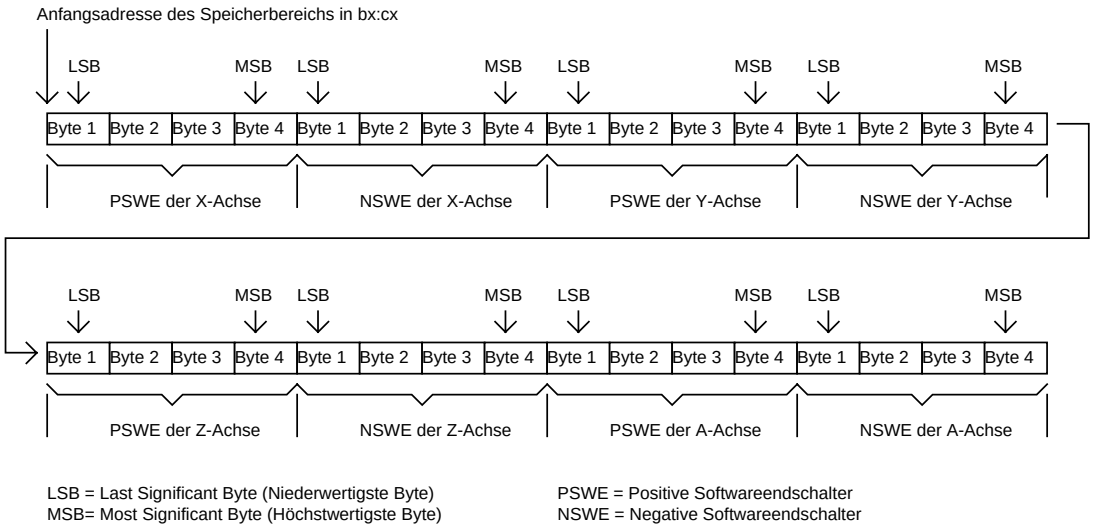


Bild 3.23: Belegung des Speichers für die Software-Endschalter

Jeder Wert der Software-Endschalter ist eine ganze Zweier-Komplementär-Zahl und belegt 4 Byte im Speicherbereich. In Abhängigkeit von der Anlagenstruktur ist die Einheit entweder Mikrometer [μm] oder Bogensekunde ["].

Die Standardwerte der Software-Endschalter können Sie über das Konfigurationsprogramm PARKON.EXE einstellen (siehe Handbuch PARKON). Beachten Sie, dass es zwar für Ihre zukünftigen Anwendungen sinnvoll ist, beim Aufruf dieser Funktion 32 Byte Speicher für den Datenaustausch zu reservieren, dies muss aber nicht sein. Wenn Ihre Anlage z. B. nur 3 Achsen hat, brauchen Sie für die 3 Achsen X, Y und Z nur 24 Byte im Speicher zu reservieren.

Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 2; 3; 5; 6; 7; 8; 9; 15; 19; 22
Siehe auch: Kapitel 3.2.1; 3.2.9; 3.2.13; 3.2.17; 3.3.15

3.3.15 Funktion 15: Sperren/Freilassen von Software-Endschaltern

Zweck	Sperren oder Freilassen der Software-Endschalter	
Aufrufparameter	ax = 15	Funktionsnummer
	bx	Unterfunktionen
	cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Mit dieser Funktion haben Sie die Möglichkeit, die Software-Endschalter zu aktivieren oder zu deaktivieren.
Nach dem Start des Treibers oder nach dem Reset sind die Software-Endschalter immer gesperrt, d. h. sie sind nicht aktiv. Nur durch diese Funktion haben Sie die Möglichkeit, die Software-Endschalter freizugeben oder zu sperren. Die Funktion 14 kann zwar neue Software-Endschalter definieren, sie kann deren Benutzung aber nicht aktivieren oder deaktivieren.
Außerdem müssen Sie noch beachten, dass Sie durch das Konfigurationsprogramm selbst festlegen können, ob jede Achse Ihrer Anlage keine oder nur eine oder zwei Hardware-Endschalter hat (siehe Handbuch für das Konfigurationsprogramm PARKON.EXE). Bei den Software-Endschaltern ist das nicht mehr so. Wenn Sie die Benutzung der Software-Endschalter offen lassen, hat jede Achse Ihrer Anlage automatisch den positiven und den negativen Software-Endschalter. Der Wert in bx und dessen Bedeutung ist:

bx	Bedeutung
0	Deaktivieren der Software-Endschalter
1	Aktivieren der Software-Endschalter
sonst	Aktivieren der Software-Endschalter

Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 2; 3; 5; 7; 8; 9; 15; 19; 22
Siehe auch: Kapitel 3.2.6; 3.2.15; 3.3.14

3.3.16 Funktion 16: Lesen eines Bytes von einem Eingabeport

Zweck	Lesen von einem Eingabeport	
Aufrufparameter	ax = 16	Funktionsnummer
	bx	Portadresse
	cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx	Inhalt vom Eingabeport
	cx, dx	undefiniert

Kommentar:
Diese Funktion ermöglicht es Ihnen, den Inhalt eines Eingabeports zu lesen. In bx müssen Sie die Portadresse ablegen. Bei der Rückkehr vom Treiber bekommen Sie den Portinhalt in bl.
Ab der Version 3.10 steht diese Funktion nicht mehr zur Verfügung (siehe Kapitel 3.2.15). Anstelle dieser Funktion sollen Sie die Funktionen 36 und 37 benutzen. Beim Aufruf dieser Funktion bekommen Sie den Fehlercode 1 „Falsche Funktionsnummer“ zurück.

Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 1; 2
Siehe auch: Kapitel 3.2.15, 3.2.17; 3.3.36; 3.3.37

3.3.17 Funktion 17: Ausgeben eines Bytes an einen Ausgabeport

Zweck	Ausgeben von einem Byte an einen Ausgabeport	
Aufrufparameter	ax = 17	Funktionsnummer
	bx	Portadresse
	cx	Ausgabewert
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Mit dieser Funktion sind Sie in der Lage, ein Byte an einem Ausgabeport auszugeben. Sie müssen bei einem Aufruf dieser Funktion die Portadresse in bx und das Ausgabebyte in cx ablegen.
Ab der Version 3.10 steht diese Funktion nicht mehr zur Verfügung (siehe Kapitel 3.2.15). Anstelle dieser Funktion sollen Sie die Funktionen 38 bis 42 benutzen. Beim Aufruf dieser Funktion bekommen Sie den Fehlercode 1 „Falsche Funktionsnummer“ zurück.

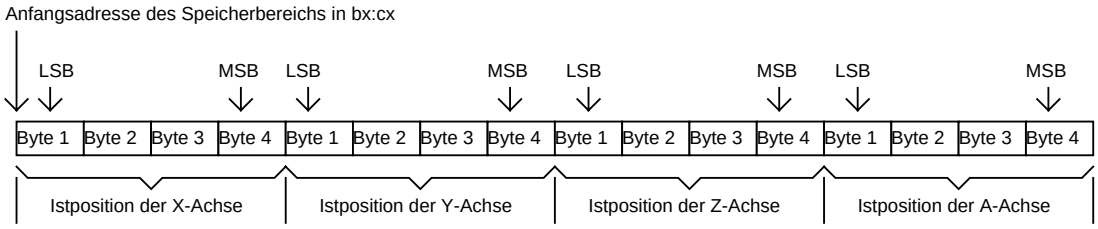
Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 1; 2
Siehe auch: Kapitel 3.2.15; 3.2.17; 3.3.38; 3.3.39; 3.3.42

3.3.18 Funktion 18: Abfragen der Ist-Positionen der Achsen

Zweck	Abfragen der aktuellen Ist-Positionen der Achsen	
Aufrufparameter	ax = 18	Funktionsnummer
	bx	Segmentadresse der Ist-Positionen
	cx	Offsetadresse der Ist-Positionen
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Diese Funktion ermöglicht es Ihnen, die aktuellen Ist-Positionen der Achsen abzufragen. Um diese Funktion nutzen zu können, müssen Sie zuerst einen Speicherbereich in Ihrem Anwenderprogramm reservieren. Die Anfangsadresse dieses Speicherbereichs legen Sie beim Aufruf dieser Funktion in bx:cx fest. Mit Hilfe dieser Anfangsadresse schreibt der Treiber die Ist-Positionen der Achsen in den von Ihnen reservierten Speicherbereich (siehe Bild 3.24).

Für jede Achse braucht der Treiber 4 Byte Speicher. Den Wert in diesen 4 Bytes müssen Sie als eine ganze Zweier-Komplementär-Zahl interpretieren. Beachten Sie, dass es zwar für Ihre zukünftigen Anwendungen sinnvoll ist, beim Aufruf dieser Funktion 16 Byte Speicher für den Datenaustausch zu reservieren, dies muss aber nicht sein. Wenn Ihre Anlage z. B. nur 3 Achsen hat, brauchen Sie für die 3 Achsen X, Y und Z nur 12 Byte im Speicher zu reservieren.



LSB = Last Significant Byte (Niederwertigste Byte)
MSB= Most Significant Byte (Höchstwertigste Byte)

Bild 3.24: Belegung des Speichers mit den aktuellen Ist-Positionen der Achsen

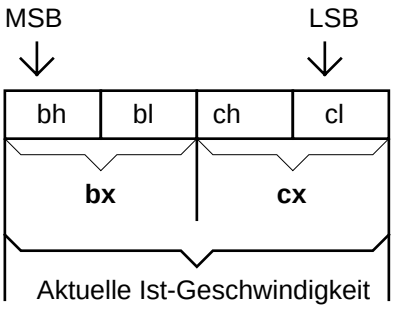
Die Ist-Positionen, die Sie zurückbekommen, sind Absolutpositionen, d. h. die Ist-Positionen haben den Werkstück-Nullpunkt als Bezugspunkt.
Beachten Sie, dass der Werkstück-Nullpunkt nach dem Laden oder nach Reset des Treibers identisch mit dem Referenzpunkt ist. In Abhängigkeit von der Anlagenstruktur ist die Einheit der Ist-Positionen entweder Mikrometer [μm] oder Bogensekunde ["].

Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 2; 9
Siehe auch: Kapitel 3.2.9; 3.2.13; 3.2.17; 3.3.19

3.3.19 Funktion 19: Abfragen der Werkzeug-Ist-Geschwindigkeit

Zweck	Abfragen der aktuellen Werkzeug-Ist-Geschwindigkeit	
Aufrufparameter	ax = 19	Funktionsnummer
	bx, cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx:cx	Aktuelle Ist-Geschwindigkeit
	dx	undefiniert

Kommentar:
Mit dieser Funktion sind Sie in der Lage, die aktuelle Ist-Geschwindigkeit des Werkzeugs jederzeit abzufragen. Der Treiber legt die Ist-Geschwindigkeit in bx:cx ab (siehe Bild 3.25).



LSB = Last Significant Byte (Niederwertigste Byte)
MSB= Most Significant Byte (Höchstwertigste Byte)

Bild 3.25: Die aktuelle Ist-Geschwindigkeit in bx:cx

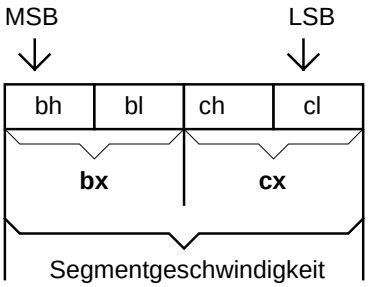
Sie müssen den Wert in bx:cx als eine ganze Zweier-Komplementär-Zahl interpretieren. Es hängt von der Anlagenstruktur ab, ob die Einheit der Werkzeuggeschwindigkeit Mikrometer/Sekunde [μm/s] oder Bogensekunde/Sekunde ["/s] ist.

Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 2; 9; 10
Siehe auch: Kapitel 3.2.9; 3.2.10; 3.2.13; 3.2.17; 3.3.18

3.3.20 Funktion 20: Setzen der Segment-Geschwindigkeit

Zweck	Setzen einer neuen Segment-Geschwindigkeit	
Aufrufparameter	ax = 20	Funktionsnummer
	bx:cx	Bahngeschwindigkeit
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Mit dieser Funktion können Sie eine neue Segment-Geschwindigkeit des Werkzeugs definieren. Solange Sie diese Funktion nach dem Laden oder nach einem Reset des Treibers noch nicht aufgerufen haben, nimmt die Werkzeuggeschwindigkeit ihren Standardwert an, den Sie über das Konfigurationsprogramm PARKON.EXE eingestellt haben (siehe Handbuch PARKON). Die neue Segment-Geschwindigkeit, die Sie dem Treiber übergeben wollen, steht in bx:cx (siehe Bild 3.26).



LSB = Last Significant Byte (Niederwertigste Byte)
MSB= Most Significant Byte (Höchstwertigste Byte)

Bild 3.26: Die neue Segment-Geschwindigkeit in bx:cx

Diese Funktion können Sie während des Fahrens eines Bewegungssegments aufrufen. Der neue Wert der Geschwindigkeit gilt aber erst beim Fahren des nächsten Bewegungssegments. Der Treiber interpretiert die Zahl in bx:cx als eine ganze Zweier-Komplementär-Zahl. In Abhängigkeit von der Anlagenstruktur ist die Einheit der Bahngeschwindigkeit entweder Mikrometer/Sekunde [μm/s] oder Bogensekunde/Sekunde ["/s].

Ausführbarkeit während der Bewegung: Ja; der neue Wert wird erst beim Fahren des nächsten Bewegungssegments angenommen.
Mögliche Fehlercodes: 0; 2; 3; 7; 8; 9; 15; 17; 19; 22
Siehe auch: Kapitel 3.2.9; 3.2.10; 3.2.13; 3.2.17; 3.3.19; 3.3.32

3.3.21 Funktion 21: Setzen des Änderungsfaktors der Werkzeuggeschwindigkeit

Zweck	Änderung der momentanen Werkzeuggeschwindigkeit	
Aufrufparameter	ax = 21	Funktionsnummer
	bx	Änderungsfaktor der Bahngeschwindigkeit
	cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Normalerweise benutzt der Interpolationsalgorithmus während der Bearbeitung bzw. während eines Eilgangs direkt die Bearbeitungs- bzw. die Eilgeschwindigkeit für seine internen Berechnungen bei der Koordinierung der Achsbewegung (die Bearbeitungsgeschwindigkeit ist entweder die Segment- oder die Bahngeschwindigkeit). Damit wäre die Werkzeuggeschwindigkeit gleich der Bearbeitungs- bzw. der Eilgeschwindigkeit.
Um aber die Werkzeuggeschwindigkeit ändern zu können, haben wir einen sogenannten Änderungsfaktor Kv eingeführt. Bei seinen Berechnungen benutzt der Interpolationsalgorithmus nicht direkt die Bearbeitungs- bzw. die Eilgeschwindigkeit, sondern das Produkt aus Kv und der Bearbeitungs- bzw. der Eilgeschwindigkeit.

Durch den Aufruf dieser Funktion können Sie Kv und damit auch die Werkzeuggeschwindigkeit ändern. Eine Ausnahme bildet der Teach-In-Modus. In diesem Modus ist die Werkzeuggeschwindigkeit immer gleich der Teach-In-Geschwindigkeit, d. h. der Änderungsfaktor Kv hat keine Wirkung im Teach-In-Modus. In diesem Modus können Sie diese Funktion nicht aufrufen.
Den Wert von Kv, den Sie in bx abgelegt haben, interpretiert der Treiber als eine ganze Zweier-Komplementär-Zahl. Dieser Wert ist der Prozentsatz von Bearbeitungs- bzw. Eilgeschwindigkeit.
Um dies zu verdeutlichen, haben wir für Sie ein paar Beispiele berechnet:

Bearbeitungs- bzw. Eilgeschwindigkeit	Wert vom Kv in bx	Berechnung	Werkzeuggeschwindigkeit nach Funktionsaufruf
70 000 [μm/s]	**0**	70 000 * **0 %**	0 [μm/s]
20 000 [μm/s]	**50**	20 000 * **50 %**	10 000 [μm/s]
50 000 [μm/s]	**100**	50 000 * **100 %**	50 000 [μm/s]
30 000 [μm/s]	**140**	30 000 * **140 %**	42 000 [μm/s]

Beachten Sie, dass diese Funktion die Bearbeitungs- bzw. die Eilgeschwindigkeit nicht ändert. Wenn z. B. die Bearbeitungsgeschwindigkeit 10 000 μm/s beträgt, ist die Werkzeuggeschwindigkeit 5 000 μm/s, nachdem Sie diese Funktion mit bx = 50 aufgerufen haben. Beim nächsten Aufruf dieser Funktion mit z. B. bx = 20 bekommen Sie für die Werkzeuggeschwindigkeit den Wert vom 2 000 μm/s (= 10 000 μm/s * 20 %) nicht aber den Wert vom 1 000 μm/s (= 5 000 μm/s * 20 %).
Nach dem Laden oder nach einem Reset ist Kv standardmäßig gleich 100. Das bedeutet:

Werkzeuggeschwindigkeit	=	Bearbeitungsgeschwindigkeit * 100 %
	=	Bearbeitungsgeschwindigkeit bzw.
Werkzeuggeschwindigkeit	=	Eilgeschwindigkeit * 100 %
	=	Eilgeschwindigkeit

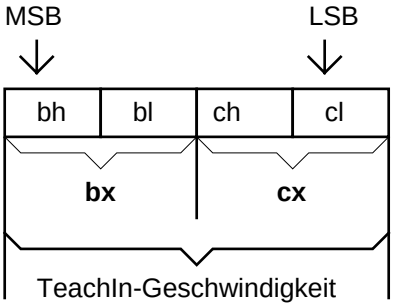
Der Wert vom Kv gilt nicht nur für das Fahren eines einzigen Segments sondern bleibt gültig bis zum erneuten Aufruf dieser Funktion oder bis zum Reset des Treibers. Kv kann einen Wert zwischen 0 bis 140 annehmen d. h. zwischen 0 % ... 140 % der Bearbeitungs- bzw. Eilgeschwindigkeit können Sie die Werkzeuggeschwindigkeit variieren.
Wenn der Wert in bx größer als 140 ist, wird Kv gleich 140 gesetzt. Falls dieser Wert kleiner als 0 ist, wird Kv = 0.

Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 2; 3; 7; 8; 9; 12; 15; 19; 22
Siehe auch: 3.2.10; 3.2.13; 3.2.17; 3.3.18; 3.3.20; 3.3.32

3.3.22 Funktion 22: Setzen der Teach-In-Geschwindigkeit

Zweck	Setzen einer neue Teach-In-Geschwindigkeit	
Aufrufparameter	ax = 22	Funktionsnummer
	bx:cx	Teach-In-Geschwindigkeit
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Im Teach-In-Modus benutzt der Interpolationsalgorithmus direkt die Teach-In-Geschwindigkeit für seine internen Berechnungen. Der Geschwindigkeitsänderungsfaktor Kv hat in diesem Modus keine Wirkung. Damit wollen wir garantieren, dass Sie in jedem Fall die Teach-In-Bewegungen ausführen können.
Mit dieser Funktion haben Sie die Möglichkeit, eine neue Teach-In-Geschwindigkeit zu definieren. Vor dem Aufruf dieser Funktion legen Sie den neuen Wert der Teach-In-Geschwindigkeit in bx:cx ab. Diesen Wert interpretiert der Treiber als eine ganze Zweier-Komplementär-Zahl (siehe Bild 3.27). Die Einheit der Geschwindigkeit ist in Abhängigkeit von der Anlagenstruktur entweder Mikrometer/Sekunde [μm/s] oder Bogensekunde/Sekunde [‘‘/s].



LSB = Last Significant Byte (Niederwertigste Byte)
MSB= Most Significant Byte (Höchstwertigste Byte)

Bild 3.27: Die neue Teach-In-Geschwindigkeit in bx:cx

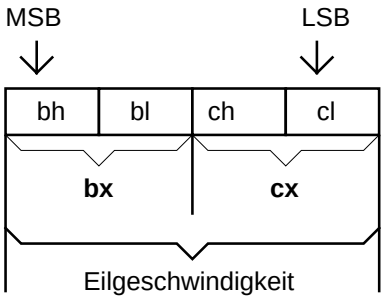
Diese Funktion können Sie zwar während der Bewegung aufrufen, der neue Wert wird vom Treiber aber erst beim Fahren des nächsten Bewegungssegments im Teach-In-Modus akzeptiert. Auf das momentane Bewegungssegment hat diese Funktion also keinen Einfluss.
Nach dem Laden oder nach einem Reset des Treibers hat die Teach-In-Geschwindigkeit den Standardwert, den Sie über das Konfigurationsprogramm PARKON.EXE eingestellt haben (siehe Handbuch PARKON).

Ausführbarkeit während der Bewegung: Ja, der neue Wert wird erst beim Fahren des nächsten Bewegungssegments angenommen.
Mögliche Fehlercodes: 0; 2; 3; 7; 8; 9; 15; 16; 19; 22
Siehe auch: Kapitel 3.2.9; 3.2.10; 3.2.13; 3.2.17; 3.3.9

3.3.23 Funktion 23: Setzen der Eilgeschwindigkeit

Zweck	Setzen einer neuen Eilgeschwindigkeit	
Aufrufparameter	ax = 23	Funktionsnummer
	bx:cx	Eilgeschwindigkeit
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Beim Eilgang (siehe Kapitel 3.3.26 und 3.3.27) benutzt der Interpolationsalgorithmus das Produkt aus dem Änderungsfaktor Kv und der Eilgeschwindigkeit für seine internen Berechnungen. Mit dieser Funktion haben Sie die Möglichkeit, eine neue Eilgeschwindigkeit zu definieren.
Vor einem Aufruf dieser Funktion legen Sie den neuen Wert in bx:cx ab.
Der Treiber interpretiert den Wert in bx:cx als eine ganze Zweier-Komplementär-Zahl (siehe Bild 3.28). Die Einheit der Geschwindigkeit ist in Abhängigkeit von der Anlagenstruktur entweder Mikrometer/Sekunde [μm/s] oder Bogensekunde/Sekunde [‘‘/s].



LSB = Last Significant Byte (Niederwertigste Byte)
MSB= Most Significant Byte (Höchstwertigste Byte)

Bild 3.28: Die neue Eilgeschwindigkeit in bx:cx

Der Änderungsfaktor Kv ist beim Eilgang weiterhin aktiv wie während einer normalen Bearbeitungsphase (siehe Kapitel 3.3.20), d. h. er hat hier weiter seine vergrößernde bzw. verkleinernde Wirkung auf die Werkzeug-geschwindigkeit.
Nach dem Laden oder nach einem Reset des Treibers hat die Eilgeschwindigkeit ihren Standardwert, der über das Konfigurationsprogramm PARKON.EXE eingestellt werden kann (siehe Handbuch PARKON).
Diese Funktion können Sie während der Bewegung aufrufen. Der neue Wert der Eilgeschwindigkeit wird aber erst beim Fahren des nächsten Bewegungs-segmetes im Eilgang akzeptiert.

Ausführbarkeit während der Bewegung: Ja, der neue Wert wird erst beim Fahren des nächsten Bewegungs-segmentes angenommen.
Mögliche Fehlercodes: 0; 2; 3; 7; 8; 9; 15; 17; 19; 22
Siehe auch: Kapitel 3.2.9; 3.2.10; 3.2.13; 3.2.17; 3.3.21; 3.3.32

3.3.24 Funktion 24: Relative Linearnormalbewegung

Zweck	Ausführen einer relativen Linearbewegung mit der Segment-Geschwindigkeit	
Aufrufparameter	ax = 24	Funktionsnummer
	bx	Segment- Adresse der Endposition
	cx	Offset-Adresse der Endposition
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Mit dieser Funktion sind Sie in der Lage, eine Linearinterpolation für bis zu 4 Achsen zu realisieren. Über die Register bx:cx müssen Sie dem Treiber eine FAR-Adresse übergeben. Diese Adresse ist die Anfangsadresse eines von Ihnen in Ihrem Anwenderprogramm reservierten Speicherbereichs, in den Sie vor dem Aufruf dieser Funktion die Endpositionen der 4 Achsen abgelegt haben. Die Einheit des Positionswertes ist entweder Mikrometer [μm] oder Bogensekunde ["]. Welche Einheit dabei benutzt wird, hängt von der Struktur Ihrer Anlage ab (siehe Kapitel 3.2.9).
Das Wort "Relativ" im Funktionsnamen hat mit dem Bezugspunkt der zu übergebenen Endpositionen zu tun (siehe Kapitel 3.2.1). Der Bezugspunkt für diese Endpositionen ist der Punkt, in dem das Werkzeug direkt vor dem Funktionsaufruf steht. Beim Fahren mehrerer Segmente hintereinander ist der Bezugspunkt der Endpunkt des letzten Segments und gleichzeitig der Anfangspunkt des zu fahrenden Segments (siehe Kapitel 3.2.1).
Das Wort "Normal" im Namen dieser Funktion hängt mit der Werkzeug-Soll-Geschwindigkeit zusammen, die das Produkt aus dem Änderungsfaktor Kv und der Bahngeschwindigkeit ist, solange der Teach-In-Modus nicht aktiv ist.

Mit Hilfe der Funktion 21 können Sie während der Bewegung den Faktor Kv und damit auch die Werkzeuggeschwindigkeit ändern.
Im Teach-In-Modus ist die Werkzeug-Soll-Geschwindigkeit immer gleich der Teach-In-Geschwindigkeit. Der Änderungsfaktor Kv hat also keine Wirkung in diesem Modus.
Nach dem Laden und nach einem Reset des Treibers haben die Segment-, die Bahn- bzw. die Teach-In-Geschwindigkeit die mit dem Konfigurations-programm eingestellten Standardwerte (siehe Handbuch für das Konfigurationsprogramm PARKON.EXE). Mit Funktion 20 bzw. 22 sind Sie in der Lage, diese Geschwindigkeiten zu ändern.
Im Folgenden wollen wir Ihnen erklären, wie die Endpositionen im Speicherbereich stehen müssen (siehe Bild 3.29).

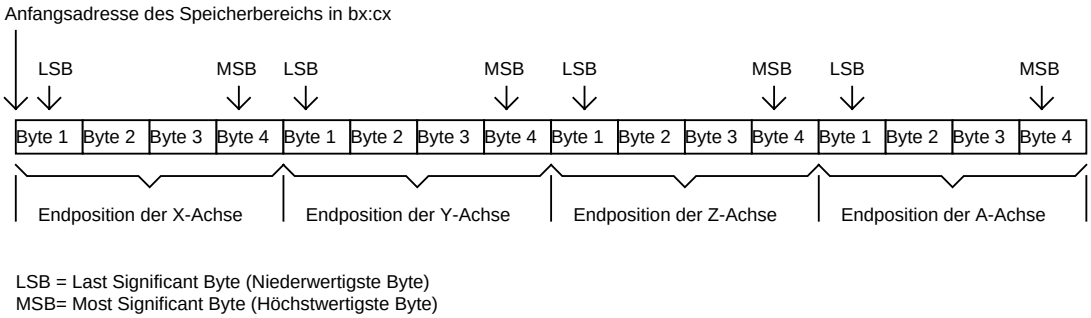


Bild 3.29: Belegung des Speicherbereichs für eine relative Linearbewegung

Jeder Positionswert belegt 4 aufeinander folgende Bytes im Speicherbereich. An der ersten Stelle steht der Wert für die X-Achse. Dann kommen die Werte für Y-; Z- und A-Achse. Der Treiber interpretiert die Positionswerte als ganze Zweier-Komplementär-Zahlen.
Beachten Sie, dass es zwar für Ihre zukünftigen Anwendungen sinnvoll ist, beim Aufruf dieser Funktion 16 Byte Speicher für den Datenaustausch zu reservieren, dies muss aber nicht sein. Wenn Ihre Anlage z. B. nur 3 Achsen hat, brauchen Sie für die 3 Achsen X, Y und Z nur 12 Byte im Speicher zu reservieren.

Während der Abarbeitung des durch diese Funktion übergebenen Bewegungssegments im Hintergrund können Sie den Teach-In-Modus einschalten nachdem die Stop-Funktion aufgerufen wurde. Im Teach-In-Modus können Sie dann andere Bewegungssegmente abfahren lassen. Damit sind die Teach-In-Bewegungen realisierbar. Diese Funktion können Sie im Teach-In-Modus ganz normal aufrufen.
Nach dem Ausschalten des Teach-In-Modus kann der Segmentrest dann ganz normal abgearbeitet werden (siehe Kapitel 3.3.9). Wenn Sie während der Bewegung die Abort-Unterfunktion aufrufen, wird die Anlage genau wie bei der Stop-Unterfunktion angehalten. Bei der Abort-Unterfunktion geht ein evtl. vorhandener Rest des Segments aber unwiderruflich verloren. Es ist unabhän-gig davon, ob der Teach-In-Modus aktiv ist oder nicht (siehe Kapitel 3.3.10).

Falls Ihre Anlage eine TTT-Struktur hat, wird eine Linearinterpolation durchgeführt, d. h. das Werkzeug bewegt sich wirklich auf einer Geraden im Raum bzw. in einer Ebene. Falls diese angenehme Struktur nicht vorliegt, realisiert der Treiber beim Aufruf dieser Funktion eine Synchron-PTP-Steuerung für Sie (siehe Kapitel 1.2 und 3.2.1).
Ausführbarkeit während der Bewegung: Ja, aber nur nachdem der Teach-In-Modus eingeschaltet ist, sonst Nein.
Mögliche Fehlercodes im Normalbetrieb: 0; 2; 3; 4; 7; 8; 9; 15; 19; 22
Mögliche Fehlercodes im Teach-In-Modus: 0; 2; 4; 9
Siehe auch: Kapitel 3.2.1; 3.2.2; 3.2.6; 3.2.9; 3.2.13; 3.3.10; 3.2.17; 3.3.9

3.3.25 Funktion 25: Absolute Linearnormalbewegung

Zweck	Ausführen einer absoluten Linearbewegung mit der Segment-Geschwindigkeit	
Aufrufparameter	ax = 25	Funktionsnummer
	bx	Segment-Adresse der Endposition
	cx	Offset-Adresse der Endposition
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Mit dieser Funktion sind Sie in der Lage, die Linearinterpolation bis zu 4 Achsen durchzuführen. Abgesehen davon, dass die über bx:cx übergebenen Endpositionen Absolutpositionen sind, ist diese Funktion in Bezug sowohl auf die Funktionalität als auch auf das Datenübergabeformat absolut identisch zu der Funktion 24. Im Kapitel 3.2.1 haben wir Ihnen bereits erklärt, dass Sie bei der Berechnung der Absolutpositionen immer den Werkstück-Nullpunkt als den Bezugspunkt nehmen müssen.

Ähnlich wie bei der Funktion 24 schreiben Sie die Endpositionen in den von Ihnen in Ihrem Anwenderprogramm reservierten Speicherbereich, dessen Anfangsadresse dann in bx:cx abgelegt wird. Jeder Positionswert belegt wieder 4 Byte in diesem Speicherbereich (siehe Bild 3.29). Die Einheit des Positionswertes ist entweder Mikrometer [μm] oder Bogensekunde ['']. Welche Einheit dabei benutzt wird, hängt von der Struktur Ihrer Anlage ab (siehe Kapitel 3.2.9).

Bei dieser Funktion wird die Soll-Geschwindigkeit für das Werkzeug auf die gleiche Art und Weise wie bei der Funktion 24 berechnet. Auch hier ist zwischen dem Normalbetrieb und dem Teach-In-Modus zu unterscheiden. Teach-In-Bewegungen können Sie auch hier problemlos realisieren, nachdem der Teach-In-Modus eingeschaltet ist. Diese Funktion können Sie im Teach-In-Modus ganz normal aufrufen.

Falls Ihre Anlage eine TTT-Struktur hat, wird eine Linearinterpolation durchgeführt, d. h. das Werkzeug bewegt sich wirklich auf einer Geraden im Raum bzw. in einer Ebene. Falls diese Struktur nicht vorliegt, realisiert der Treiber beim Aufruf dieser Funktion eine Synchron-PTP-Steuerung für Sie (siehe Kapitel 1.2 und 3.2.1).
Ausführbarkeit während der Bewegung: Ja, aber nur nachdem der Teach-In-Modus eingeschaltet ist, sonst Nein.
Mögliche Fehlercodes im Normalbetrieb: 0; 2; 3; 4; 7; 8; 9; 15; 19; 22
Mögliche Fehlercodes im Teach-In-Modus: 0; 2; 4; 9
Siehe auch: Kapitel 3.2.1; 3.2.2; 3.2.6; 3.2.9; 3.2.13; 3.2.15; 3.3.9; 3.3.10; 3.3.24

3.3.26 Funktion 26: Relative Linear-Eilbewegung

Zweck	Ausführen einer relativen Linearbewegung mit der Eilgeschwindigkeit	
Aufrufparameter	ax = 26	Funktionsnummer
	bx	Segment-Adresse der Endposition
	cx	Offset-Adresse der Endposition
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Mit dieser Funktion können Sie die Linearinterpolation für bis zu 4 Achsen realisieren. Vor dem Aufruf müssen Sie die Endpositionen des zu fahrenden Bewegungssegmentes in den von Ihnen in Ihrem Anwenderprogramm reservierten Speicherbereich ablegen, dessen Anfangsadresse Sie dem Treiber durch bx:cx übergeben (siehe Bild 3.29).
Diese Funktion ist absolut identisch zur Funktion 24 mit einer Ausnahme, dass die Eilgeschwindigkeit anstelle der Bahngeschwindigkeit steht.

Nach dem Laden oder nach einem Reset des Treibers hat die Eilgeschwindigkeit ihren Standardwert, den Sie mit dem mitgelieferten Konfigurationsprogramm eingestellt haben. Durch den Aufruf der Funktion 23 sind Sie in der Lage, diese Geschwindigkeit zu ändern. Es wird zwischen dem Normalbetrieb und dem Teach-In-Modus unterschieden.
Im Normalbetrieb können Sie mit der Funktion 21 den Änderungsfaktor Kv und damit auch die Werkzeuggeschwindigkeit während der Bewegung ändern. Im Teach-In-Modus ist die Werkzeug-Soll-Geschwindigkeit immer gleich der Teach-In-Geschwindigkeit. Der Faktor Kv hat in diesem Modus keine Wirkung. Diese Funktion können Sie im Teach-In-Modus *nicht* aufrufen.

Diese Funktion ist besonders nützlich, wenn Sie einen Eilgang (Eilbewegung) realisieren wollen, um z. B. Ihre Anlage einzurichten oder das Werkzeug während der Bearbeitung zu wechseln oder ein Werkstück zu holen.

Die Endpositionen, die Sie vor dem Funktionsaufruf in den Speicherbereich ablegen, sind relative Koordinaten, d. h. der Bezugspunkt für diese Endpositionen ist der Punkt, in dem das Werkzeug direkt vor dem Funktionsaufruf steht. Beim Fahren mehrerer Segmente hintereinander ist der Bezugspunkt der Endpunkt des letzten Segments und gleichzeitig der Anfangspunkt des zu fahrenden Segments (siehe Kapitel 3.2.1).

Die Einheit des Positionswertes ist entweder Mikrometer [μm] oder Bogensekunde ['']. Welche Einheit dabei benutzt wird, hängt von der Struktur Ihrer Anlage ab (siehe Kapitel 3.2.9).
Falls Ihre Anlage eine TTT-Struktur hat, wird eine Linearinterpolation durchgeführt, d. h. das Werkzeug bewegt sich wirklich auf einer Geraden im Raum bzw. in einer Ebene. Falls diese Struktur nicht vorliegt, realisiert der Treiber beim Aufruf dieser Funktion eine Synchron-PTP-Steuerung für Sie (siehe Kapitel 1.2 und 3.2.1).

Ausführbarkeit während der Bewegung: Nein
Mögliche Fehlercodes: 0; 2; 3; 4; 7; 8; 9; 12; 15; 19; 22
Siehe auch: Kapitel 3.2.1; 3.2.2; 3.2.6; 3.2.9; 3.2.13; 3.2.17; 3.3.9; 3.3.24

3.3.27 Funktion 27: Absolute Linear-Eilbewegung

Zweck	Ausführen einer absoluten Linearbewegung mit der Eilgeschwindigkeit	
Aufrufparameter	ax = 27	Funktionsnummer
	bx	Segment-Adresse der Endposition
	cx	Offset-Adresse der Endposition
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Mit dieser Funktion können Sie die Linearinterpolation von bis zu 4 Achsen realisieren. Diese Funktion ist für die Realisierung von Eilbewegungen gedacht. Abgesehen davon, dass die über bx:cx übergebenen Endpositionen Absolutpositionen sind, ist diese Funktion in Bezug sowohl auf die Funktionalität als auch auf das Datenübergabeformat absolut identisch zur Funktion 26 (siehe Bild 3.29). Bei der Berechnung der Absolutpositionen dient der Werkstück-Nullpunkt als Bezugspunkt.
Bei der Berechnung der Soll-Geschwindigkeit für das Werkzeug wird die Eilgeschwindigkeit anstelle der Bahngeschwindigkeit benutzt, wenn der Teach-In-Modus nicht eingeschaltet ist.
Durch die Änderung des Faktors Kv können Sie im Normalbetrieb die Werkzeuggeschwindigkeit ändern. Im Teach-In-Modus geht das natürlich nicht, weil die Soll-Geschwindigkeit für das Werkzeug immer gleich der Teach-In-Geschwindigkeit ist. Der Faktor Kv hat hier wie immer keine Wirkung. Im Teach-In-Modus können Sie diese Funktion nicht aufrufen.

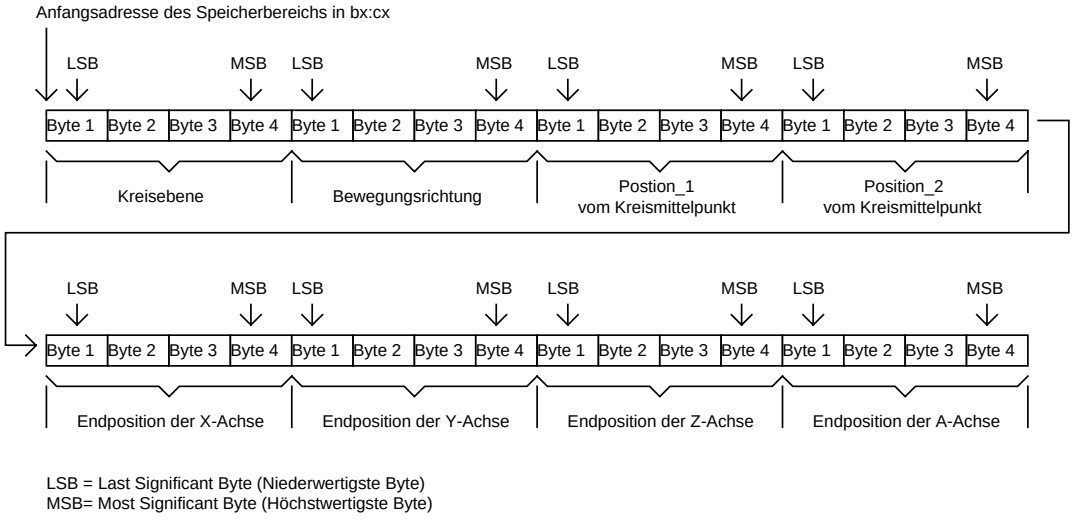
Die Einheit des Positionswertes ist entweder Mikrometer [μm] oder Bogensekunde ['']. Welche Einheit dabei benutzt wird, hängt von der Struktur Ihrer Anlage ab (siehe Kapitel 3.2.9).
Falls Ihre Anlage eine TTT-Struktur hat, wird eine Linearinterpolation durchgeführt, d. h. das Werkzeug bewegt sich wirklich auf einer Geraden im Raum bzw. in einer Ebene. Falls diese Struktur nicht vorliegt, realisiert der Treiber beim Aufruf dieser Funktion eine Synchron-PTP-Steuerung für Sie (siehe Kapitel 1.2 und 3.2.1).

Ausführbarkeit während der Bewegung: Nein
Mögliche Fehlercodes: 0; 2; 3; 4; 7; 8; 9; 12; 15; 19; 22
Siehe auch: Kapitel 3.2.1; 3.2.2; 3.2.6; 3.2.9; 3.2.13; 3.2.17; 3.3.9; 3.3.24; 3.3.26

3.3.28 Funktion 28: Relative Kreisbewegung

Zweck	Ausführen einer relativen Kreisbewegung mit der Segment-Geschwindigkeit	
Aufrufparameter	ax = 28	Funktionsnummer
	bx	Segment-Adresse der Kreisparameter
	cx	Offset-Adresse der Kreisparameter
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Mit dieser Funktion sind Sie in der Lage, eine Kreisinterpolation auf einer der Ebenen XY, YZ, ZX durchzuführen. Im Unterschied zu einer Linearbewegung haben Sie hier in bx:cx die Anfangsadresse eines Speicherbereich, in dem nicht nur die Endpositionen sondern auch die Informationen über die Kreisebene, den Kreismittelpunkt und die Bewegungsrichtung stehen. Das Bild 3.30 zeigt Ihnen, wie die Informationen in diesem Speicherbereich stehen.



LSB = Last Significant Byte (Niederwertigste Byte)
MSB= Most Significant Byte (Höchstwertigste Byte)

Bild 3.30: Belegung des Speicherbereichs für eine Kreisbewegung

Der von Ihnen in Ihrem Anwenderprogramm reservierte Speicherbereich, in dem die Kreisinformationen stehen, ist 32 Bytes groß (aufgrund des Papierformates wird dieser Speicherbereich im Bild 3.30 in zwei Teile geteilt. In Wirklichkeit gehören diese beiden Bereiche zusammen).
Die Position-1 und die Position-2 sind die Positionen des Kreismittelpunkts. In Abhängigkeit von der Kreisebene sind dies die Positionen der X- oder Y- oder Z-Achse. Durch den Wert, der in den ersten 4 Bytes des Speicherbereiches liegt, können Sie dann selbst bestimmen, auf welcher Ebene die Kreisinterpolation stattfinden soll. Die Zuordnung zwischen diesem Wert und den anderen Größen lautet:

Wert	Kreisebene	Position-1	Position-2
0	XY-Ebene	X-Achse	Y-Achse
1	XZ-Ebene	X-Achse	Z-Achse
2	YZ-Ebene	Y-Achse	Z-Achse
(sonst)	YZ-Ebene	Y-Achse	Z-Achse

Wir müssen Sie hier aber unbedingt darauf aufmerksam machen, das Sie nur einen Kreis bekommen können, wenn die Koordinaten der nicht zur Kreisebene gehörenden Achsen gleich 0 sind. Ansonsten bekommen Sie eine Spirale mit dem Grundkreis in der entsprechenden Ebene. In diesem Fall haben Sie dann die sogenannte Helixinterpolation (siehe Kapitel 3.3.30 und 3.3.31).
Für die Kreisinterpolation ist noch die Bewegungsrichtung wichtig.
In die nächsten 4 Bytes direkt hinter dem Wert für die Kreisebene können Sie den Wert setzen, der die Bewegungsrichtung definiert. Die Zuordnung zwischen diesem Wert und der Bewegungsrichtung lautet:

Wert	Bewegungsrichtung
0	Uhrzeigersinn
1	gegen Uhrzeigersinn
(sonst)	gegen Uhrzeigersinn

Wie Sie sicher selbst bemerkt haben, haben wir für jeden Kreisparameter 4 Bytes im Speicherbereich reserviert. Es bestünde hier zwar keine Notwendigkeit für 4 Bytes (besonders bei der Kreisebene und bei der Bewegungsrichtung), aber dadurch erreichen wir eine einheitliche Darstellung für alle Parameter. Alle Kreisparameter werden vom Treiber als ganze Zweier-Komplementär-Zahlen interpretiert.

Beachten Sie, dass es zwar für Ihre zukünftigen Anwendungen sinnvoll ist, beim Aufruf dieser Funktion extra 16 Byte am Ende des Speicherbereichs für die Endpositionen zu reservieren, dies muss aber nicht sein. Wenn Ihre Anlage z. B. nur 3 Achsen hat, brauchen Sie für die Endpositionen der 3 Achsen X, Y und Z nur 12 Byte am Ende des Speicherbereichs zu reservieren.
Die Einheit sowohl der Mittelpunktpositionen Position-1 und Position-2 als auch der Endpunktpositionen der Achse ist entweder Mikrometer [μm] oder

Bogensekunde [‘‘]. Welche Einheit dabei benutzt wird, hängt von der Struktur Ihrer Anlage ab (siehe Kapitel 3.2.9).
Das Wort "Relativ" im Funktionsnamen hat mit dem Bezugspunkt der zu übergebenden Positionen (sowohl für den Kreismittelpunkt als auch für den Endpunkt) etwas zu tun (siehe Kapitel 3.2.1). Der Bezugspunkt für diese Endpositionen ist der Punkt, in dem das Werkzeug direkt vor dem Funktionsaufruf steht. Beim Fahren mehrerer Segmente hintereinander ist der Bezugspunkt der Endpunkt des letzten Segments und gleichzeitig der Anfangspunkt des zu fahrenden Segments (siehe Kapitel 3.2.1).

Im Unterschied zu einem Linearsegment, bei dem der Interpolationsalgorithmus in Abhängigkeit von der aufgerufenen Treiberfunktion (Funktionen 24, 25 oder Funktionen 26, 27) entweder die Segment-Geschwindigkeit (siehe Kapitel 3.3.20) oder die Eilgeschwindigkeit (siehe Kapitel 3.3.23) für seine internen Berechnungen nutzt, kann ein Kreissegment nur mit der Segmentgeschwindigkeit (siehe Kapitel 3.3.20) interpoliert werden. Ein weiterer Unterschied zu einem Linearsegment besteht darin, dass die Kreisinterpolation nur bei einer Anlage mit TTT-Struktur möglich ist (siehe Kapitel 1.2).

Ähnlich wie bei einem Linearsegment können Sie während des Fahrens eines Kreissegmentes den Teach-In-Modus einschalten, nachdem die Stop-Unterfunktion aufgerufen wurde (siehe Kapitel 3.3.9). Im Teach-In-Modus können Sie dann problemlos eine der Funktionen 24 oder 25 aufrufen, um Teach-In-Bewegungen zu realisieren. Nach dem Ausschalten des Teach-In-Modus lässt sich der evtl. vorhandene Rest des Kreissegments weiter wie normal abarbeiten (siehe Kapitel 3.3.9).

Wenn Sie während der Bewegung die Abort-Unterfunktion aufrufen, wird die Anlage genau wie bei der Stop-Unterfunktion angehalten. Ein evtl. vorhandener Rest des Segments geht aber verloren. Dabei wird kein Fehlerflag gesetzt (siehe Kapitel 3.3.9). Im Teach-In-Modus können Sie die Kreisfunktionen *nicht* benutzen.
Während des Fahrens eines Kreissegments können Sie durch einen Aufruf der Funktion 21 den Faktor Kv und damit die Werkzeuggeschwindigkeit ändern.

Ausführbarkeit während der Bewegung: Nein
Mögliche Fehlercodes: 0; 2; 3; 4; 7; 8; 9; 12; 13; 15; 19; 22
Siehe auch: Kapitel 3.2.1; 3.2.2; 3.2.6; 3.2.9; 3.2.13; 3.2.17; 3.3.9; 3.3.10; 3.3.30; 3.3.31

3.3.29 Funktion 29: Absolute Kreisbewegung

Zweck	Ausführen einer absoluten Kreisbewegung mit der Segment-Geschwindigkeit	
Aufrufparameter	ax = 29	Funktionsnummer
	bx	Segment-Adresse der Kreisparameter
	cx	Offset-Adresse der Kreisparameter
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Mit dieser Funktion sind Sie in der Lage, eine Kreisinterpolation auf einer der Ebenen XY, YZ, ZX durchzuführen. Abgesehen davon, dass die über bx:cx übergebenen Positionen des Kreismittelpunktes und des Segment-Endpunktes Absolutpositionen sind, ist diese Funktion in Bezug sowohl auf die Funktionalität als auf das Datenübergabeformat absolut identisch zu der Funktion 28 (siehe Bild 3.30).
Bei der Berechnung der Absolutpositionen dient der Werkstück-Nullpunkt wie immer als Bezugspunkt (siehe Kapitel 3.2.1). In Abhängigkeit von der Anlagenstruktur ist die Einheit der Positionen entweder Mikrometer [μm] oder Bogensekunde ['‘] (siehe Kapitel 3.2.9).
Diese Funktion können Sie im Teach-In-Modus nicht aufrufen.

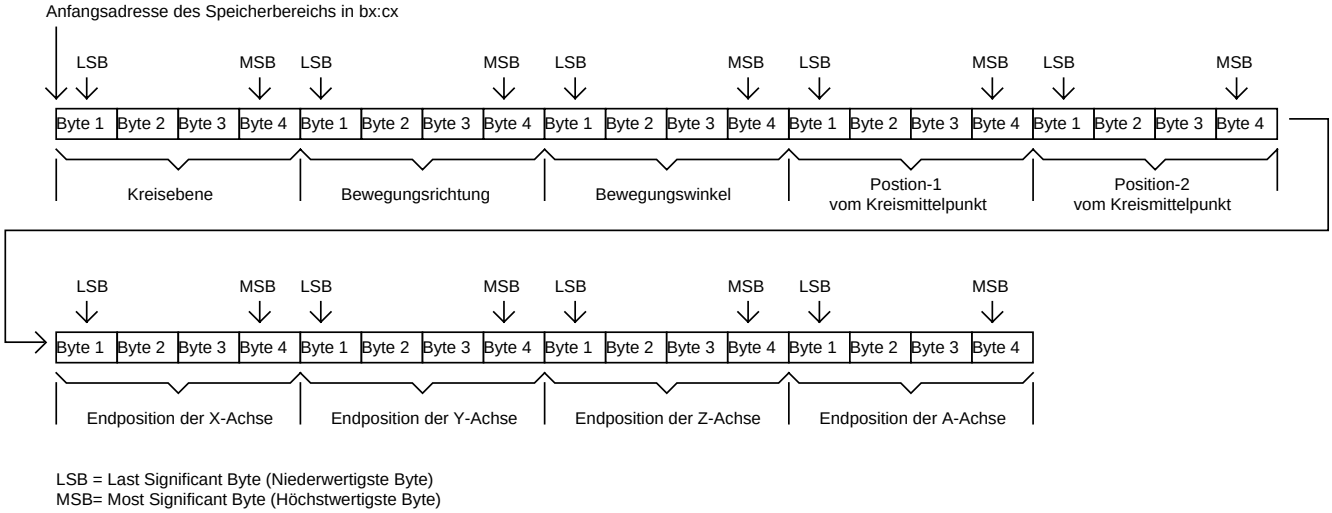
Ausführbarkeit während der Bewegung: Nein
Mögliche Fehlercodes: 0; 2; 3; 4; 7; 8; 9; 12; 13; 15; 19; 22
Siehe auch: Kapitel 3.2.1; 3.2.2; 3.2.6; 3.2.9; 3.2.13; 3.2.17; 3.3.10; 3.3.28; 3.3.30; 3.3.31.

3.3.30 Funktion 30: Relative Helixbewegung

Zweck	Ausführen einer relativen Helixbewegung mit der Segment-Geschwindigkeit	
Aufrufparameter	ax = 30	Funktionsnummer
	bx	Segment-Adresse der Helixparameter
	cx	Offset-Adresse der Helixparameter
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Mit dieser Funktion sind Sie in der Lage, eine Helixinterpolation durchzuführen. Genaugenommen ist die Kreisinterpolation (Funktionen 28 und 29) ein Sonderfall der Helixinterpolation (siehe Kapitel 1.2).
Aufgrund der Angabe vom Kreisendpunkt durch die kartesischen Koordinaten kann man bei der Kreisinterpolation aber nicht mehr als eine Umdrehung in der Kreisebene bewegen. Diese Beschränkung wird durch die Helixinterpolation (Funktionen 30 und 31) aufgehoben. Ähnlich wie bei der Kreisbewegung

haben Sie hier in bx:cx die Anfangsadresse eines Speicherbereichs, in dem die Parameter für die Helixbewegung stehen. Das Bild 3.31 zeigt Ihnen, wie die Informationen in diesem Speicherbereich stehen.



LSB = Last Significant Byte (Niederwertigste Byte)
MSB= Most Significant Byte (Höchstwertigste Byte)

Bild 3.31: Belegung des Speicherbereichs für eine Helixbewegung

Der von Ihnen in Ihrem Anwenderprogramm reservierte Speicherbereich, in dem die Helixinformationen stehen, ist 36 Bytes groß (Aufgrund des Papierformats wird dieser Speicherbereich im Bild 3.31 in zwei Teil geteilt. In Wirklichkeit gehören diese beiden Bereiche zusammen).
Durch den in den ersten 4 Bytes liegenden Wert für die Kreisebene können Sie selbst bestimmen, auf welcher Ebene der Grundkreis für die Helixinterpolation liegen soll. Die Position-1 und die Position-2 sind die Positionen des Kreismittelpunktes. Zwischen der Kreisebene und diesen Positionen gilt die gleiche Zuordnung wie bei einer Kreisinterpolation:

Wert	Kreisebene	Position-1	Position-2
0	XY-Ebene	X-Achse	Y-Achse
1	XZ-Ebene	X-Achse	Z-Achse
2	YZ-Ebene	Y-Achse	Z-Achse
(sonst)	YZ-Ebene	Y-Achse	Z-Achse

Die Bewegungsrichtung für die Helixinterpolation wird durch den Wert in den nächsten 4 Bytes direkt hinter dem Wert für die Kreisebene definiert. Hier gilt das Gleiche wie bei einer Kreisinterpolation:

Wert	Bewegungsrichtung
0	Uhrzeigersinn
1	gegen Uhrzeigersinn
(sonst)	gegen Uhrzeigersinn

Bei der Helixinterpolation bewegt sich das Werkzeug bekanntlich in der Kreisebene auf einem Kreissegment.

Über den Wert für den Bewegungswinkel kann man selbst definieren, wie groß das Kreissegment auf der Kreisebene sein soll (siehe Bild 3.32).

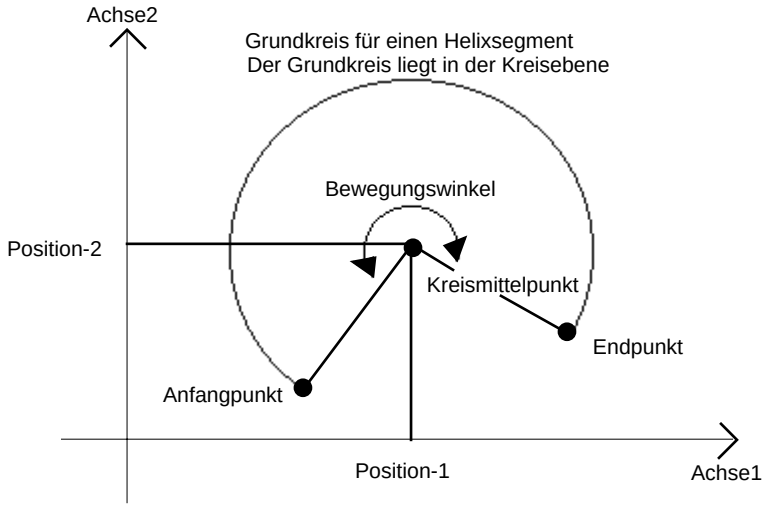


Bild 3.32: Bewegungswinkel für ein Helixsegment

Der Bewegungswinkel ist im Prinzip der Kreiswinkel zwischen dem auf die Kreisebene projezierten Segment-Anfangspunkt (d. h. dem Endpunkt des letzten Segments) und dem ebenfalls auf die Kreisebene projezierten Segment-Endpunkt.
Die Einheit des Bewegungswinkels ist Bogensekunde (siehe Kapitel 3.2.9). Der Bewegungswinkel kann einen beliebigen Wert zwischen 0 ... (2^{31}-1) Bogensekunden annehmen. Weil man die Bewegungsrichtung aber separat definieren kann, darf der Bewegungswinkel deshalb keinen negativen Wert annehmen. Durch die Umrechnung ergibt sich der Werbereich für den Bewegungswinkel:

0 ... 2 147 483 647	**Bogensekunde**	**["]**	**oder**
0 ... 35 791 394,11667	Bogenminute	[']	oder
0 ... 596 523,2352778	Grad	[°]	oder
0 ... 1 657,008986883	Umdrehungen		oder
0 ... 10 411,29452025	Radiant		

Die letzten 16 Byte des im Anwenderprogramm reservierten Speicherbereichs sind für die Übergabe der Endpunktpositionen bestimmt. Die Positionswerte der beiden Achsen, die die Kreisebene bilden, werden vom Treiber einfach ignoriert, weil der Treiber die Positionen für diese beiden Achsen aus dem Bewegungswinkel selbst berechnet. Für diese Achsen können Sie theoretisch beliebige Werte übergeben. Wir empfehlen, den Wert 0 für diese Achsen zu übergeben.

Es sei ausdrücklich betont, dass Sie die Übergabe für diese beiden Achsen nicht weglassen dürfen, obwohl die übergebenen Werte vom Treiber nicht bewertet werden.

Die Reihenfolge für die Übergabe der Achskoordinaten ist ähnlich wie bei der Kreisinterpolation, d. h. zuerst die Koordinate für die X-Achse, zuletzt die Koordinate für die A-Achse. Sie sollten beachten, dass es zwar für Ihre zukünftigen Anwendungen sinnvoll ist, beim Aufruf dieser Funktion extra 16 Byte am Ende des Speicherbereichs für die Endpositionen zu reservieren, dies muss aber nicht sein. Wenn Ihre Anlage z. B. nur 3 Achsen hat, brauchen Sie für die Endpositionen der 3 Achsen X, Y und Z nur 12 Byte am Ende des Speicherbereichs zu reservieren. Die Einheit der Mittelpunktpositionen Position-1 und Position-2 sowie der Endpunktpositionen der die Kreisebene bildenden beiden Achsen ist Mikrometer [μm]. Die Einheit der Endpositionen der beiden anderen Achsen ist entweder Mikrometer [μm] oder Bogensekunde ["].
Welche Einheit dabei benutzt wird, hängt davon ab, ob die Achse eine Linearachse oder eine Drehachse ist (siehe Kapitel 3.2.9).

Das Wort "Relativ" im Funktionsnamen hat mit dem Bezugspunkt der zu übergebenden Positionen (sowohl für den Mittelpunkt des Grundkreises als auch für den Endpunkt) etwas zu tun (siehe Kapitel 3.2.1).
Der Bezugspunkt für diese Positionen ist der Punkt, in dem das Werkzeug direkt vor dem Funktionsaufruf steht. Beim Fahren mehrerer Segmente hintereinander ist der Bezugspunkt der Endpunkt des letzten Segments und gleichzeitig der Anfangspunkt des zu fahrenden Segments (siehe Kapitel 3.2.1).

Im Unterschied zu einem Linearsegment, bei dem der Interpolationsalgorithmus in Abhängigkeit von der aufgerufenen Treiberfunktion (Funktionen 24, 25 oder Funktionen 26, 27) entweder die Segment-Geschwindigkeit (siehe Kapitel 3.3.20) oder die Eilgeschwindigkeit (siehe Kapitel 3.3.23) für seine internen Berechnungen nutzt, kann ein Helixsegment ähnlich wie bei einem Kreissegment nur mit der Segment-Geschwindigkeit (siehe Kapitel 3.3.20) interpoliert werden. Ein anderer Unterschied zu einem Linearsegment besteht darin, dass die Helixinterpolation nur bei einer Anlage mit XY-TTT- oder mit XYZ-TTT-Struktur möglich ist (siehe Kapitel 1.2).

Ähnlich wie bei einem Linearsegment können Sie während des Fahrens eines Helixsegments den Teach-In-Modus einschalten, nachdem die Stop-Funktion aufgerufen wurde (siehe Kapitel 3.3.9). Im Teach-In-Modus können Sie dann problemlos eine der Funktionen 24 oder 25 aufrufen, um Teach-In-Bewegungen zu realisieren. Nach dem Ausschalten des Teach-In-Modus lässt sich der evtl. vorhandene Rest des Helixsegments weiter wie normal abarbeiten (siehe Kapitel 3.3.9).
Falls Sie während der Bewegung die Abort-Unterfunktion aufrufen, wird die Anlage genau wie bei der Stop-Funktion angehalten. Hier geht ein evtl. vorhandener Rest des Segments aber verloren. Es wird kein Fehlerflag gesetzt

(siehe Kapitel 3.3.10). Im Teach-In-Modus können Sie keine Helixfunktionen aufrufen.
Während des Fahrens eines Helixsegments können Sie durch einen Aufruf der Funktion 21 den Faktor Kv und damit die Werkzeuggeschwindigkeit ändern.

Ausführbarkeit während der Bewegung: Nein
Mögliche Fehlercodes: 0; 2; 3; 4; 7; 8; 9; 12; 15; 18; 19; 22
Siehe auch: Kapitel 3.2.1; 3.2.2; 3.2.3; 3.2.6; 3.2.13; 3.2.17; 3.3.9; 3.3.10; 3.3.28; 3.3.29

3.3.31 Funktion 31: Absolute Helixbewegung

Zweck	Ausführen einer absoluten Helixbewegung mit der Segment-Geschwindigkeit	
Aufrufparameter	ax = 31	Funktionsnummer
	bx	Segment-Adresse der Kreisparameter
	cx	Offset-Adresse der Kreisparameter
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Mit dieser Funktion sind Sie in der Lage, eine Helixinterpolation mit dem Grundkreis auf einer der Ebenen XY, YZ, ZX durchzuführen.

Abgesehen davon, dass die über bx:cx übergebenen Positionen des Kreismittelpunkts und des Segmentendpunktes Absolutpositionen sind, ist diese Funktion in Bezug sowohl auf die Funktionalität als auf das Datenübergabeformat absolut identisch zu der Funktion 30 (siehe Bild 3.31).

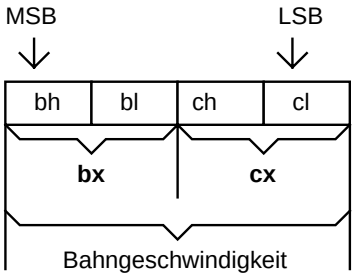
Bei der Berechnung der Absolutpositionen dient der Werkstück-Nullpunkt wie immer als Bezugspunkt (siehe Kapitel 3.2.1). In Abhängigkeit von der Achsenart (Linear- oder Drehachse) ist die Einheit der Positionen entweder Mikrometer [μm] oder Bogensekunde (") (siehe Kapitel 3.2.9).
Der Bewegungswinkel ist ähnlich wie bei einem relativen Helixsegment weiterhin der Kreiswinkel zwischen dem auf die Kreisebene projezierten Segment-Anfangspunkt (d. h. dem Endpunkt des letzten Segments) und dem ebenfalls auch auf die Kreisebene projezierten Segment-Endpunkt.
Die Einheit des Bewegungswinkels ist Bogensekunde (siehe Kapitel 3.2.9).
Diese Funktion können Sie im Teach-In-Modus nicht aufrufen.

Ausführbarkeit während der Bewegung: Nein
Mögliche Fehlercodes: 0; 2; 3; 4; 7; 8; 9; 12; 15; 18; 19; 22
Siehe auch: Kapitel 3.2.1; 3.2.2; 3.2.6; 3.2.9; 3.2.13; 3.2.17; 3.3.9; 3.3.10; 3.3.28; 3.3.29

3.3.32 Funktion 32: Setzen der Bahngeschwindigkeit

Zweck	Setzen einer neuen Bahngeschwindigkeit	
Aufrufparameter	ax = 32	Funktionsnummer
	bx:cx	Bahngeschwindigkeit
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Mit dieser Funktion können Sie eine neue Geschwindigkeit des Werkzeugs für das Bahnfahren definieren (siehe Kapitel 3.3.33). Solange Sie diese Funktion nach dem Laden oder nach einem Reset des Treibers noch nicht aufgerufen haben, nimmt die Werkzeuggeschwindigkeit während des Bahnfahrens ihren Standardwert an, den Sie über das Konfigurationsprogramm PARKON.EXE eingestellt haben (siehe Handbuch PARKON). Die neue Bahngeschwindigkeit, die Sie dem Treiber übergeben wollen, steht in bx:cx (siehe Bild 3.33).



LSB = Last Significant Byte (Niederwertigste Byte)
MSB= Most Significant Byte (Höchstwertigste Byte)

Bild 3.33: Die neue Bahngeschwindigkeit in bx:cx

Diese Funktion können Sie während des Bahnfahrens aufrufen. Der neue Wert der Bahngeschwindigkeit gilt aber nicht mehr für die momentane Bahn, sondern für die nächste Bahn, die evtl. noch gefahren wird. Der Treiber interpretiert die Zahl in bx:cx als eine ganze Zweier-Komplementär-Zahl. Eine negative Zahl wird aber nicht akzeptiert. Die Einheit der Bahngeschwindigkeit ist Mikrometer/Sekunde [μm/s].

Ausführbarkeit während der Bewegung: Ja, der neue Wert wird erst beim Fahren der nächsten Bahn angenommen.
Mögliche Fehlercodes: 0; 2; 3; 7; 8; 9; 15; 17; 19; 22
Siehe auch: Kapitel 3.2.3; 3.2.4; 3.2.5; 3.2.9; 3.2.10; 3.2.17; 3.3.19; 3.3.20; 3.3.22.

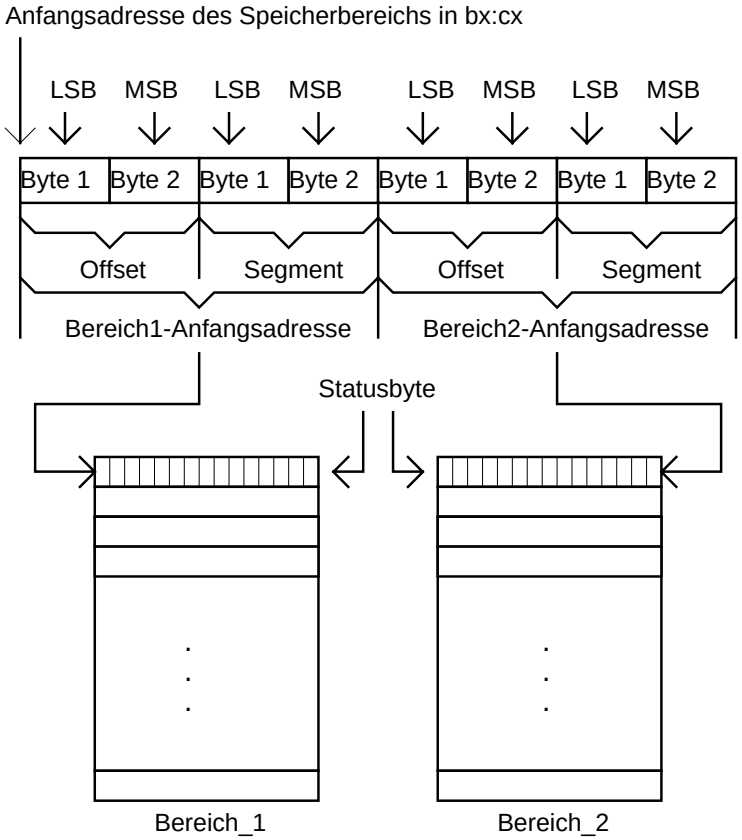
3.3.33 Funktion 33: Bahnbewegung

Zweck	Ausführen einer Bahnbewegung mit der Bahngeschwindigkeit	
Aufrufparameter	ax = 33	Funktionsnummer
	bx	Segment-Adresse des Speicherbereichs
	cx	Offset-Adresse des Speicherbereichs
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Mit dieser Funktion sind Sie in der Lage, eine Kontur mit der Bahnbearbeitung zu erzeugen. Diese Funktion ist die komplexeste Funktion, die der Treiber Ihnen anbietet.

In den Kapiteln 3.2.4 und 3.2.5 haben Sie bereits die Prinzipien der Bahnbearbeitung kennengelernt. Dort wurde bereits erklärt, wie das Geschwindigkeitsprofil berechnet werden kann und wie die Bahndaten in einer Datei abgespeichert sind.

Mit Hilfe dieser Funktion müssen Sie dem Treiber die auf dieser Weise berechneten Bahndaten übergeben, um das Bahnfahren zu ermöglichen. Über die Register bx:cx übergeben Sie dem Treiber die Anfangsadresse eines im Anwenderprogramm reservierten Speicherbereich, in dem wiederum Anfangsadressen von zwei im Anwenderprogramm reservierten Speicherbereichen stehen (siehe Bild 3.34).



LSB = Last Significant Byte (Niederwertigste Byte)
MSB= Most Significant Byte (Höchstwertigste Byte)

Bild 3.34: Organisation der Speicherbereiche für die Übergabe der Bahndaten

Über die beiden Speicherbereiche *Bereich-1* und *Bereich-2* übergibt das Anwenderprogramm dem Treiber die notwendigen Bahndaten. Das erste Byte des jeweiligen Bereichs ist das Statusbyte, das kennzeichnet, ob der jeweilige Bereich voll mit Bahndaten oder leer ist. Die Zuordnung zwischen dem Wert in den Statusbytes und dem Zustand der Bereiche lautet:

Statusbyte	Zustand
0	leer
1	voll
(sonst)	voll

Die Bahndaten dürfen erst ab dem zweiten Byte im jeweiligen Speicherbereich stehen. Bevor diese Funktion aufgerufen wird, muss das Anwenderprogramm die Bahndaten aus der mit Hilfe von *BahnDatenGenerator()* erzeugten Datendatei lesen und diese Bahndaten unverändert in die beiden Speicherbereiche schreiben.
Es ist bekannt, dass mehrere bahngesteuerte Konturen in einer Datendatei existieren können. Jede bahngesteuerte Kontur fängt mit dem Befehl PATH an und endet mit dem Befehl PATHEND. Auf der Zeile direkt hinter dem Befehl PATH steht die Anzahl der Kontursegmente.

Diese Information gehört nicht zu den eigentlichen Bahndaten. Sie ist viel mehr eine Hilfsinformation, die die Funktion *BahnDatenGenerator()* dem Anwenderprogramm zur Verfügung stellt. Deswegen darf diese Zeile dem Treiber gar nicht übergeben werden.
Die eigentlichen Bahndaten fangen erst ab der nächsten Zeile an, auf der die lokale Nummer des ersten Segments steht. Die Bahndaten für jedes Segment betragen genau 140 Byte und in 3 Zeilen geteilt.
Das Anwenderprogramm muss die Bahndaten segmentweise lesen, d. h. die Bahndaten für einen Segment dürfen nicht geteilt werden. Daraus folgt, dass die Anzahl der Datenbytes in den Speicherbereichen *Bereich-1* und *Bereich-2* immer ein Vielfaches von 140 Byte ist. Nachdem die beiden Bereiche mit Daten gefüllt sind, müssen die Statusbytes gleich 1 gesetzt werden, um die Bereiche als voll zu kennzeichnen (es kann natürlich auch sein, dass die Kontur so wenig Segmente hat, dass das Anwenderprogramm nur den ersten Speicherbereich *Bereich-1* braucht).

Der Aufruf dieser Funktion aktiviert den Treiber. Ab diesem Zeitpunkt arbeitet der Treiber selbständig im Hintergrund. Die Kontrolle über den Rechner hat weiterhin das Anwenderprogramm. Über die Register bx:cx ermittelt der Treiber zuerst die Anfangsadressen der beiden Speicherbereiche *Bereich-1* und *Bereich-2*. Mit Hilfe dieser Anfangsadressen ist der Treiber in der Lage, die Bahndaten aus den Speicherbereichen zu lesen und das Bahnfahren zu generieren.
Beachten Sie unbedingt, dass der Treiber immer mit den Daten im ersten Speicherbereich *Bereich-1* anfängt.
Wenn alle Daten in diesem Bereich abgearbeitet sind, schaltet der Treiber auf den zweiten Bereich. In der Zeit, in der der Treiber mit den Daten aus dem zweiten Bereich arbeitet, kann das Anwenderprogramm parallel dazu weitere Bahndaten lesen und den ersten Bereich ausfüllen. Wenn der zweite Bereich irgendwann leer wird, schaltet der Treiber wieder auf den ersten Bereich um. Es geht immer so weiter mit dem Umschalten zwischen den beiden Bereichen, bis die Bahndaten zu Ende sind.

Wie dieses Spiel im Detail aussieht, wollen wir Ihnen Im Folgenden erläutern. Die Bahndaten in dem jeweiligen Speicherbereich werden vom Treiber nacheinander gelesen. Bei Umschalten auf einen neuen Bereich kontrolliert der Treiber immer zuerst das dazu gehörige Statusbyte. Hier gibt es zwei Fälle zu unterscheiden.

a) Falls das Statusbyte einen leeren Speicherbereich zeigt, interpretiert der Treiber diesen Zustand als ein Bahndaten-Nachspeichern-Fehler (Fehlercode 19). Die Bewegung wird sofort unterbrochen. Der Treiber deaktiviert sich selbst. Die Funktion ist somit beendet.

b) Wenn das Statusbyte einen vollen Speicherbereich zeigt, fängt der Treiber an, die Bahndaten segmentweise zu lesen und abzuarbeiten.
Bei jedem Segment liest der Treiber zuerst nur die lokale Segmentnummer. Der gelesene Wert entscheidet dann über den weiteren Ablauf.

- Falls die lokale Segmentnummer größer als 0 ist, liest der Treiber die zu diesem Segment gehörigen Bahndaten. Mit Hilfe dieser Bahndaten wird das Segment abgearbeitet. Wenn dieses Segment fertig ist, liest der Treiber wieder die lokale Segmentnummer des nächsten Segments. Es wiederholt sich bis der Bereich leer wird.

- Eine lokale Segmentnummer gleich -1 wird vom Treiber als das Bahnende interpretiert. Der Treiber deaktiviert sich selbst. Das Bahnfahren ist somit fehlerfrei abgeschlossen.

- Eine lokale Segmentnummer gleich 0 bedeutet, dass dieser Speicherbereich zu Ende ist. Das Statusbyte des momentan aktiven Speicherbereich wird mit 0 geladen, um dem Anwenderprogramm mitzuteilen, dass der Speicherbereich wieder leer ist. Der Treiber wechselt den Speicherbereich. Als erstes kontrolliert der Treiber bei dem neuen Speicherbereich wieder das Status-Byte. Das Spiel geht wieder von vorne los und wiederholt sich, bis die Bahn zu Ende ist.

Während des Bahnfahrens hat das Anwenderprogramm die volle Kontrolle über den Rechner. Der Treiber erledigt seine Aufgabe im Hintergrund.
Während des Bahnfahrens muss das Anwenderprogramm dafür sorgen, dass die beiden Speicherbereiche immer voll mit Bahndaten sind. Das kann erledigt werden, in dem das Anwenderprogramm die beiden Status-Bytes ständig abfragt. Wenn einer der Bereiche leer ist, muss das Anwenderprogramm die Daten aus der Datei lesen und in den leeren Bereich schreiben. Danach wird das Status-Byte dieses Speicherbereichs gleich 1 gesetzt, um dem Treiber mitteilen zu können, dass der Bereich wieder voll ist.
Um das Ende der Bahndaten in diesem Speicherbereich zu kennzeichnen, muss das Anwenderprogramm direkt hinter den gerade gespeicherten Bahndaten die lokale Segmentnummer gleich 0 schreiben. Falls die bahngesteuerte Kontur zu Ende ist, muss die lokale Segmentnummer gleich -1 hier stehen. Wenn das Anwenderprogramm seine Aufgabe versäumt oder nicht korrekt durchführt, tritt der Bahndaten-Nachspeichern-Fehler (Fehlercode 19) sofort ein.

Die lokale Segmentnummer, die neben den Status-Bytes eine entscheidende Bedeutung für den Ablauf hat, ist in ASCII-Klar-Text-Format dargestellt. Sie ist eine Zeichenkette, die einschließlich der Zeichen CARRIAGE RETURN und LINEFEED genau 12 Zeichen beträgt. Solange Sie aber nur die Bahndaten von der Datei lesen und in die Speicherbereiche schreiben, brauchen Sie sich über das Format der lokalen Segmentnummer keine Gedanken zu machen, weil die lokale Segmentnummer bereits in dem entsprechenden Format dargestellt ist.

Wenn Sie aber das Ende der bahngesteuerten Kontur oder das Ende des Speicherbereichs kennzeichnen wollen, müssen Sie das Format unbedingt beachten. Die Zeichenkette für die lokale Segmentnummer gleich -1 (Ende der bahngesteuerten Kontur) ist "-000000001\r\n". Die Zeichenkette für die lokale Segmentnummer gleich 0 (Ende des Speicherbereichs) ist "0000000000\r\n". Dabei sind die Zeichen **r** und **n** CARRIAGE RETURN und LINEFEED.
Im Bild 3.35 wollen wir Ihnen ein bisschen genauer erklären, wie diese Zeichenketten in den Speicherbereichen stehen müssen.

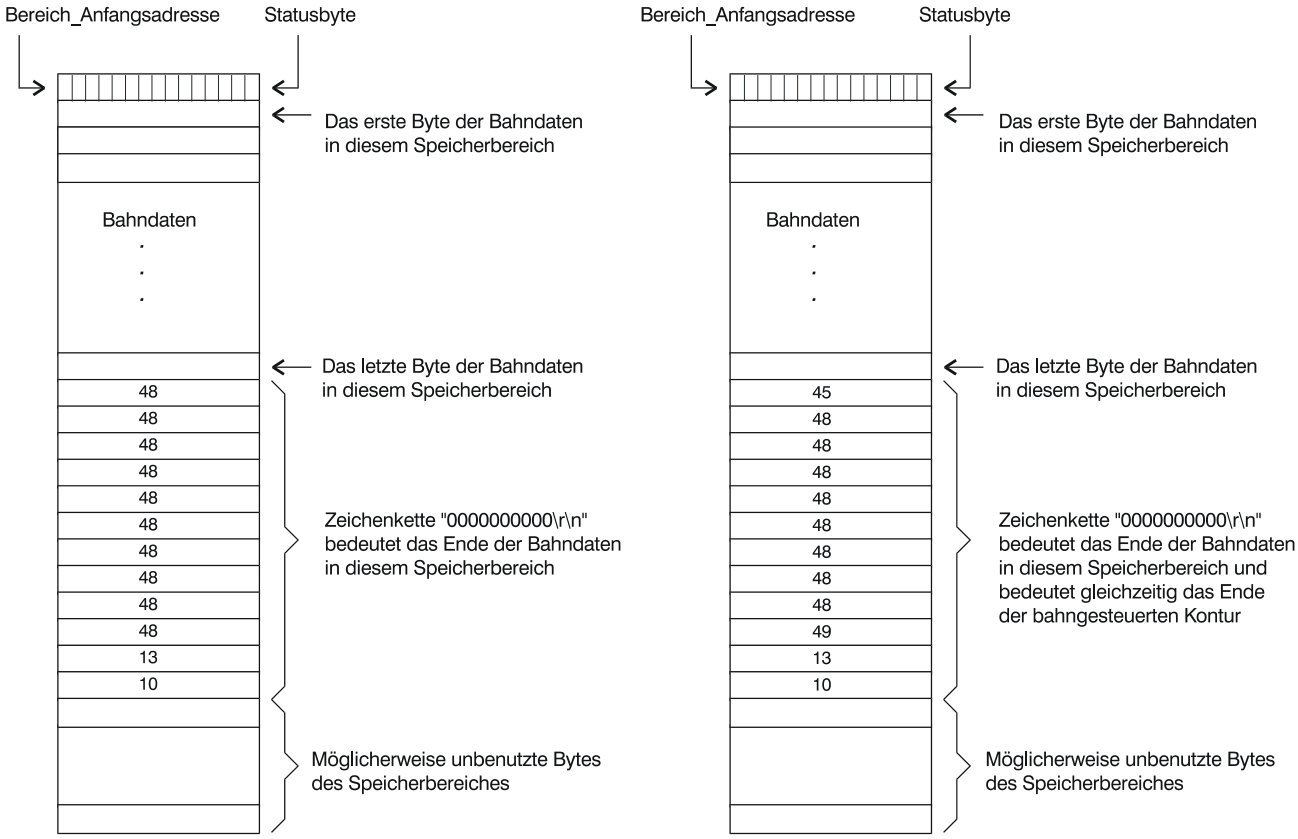


Bild 3.35: Ende eines Speicherbereichs und Ende der bahngesteuerten Kontur

Die beiden Speicherbereiche *Bereich-1* und *Bereich-2* muss das Anwenderprogramm selbst reservieren. Diese Bereiche müssen genügend groß sein. Damit hat das Anwenderprogramm genug Zeit, die Bahndaten aus der Datei zu lesen, den leeren Speicherbereich zu füllen, während der Treiber mit dem anderen Speicherbereich arbeitet. Falls die Speicherbereiche zu klein sind, kann es sehr leicht zum Bahndaten-Nachspeichern-Fehler (Fehlercode 19) kommen. Wir haben mit Speicherbereichen von jeweils 28 012 Bytes (entspricht 200 Segmenten) ganz gute Erfahrungen gemacht.
Je größer die Speicherbereiche sind, desto besser ist es.

Abgesehen vom ständigen Nachladen der Bahndaten aus einer Datei verhält sich das Bahnfahren gegenüber dem Anwenderprogramm genau wie das Segmentfahren, d. h. während des Bahnfahrens kann das Anwenderprogramm problemlos die Ist-Positionen oder die Ist-Geschwindigkeit überwachen (siehe

Kapitel 3.3.18 und 3.3.19). Die Bahngeschwindigkeit lässt sich ohne weiteres mit der Funktion 21 ändern. Selbst der Teach-In-Modus ist während des Bahnfahrens nach einem Aufruf der Stop-Unterfunktion weiterhin möglich (siehe Kapitel 3.3.9. und 3.3.10).
Nach dem Ausschalten des Teach-In-Modus kann der Rest der Bahn problemlos abgearbeitet werden. Genau wie bei der Segmentbearbeitung führt ein Aufruf der Abort-Unterfunktion zu einem rampenmäßigen Anhalten der Anlage. Ein evtl. vorhandener Bahnrest geht verloren. Es wird kein Fehlerflag gesetzt (siehe Kapitel 3.3.10).

Alle Hardware- und Software-Endschalter werden weiterhin im Hintergrund vom Treiber überwacht. Kurz gesagt, alle Ihnen während des Segmentfahrens zur Verfügung stehenden Möglichkeiten können Sie auch während des Bahnfahrens nutzen.
Das gleiche gilt auch für die Beschränkungen. Diese Bahnfunktion können Sie im Teach-In-Modus nicht aufrufen.

Ausführbarkeit während der Bewegung: Nein
Mögliche Fehlercodes: 0; 2; 3; 4; 7; 8; 9; 12; 15; 19; 22
Siehe auch: Kapitel 3.2.3; 3.2.4; 3.2.5; 3.2.6; 3.2.17; 3.3.9; 3.3.10; 3.3.32; 3.3.34

3.3.34 Funktion 34: Abfragen der Bahnparameter

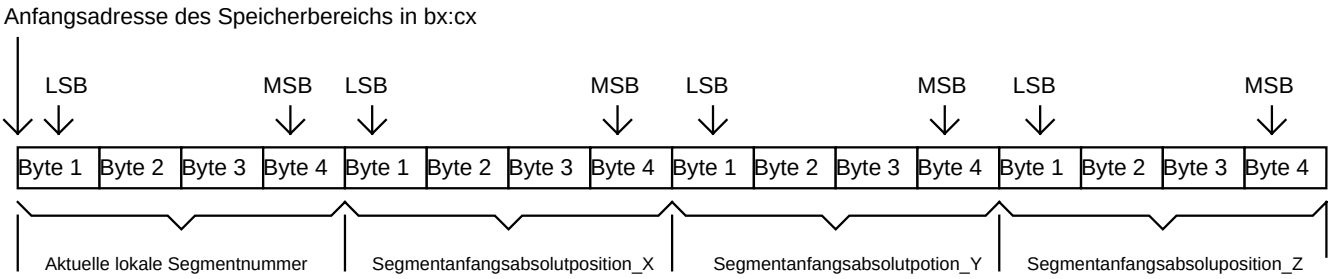
Zweck	Abfragen der aktuellen lokalen Segmentnummer und der Absolutpositionen des aktuellen Segments im Bezug auf den Referenzpunkt der Anlage	
Aufrufparameter	ax = 34	Funktionsnummer
	bx	Segment-Adresse des Speicherbereichs
	cx	Offset-Adresse des Speicherbereichs
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Während des Bahnfahrens werden die Segmente der Konturen nacheinander abgearbeitet. Bevor der Treiber beginnt, ein Segment abzuarbeiten, wird die lokale Nummer dieses Segments abgespeichert. Gleichzeitig werden die Anfangskoordinaten dieses Segments im Bezug auf den Referenzpunkt der Anlage berechnet und ebenfalls vom Treiber abgespeichert. Mit Hilfe dieser Funktionen können Sie jederzeit während des Bahnfahrens diese Informationen des gerade in Bearbeitung befindlichen Segments abfragen. Wenn das Bahnfahren fehlerfrei abgeschlossen ist, werden diese Informationen gelöscht.

Der Sinn dieser Funktion liegt auf der einen Seite darin, dass das Anwenderprogramm jederzeit den Stand der Bearbeitung abfragen kann. Auf der anderen Seite ist diese Funktion eine wertvolle Hilfe im Fehlerfall.
Falls die Bahnbewegung im Fehlerfall (Fräser abgebrochen oder Bahndaten-Nachspeichern-Fehler oder Hardware-Endschalter, ...) unterbrochen wurde, bleiben diese Informationen erhalten. Das Anwenderprogramm kann ohne weiteres diese Informationen abfragen.
Somit ist es möglich, die Bahnbearbeitung nach dem Aufheben des Fehlers an der Stelle fortzusetzen, an der die Bahnbearbeitung unterbrochen wurde.
Diese Informationen werden bei einem Reset (siehe Kapitel 3.3.2) gelöscht.

Beachten Sie, dass die durch einen Aufruf dieser Funktion erhaltenen Absolutpositionen nicht wie üblich den aktuellen Werkstück-Nullpunkt, sondern den Referenzpunkt als Bezugspunkt haben. Das ist deswegen so, weil eine Weiterbearbeitung im Fehlerfall nur nach einem Reset möglich ist. Das Reset löscht aber den zuletzt definierten Werkstück-Nullpunkt. Somit werden die Informationen über die Absolutkoordinaten des Segmentanfangspunktes wertlos, falls der Werkstück-Nullpunkt der Bezugspunkt ist.

Vor einem Aufruf dieser Funktion müssen Sie einen Speicherbereich in Ihrem Anwenderprogramm reservieren. Die Anfangsadresse dieses Speicherbereichs wird dem Treiber über die Register bx:cx übergeben. Die gewünschten Informationen werden vom Treiber in diesen Speicherbereich abgelegt (siehe Bild 3.36).



LSB = Last Significant Byte (Niederwertigste Byte)
MSB= Most Significant Byte (Höchstwertigste Byte)

Bild 3.36: Belegung des Speicherbereichs für das Abfragen der Bahninformationen

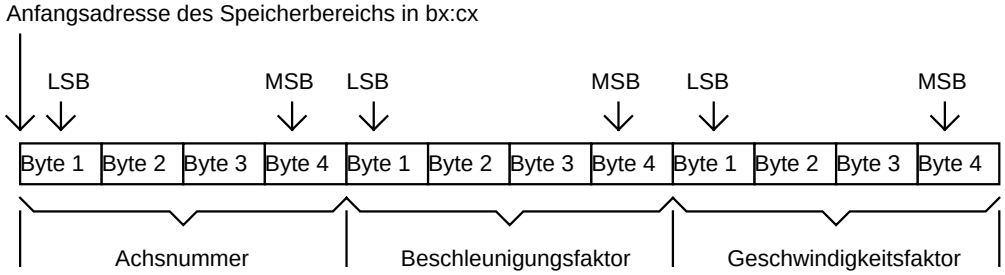
Nach der Referenzfahrt können Sie jederzeit diese Funktion aufrufen. Im fehlerfreien Zustand des Treibers und nicht während des Bahnfahrens bekommen Sie laufend Nullwerte.

Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 2; 3; 4; 7; 8; 9; 12; 15; 18; 19; 22
Siehe auch: Kapitel 3.2.1; 3.2.4; 3.2.5; 3.2.17; 3.3.2; 3.3.11; 3.3.12; 3.3.33

3.3.35 Funktion 35: Änderung der Rampenparameter

Zweck	Änderung der Rampenparameter einer Achse	
Aufrufparameter	ax = 35	Funktionsnummer
	bx	Segment-Adresse der Rampenparameter
	cx	Offset-Adresse der Rampenparameter
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Bei vielen Anwendungsfällen besteht oft der Wunsch, die Rampenparameter der Achsen situationsgerecht zu ändern. Hier denken wir in erster Linie an Aufgaben in Handlingbereichen, wo sich das zu bewegende Gewicht ständig ändert. Es besteht somit die Notwendigkeit, die dynamischen Eigenschaften der Anlagen durch die Änderung der Rampenparameter anzupassen.
Vor dem Aufruf dieser Funktion muss das Anwenderprogramm einen Speicherbereich reservieren, in den die gewünschten Parameteränderungen abgelegt werden. Die Anfangsadresse dieses Speicherbereichs wird dem Treiber über die Register bx:cx übergeben. Im Bild 3.37 können Sie sehen, wie die Belegung dieses Speicherbereichs aussehen muss.



LSB = Last Significant Byte (Niederwertigste Byte)
MSB= Most Significant Byte (Höchstwertigste Byte)

Bild 3.37: Belegung des Speichers für die Rampenparameter-Änderung

Aus dem Feld *Achsnummer* kann der Treiber entnehmen, bei welcher Achse die Änderung vorgenommen werden muss. Pro Aufruf können Sie die Rampenparameter nur bei einer Achse ändern.
Zuordnung zwischen dem Wert in diesen 4 Bytes und der Anlagenachse:

Achsnummer	Anlagenachse
1	X-Achse
2	Y-Achse
3	Z-Achse
4	A-Achse.

Bei seiner Installation liest der Treiber unter anderem die Werte der Maximal-Achsbeschleunigung und der Maximal-Achsgeschwindigkeit aus der Initialisierungsdatei. Diese Werte werden intern im Treiber abgespeichert.

Beim Aufruf dieser Funktion interpretiert der Treiber den Beschleunigungsfaktor und den Geschwindigkeitsfaktor als Prozentsätze dieser intern abgespeicherten Werte. Mit Hilfe dieser Prozentsätze werden die neuen Werte der Maximal-Achsbeschleunigung und der Maximal-Achsgeschwindigkeit berechnet. Um dies zu verdeutlichen, haben wir für Sie ein paar Beispiele der Beschleunigung berechnet:

Maximalachs-beschleunigung aus der Initiali-sierungsdatei	Beschleunigungs-faktor	Berechnung	Maximalachs-beschleunigung nach dem Funktionsaufruf
1 500 000 [µm/s/s]	**5**	1 500 000 * **5 %**	75 000 [µm/s/s]
200 000 [µm/s/s]	**25**	200 000 * **25 %**	40 000 [µm/s/s]
4 000 000 [µm/s/s]	**100**	4 000 000 * **100 %**	4 000 000 [µm/s/s]

Die gleiche Berechnungsweise gilt auch für die Achsgeschwindigkeit:

Maximalachs-Geschwindigkeit aus der Initiali-sierungsdatei	Geschwindigkeits-faktor	Berechnung	Maximalachs-geschwindigkeit nach dem Funktions-aufruf
1 000 000 [µm/s]	**5**	1 000 000 * **5 %**	50 000 [µm/s]
500 000 [µm/s]	**40**	500 000 * **40 %**	200 000 [µm/s]
2 000 000 [µm/s]	**100**	2 000 000 * **100 %**	2 000 000 [µm/s]

Beachten Sie, dass diese Funktion die aus der Initialisierungsdatei entnommenen Werte der Maximal-Achsbeschleunigung und der Maximal-Achsgeschwindigkeit nur als Grundlage für die Berechnung nimmt und diese Werte nicht ändert. Wenn z. B. die aus der Initialisierungsdatei entnommene Maximal-Achsbeschleunigung 1 000 000 µm/s/s beträgt, ist die Maximal-Achsbeschleunigung 300 000 µm/s/s, nachdem Sie diese Funktion mit z. B. einem Beschleunigungswert gleich 30 aufrufen.
Beim nächsten Aufruf dieser Funktion mit z. B. einem Beschleunigungswert gleich 50 hat die Maximal-Achsbeschleunigung den Wert 500 000 µm/s/s (= 1 000 000 µm/s/s * 50 %) nicht aber den Wert 150 000 µm/s/s (= 300 000 µm/s/s * 50 %).

Nach dem Laden oder nach einem Reset des Treibers haben alle Achsen der Anlage die aus der Initialisierungsdatei gelesenen Werte als ihre Maximal-Achsbeschleunigungen bzw. ihre Maximal-Achsgeschwindigkeiten.
Sowohl der Beschleunigungsfaktor als auch der Geschwindigkeitsfaktor können einen Wert zwischen 5 ... 100 annehmen, d. h. zwischen 5 % ... 100 % können Sie die Maximalbeschleunigung sowie die Maximalgeschwindigkeit der Achse variieren. Ein Prozentsatz kleiner als 5 wird ohne eine Fehlermeldung auf 5 gesetzt. Einen Prozentsatz größer als 100 setzt der Treiber ebenfalls auch

ohne Fehlermeldung auf 100 zurück. Daraus ergibt sich, dass Sie in der Initialisierungsdatei maximal mögliche Beschleunigungs- sowie Geschwindigkeitswerte abspeichern sollten, weil Sie mit dieser Funktion die Beschleunigung sowie die Geschwindigkeit im Vergleich zu den aus der Initialisierungsdatei gelesenen Werten nur verkleinern können.
Während der Bewegung, im Teach-In-Modus und im Hand-Modus können Sie diese Funktion nicht aufrufen.

Ausführbarkeit während der Bewegung: Nein
Mögliche Fehlercodes: 0; 2; 3; 4; 7; 8; 9; 10; 12; 15; 19; 22
Siehe auch: Kapitel 3.2.2; 3.2.17

3.3.36 Funktion 36: Lesen eines Bits eines vordefinierten Eingabeports

Zweck	Lesen eines Bits von einem vordefinierten Eingabeport mit Hilfe der logischen Kanalnummer	
Aufrufparameter	ax = 36	Funktionsnummer
	bx	Kanalnummer
	cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx	Inhalt vom Eingabe-Bit
	cx, dx	undefiniert

Kommentar:
Über das Konfigurationsprogramm PARKON.EXE können Sie bis zu 4 Eingabeports mit logischen Nummern versehen (siehe Handbuch PARKON). Der Treiber verwaltet diese Eingabeports sowohl bitweise als auch byteweise. Das Einlesen jedes dieser vordefinierten Eingabeports geschieht über die entsprechende logische Portnummer (0 bis 3). Das Einlesen jedes Bits dieser vordefinierten Eingabeports läuft über die entsprechenden logischen Kanalnummern (0 bis 31). Die Zuordnung zwischen den logischen Portnummern und den logischen Kanalnummern lautet:

Portnummer	Kanalnummer
0	7 bis 0
1	15 bis 8
2	23 bis 16
3	31 bis 24

In jedem Port gilt, je größer die Kanalnummer, desto höherwertiger ist das zugeordnete Bit. Der entscheidende Vorteil eines Zugriffs über die logischen Nummer besteht in der Trennung zwischen einem Anwenderprogramm und der Hardware.
Mit Hilfe dieser Funktion sind Sie in der Lage, jedes beliebige Bit der vordefinierten Eingabeports über die entsprechende Kanalnummer zu lesen.

Beim Aufruf dieser Funktion steht die Kanalnummer im Register bx. Als Ergebnis bekommen Sie den Wert des gewünschten Bits im Register bx.
Einer der möglichen Fehler dieser Funktion ist der Kanal-Fehler (Fehlercode 23). Dieser Fehler deutet daraufhin, dass entweder die Kanalnummer außerhalb des Bereichs 0 bis 31 liegt oder dass die Kanalnummer zwar zu einem vordefinierten Eingabeport gehört, der aber für die Benutzung nicht frei gegeben ist. Definition und Freigabe eines Eingabeports geschieht ausschließlich über das Konfigurationsprogramm PARKON.EXE (siehe Handbuch PARKON).

Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 2; 23
Siehe auch: Kapitel 3.2.17; 3.3.16; 3.3.36; 3.3.37

3.3.37 Funktion 37: Lesen eines vordefinierten Eingabeports

Zweck	Lesen eines Bytes von einem vordefinierten Eingabeport mit Hilfe der logischen Portnummer	
Aufrufparameter	ax = 37	Funktionsnummer
	bx	Portnummer
	cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx	Inhalt vom Eingabeport
	cx, dx	undefiniert

Kommentar:
Die vordefinierten Eingabeports werden von 0 bis 3 nummeriert (siehe Kapitel 3.3.36). Mit Hilfe dieser Funktion können Sie auf einmal alle 8 Kanäle des gewünschten Ports in Form eines Bytes einlesen. Auf diese Weise können Sie Ihren Aufwand im Vergleich zur Funktion 36 reduzieren, falls Sie byteweise arbeiten wollen. Beim Aufruf dieser Funktion steht die Portnummer im Register bx. Als Ergebnis bekommen Sie den Eingabewert des gewünschten Ports im Register bx.
Einer der möglichen Fehler dieser Funktion ist der Portfehler (Fehlercode 24). Dieser Fehler deutet darauf hin, dass entweder die Portnummer außerhalb des Bereichs 0 bis 3 liegt oder dass die Portnummer zwar zu einem vordefinierten Eingabeport gehört, der aber für die Benutzung nicht freigegeben ist. Definition und Freigabe eines Eingabeports geschieht ausschließlich über das Konfigurationsprogramm PARKON.EXE (siehe Handbuch PARKON).

Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 2; 24
Siehe auch: Kapitel 3.2.17; 3.3.16; 3.3.36

3.3.38 Funktion 38: Ausgeben eines Bits an einen vordefinierten Ausgabeports

Zweck	Ausgabe eines Bits an einen vordefinierten Ausgabeport mit Hilfe der logischen Kanalnummer	
Aufrufparameter	ax = 38	Funktionsnummer
	bx	Kanalnummer
	cx	Ausgabewert
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Ähnlich wie bei den Eingabeports können Sie mit Hilfe des Konfigurationsprogramms PARKON.EXE bis zu 4 Ausgabeports definieren, die bitweise über die Kanalnummer und byteweise über die Portnummer zugreifbar sind (siehe Handbuch PARKON). Details über die Kanal- und die Portnummer können Sie im Kapitel 3.3.36 nachlesen. Hier gilt das gleiche wie bei Eingabeports. Neben dem Vorteil einer Trennung zwischen Anwenderprogrammen und der Hardware, hat die Benutzung der logischen Nummer noch einen weiteren Vorteil.
Weil man an einen Port nur byteweise ausgeben kann, stellt die bitweise Ausgabe an Ausgabeports immer ein Problem dar, falls Ausgabeports nicht gepuffert sind, weil man solche Ausgabeports nicht zurücklesen kann.
Aus diesem Grund reserviert der Treiber intern für jeden der 4 vordefinierten Ausgabeports ein Byte, das als ein Zwischenpuffer des jeweiligen Ausgabeports fungiert. Der Wert für jede Ausgabe wird in diesem Puffer abgespeichert. Auf diese Weise ist es für den Treiber kein Problem mehr, bitweise Ausgaben zu realisieren. Es ist unabhängig davon, ob die Ausgabeports rücklesbar sind oder nicht.

Beim Neustart oder beim Aufruf der Funktion 42 werden die Puffer und die Ausgabeports mit den über das Konfigurationsprogramm definierbaren Anfangswerten initialisiert.
Beachten Sie jedoch, dass die Werte der Ausgabeports und deren Puffer bei einem Reset nicht verändert werden.

Mit dieser Funktion sind Sie in der Lage, einen Wert an jedem beliebigen Bit der vordefinierten Ausgabeports über die entsprechende logische Kanalnummer im Bereich von 0 bis 31 auszugeben. Bei einem Aufruf dieser Funktion enthält das Register bx die logische Kanalnummer und das Register cx den Ausgabewert. In Bezug auf den Ausgabewert gilt, dass ein Ausgabewert ungleich Null vom Treiber als ein Ausgabewert gleich 1 betrachtet wird. Einer der möglichen Fehler dieser Funktion ist der Kanalfehler (Fehlercode 23). Dieser Fehler deutet daraufhin, dass entweder die Kanalnummer außerhalb des Bereichs 0 bis 31 liegt oder dass die Kanalnummer zwar zu einem vordefinierten Ausgabeport gehört, der aber für die Benutzung nicht frei gegeben ist. Definition und Freigabe eines

Ausgabeports geschieht ausschließlich über das Konfigurationsprogramm PARKON.EXE (siehe Handbuch PARKON).

Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 2; 23
Siehe auch: Kapitel 3.2.17; 3.3.17; 3.3.36; 3.3.39; 3.3.40; 3.3.41; 3.3.42

3.3.39 Funktion 39: Ausgeben an einen vordefinierten Ausgabeport

Zweck	Ausgabe an einen vordefinierten Ausgabeport mit Hilfe der logischen Portnummer	
Aufrufparameter	ax = 39	Funktionsnummer
	bx	Portnummer
	cx	Ausgabewert
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Die vordefinierten Ausgabeports werden von 0 bis 3 nummeriert (siehe Kapitel 3.3.36). Mit Hilfe dieser Funktion können Sie auf einmal ein Byte an allen 8 Kanälen des gewünschten Ausgabeports ausgeben. Auf diese Weise können Sie Ihren Aufwand im Vergleich zur Funktion 38 reduzieren, falls Sie byteweise arbeiten wollen.
Beim Aufruf dieser Funktion steht die Portnummer im Register bx und der Ausgabewert im Register cx. Einer der möglichen Fehler dieser Funktion ist der Portfehler (Fehlercode 24). Dieser Fehler deutet daraufhin, dass entweder die Portnummer außerhalb des Bereichs 0 bis 3 liegt oder dass die Portnummer zwar zu einem vordefinierten Ausgabeport gehört, der aber für die Benutzung nicht frei gegeben ist. Definition und Freigabe eines Ausgabeports geschieht ausschließlich über das Konfigurationsprogramm PARKON.EXE (siehe Handbuch PARKON).

Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 2; 24
Siehe auch: Kapitel 3.2.17; 3.3.17, 3.3.36; 3.3.38, 3.3.40; 3.3.41; 3.3.42

3.3.40 Funktion 40: Lesen eines Bits eines vordefinierten Ausgabeport

Zweck	Lesen eines Bits von einem vordefinierten Ausgabeport mit Hilfe der logischen Kanalnummer	
Aufrufparameter	ax = 40	Funktionsnummer
	bx	Kanalnummer
	cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx	Inhalt vom Ausgabebit
	cx, dx	undefiniert

Kommentar:
Weil der Treiber für jeden vordefinierten Ausgabeport intern einen Puffer reserviert und weil der Wert im Puffer bei jeder Ausgabe aktualisiert wird, besteht absolut kein Problem, den Wert jedes einzelnen Bits jederzeit abzufragen. Mit Hilfe dieser Funktion können Sie das erledigen.
Beim Aufruf dieser Funktion steht die Kanalnummer im Register bx. Als Ergebnis bekommen Sie den Wert des gewünschten Bits im Register bx.
Einer der möglichen Fehler dieser Funktion ist der Kanalfehler (Fehlercode 23). Dieser Fehler deutet daraufhin, dass entweder die Kanalnummer außerhalb den Bereich 0 bis 31 liegt oder dass die Kanalnummer zwar zu einem vordefinierten Ausgabeport gehört, der aber für die Benutzung nicht frei gegeben ist. Definition und Freigabe eines Ausgabeports geschieht ausschließlich über das Konfigurationsprogramm PARKON.EXE (siehe Handbuch PARKON).

Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 2; 23
Siehe auch: Kapitel 3.2.17; 3.3.36; 3.3.38; 3.3.41

3.3.41 Funktion 41: Lesen eines vordefinierten Ausgabeports

Zweck	Lesen eines Byte von einem vordefinierten Ausgabeport mit Hilfe der logischen Portnummer	
Aufrufparameter	ax = 41	Funktionsnummer
	bx	Portnummer
	cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx	Inhalt vom Ausgabeport
	cx, dx	undefiniert

Kommentar:
Mit Hilfe dieser Funktion sind Sie jederzeit in der Lage, den Wert an jedem vordefinierten Ausgabeport zurückzulesen. Beim Aufruf dieser Funktion steht die Portnummer im Register bx. Als Ergebnis bekommen Sie den Ausgabewert an den gewünschten Ports im Register bx. Einer der möglichen Fehler dieser Funktion ist der Portfehler (Fehlercode 24). Dieser Fehler deutet daraufhin, dass entweder die Portnummer außerhalb des Bereichs 0 bis 3 liegt oder dass die Portnummer zwar zu einem vordefinierten Ausgabeport gehört, der aber für die Benutzung nicht frei gegeben ist. Definition und Freigabe eines Ausgabeports geschieht ausschließlich über das Konfigurationsprogramm PARKON.EXE (siehe Handbuch PARKON).

Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 24
Siehe auch: Kapitel 3.2.17; 3.3.36; 3.3.38; 3.3.40

3.3.42 Funktion 42: Initialisieren aller vordefinierten Ausgabeports

Zweck	Initialisieren aller Ausgabeports, die über die logischen Nummern zugreifbar sind	
Aufrufparameter	ax = 42	Funktionsnummer
	bx, cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Alle vordefinierten Ausgabeports haben ihre Anfangswerte, die über das Konfigurationsprogramm PARKON.EXE definierbar sind (siehe Handbuch PARKON). Beim Laden des Treibers werden diese Anfangswerte an die entsprechenden Ausgabeports ausgegeben, falls diese Ports für die Benutzung freigegeben sind. Die internen Puffer dieser Ausgabeports werden ebenfalls mit diesen Anfangswerten geladen. Bei einem Reset werden sowohl die Ausgabewerte als auch die Pufferwerte nicht geändert. Wenn Sie aus irgendwelchen Gründen den Wunsch haben, alle vordefinierten Ausgabeports auf ihre Anfangswerte zurückzusetzen, ist diese Funktion gerade das richtige Mittel dafür. Diese Funktion braucht außer der Funktionsnummer im Register ax keine weiteren Parameter. Nach einem Aufruf dieser Funktion werden die Anfangswerte an die entsprechenden Ausgabeports ausgegeben, falls sie frei gegeben sind. Die dazugehörigen Puffer werden ebenfalls mit den Anfangswerten geladen, d. h. es ist genauso wie beim Laden des Treibers.

Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 2
Siehe auch: Kapitel 3.2.17; 3.3.17, 3.3.36; 3.3.38; 3.3.39; 3.3.40; 3.3.41

3.3.43 Funktion 43: Ein- und Ausschalten der geschwindigkeitsabhängigen Ausgabe an einem vordefinierten Ausgabeport

Zweck	Ein- und Ausschalten der geschwindigkeitsabhängigen Ausgabe	
Aufrufparameter	ax = 43	Funktionsnummer
	bx	Unterfunktionen
	cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Falls die Ausgabe eines geschwindigkeitsabhängigen Wertes an einem vom Anwender definierten Ausgabeport über das Konfigurationsprogramm PARKON.EXE zugelassen ist, können Sie mit Hilfe dieser Funktion die Ausgabe in Abhängigkeit von dem Wert im Register bx freigeben oder sperren (siehe Kapitel 3.2.8).

Die Bedeutung des Wertes im Register bx ist:

bx	Unterfunktionen
0	Freilassen der Ausgabe
1	Sperren der Ausgabe
sonst	Sperren der Ausgabe.

Nach dem Laden des Treibers sowie nach einem Reset wird die Ausgabe gesperrt. Wenn die Ausgabe einmal freigegeben worden ist, erfolgt die geschwindigkeitsabhängige Ausgabe automatisch bei jeder Bewegung, falls der Teach-In-Modus nicht aktiv ist (siehe Kapitel 3.3.9). Während einer Bewegung können Sie diese Funktion für die Freigabe der Ausgabe nicht aufrufen. Aber das Sperren der Ausgabe können Sie jederzeit aktivieren.
Beim Sperren wird der von Ihnen definierte Minimalwert automatisch ausgegeben. Diese Funktion können Sie nicht aufrufen, falls Sie mit Hilfe des Konfigurationsprogramms PARKON.EXE schon ausdrücklich festgelegt haben, dass die geschwindigkeitsabhängige Ausgabe nicht benutzt wird (siehe Handbuch PARKON).

Ausführbarkeit während der Bewegung: Ja, aber nur für das Sperren der Ausgabe

Mögliche Fehlercodes: 0; 2; 25
Siehe auch: Kapitel 3.2.8; 3.2.17

3.3.44 Funktion 44: Reservieren oder Freilassen eines Datenbytes

Zweck	Reservieren oder Freilassen eines anwendereigenen Datenbytes	
Aufrufparameter	ax = 44	Funktionsnummer
	bx	Geheimcode
	cx	Unterfunktion
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Beim Beenden eines Anwenderprogramms gehen alle Laufzeitinformationen normalerweise verloren. Es gibt aber sehr viele Anwendungsfälle, bei denen es notwendig ist, die Laufzeitinformationen abzuspeichern.
Diese Informationen können dann von anderen Anwenderprogrammen oder vom selben Programm, das die Information abgespeichert hat, beim nächsten Aufruf benutzt werden.

Die Speicherung der Informationen erfolgt normalerweise über solche Speichermedien wie z. B. Festplatten oder Diskette. Das ist meistens sehr aufwendig. Deswegen bietet der Treiber den Anwendern die Möglichkeit, intern im Treiber bis zu 8 Datenbytes für die eigene Benutzung zu reservieren.

Das Reservieren eines Datenbytes erfolgt mit Hilfe dieser Funktion und über den anwendereigenen Geheimcode im Register bx. Der Geheimcode ist notwendig, um das Durcheinander zwischen den verschiedenen Anwenderprogrammen zu vermeiden. Ein Datenbyte, das mit einem bestimmten Geheimcode reserviert wurde, ist nur über diesen Geheimcode ansprechbar sowohl beim Lesen als auch beim Schreiben sowie beim Freilassen.

Der Geheimcode ist ein beliebiger Wert zwischen 1 ... 255. Falls das Anwenderprogramm irgendwann das reservierte Byte nicht mehr braucht, soll dieses Byte auch wieder mit Hilfe dieser Funktion aber nur über den zugehörigen Geheimcode freigelassen werden. Der Wert im Register cx entscheidet darüber, ob ein Datenbyte reserviert oder freigelassen werden soll. Die Bedeutung des Wertes im Register cx ist:

cx	Unterfunktionen
0	Freilassen eines reservierten Bytes
1	Reservieren eines Bytes
sonst	Reservieren eines Bytes.

Beachten Sie, dass selbst ein Reset die Zustände der anwendereigenen Bytes nicht ändern kann. Beim Neustart des Treibers sowie direkt nach dem Freilassen hat das jeweilige Byte den Wert 0.

Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 2; 26
Siehe auch: Kapitel 3.2.17; 3.3.45; 3.3.46

3.3.45 Funktion 45: Lesen eines reservierten Datenbytes

Zweck	Lesen eines reservierten Datenbytes	
Aufrufparameter	ax = 45	Funktionsnummer
	bx	Geheimcode
	cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx	Inhalt des reservierten Bytes
	cx, dx	undefiniert

Kommentar:
Mit Hilfe dieser Funktion können Sie den Inhalt eines von Ihnen reservierten Bytes lesen. Das Lesen eines reservierten Bytes erfolgt ausschließlich über den Geheimcode im Register bx. Im Fall, dass der Treiber den Geheimcode nicht identifizieren kann, bekommt das Anwenderprogramm den Fehlercode 26 zurück.

Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 2; 26
Siehe auch: Kapitel 3.2.17; 3.3.44; 3.3.46

3.3.46 Funktion 46: Schreiben eines reservierten Datenbyte

Zweck	Schreiben eines reservierten Datenbytes	
Aufrufparameter	ax = 46	Funktionsnummer
	bx	Geheimcode
	cx	Wert
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Mit Hilfe dieser Funktion können Sie ein beliebiges Byte (Wertebereich: 0 ... 255) in ein von Ihnen reserviertes Byte einzuschreiben. Über den Geheimcode im Register bx kann der Treiber das von Ihnen reservierte Byte identifizieren. Über diesen Geheimcode können Sie dann mit Hilfe der Funktion 45 den Wert dieses Bytes jederzeit zurücklesen.

Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 2; 26
Siehe auch: Kapitel 3.2.17; 3.3.44; 3.3.45

3.3.47 Funktion 47: Ein- oder Ausschalten des Sleep-Modus

Zweck	Ein- oder Ausschalten des Sleep-Modus der Anlage	
Aufrufparameter	ax = 47	Funktionsnummer
	bx, cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Nur aus Kompatibilitätsgründen zu den Schrittmotorsteuerungen der Firma *iselautomation* wird diese Funktion eingeführt. Ein Aufruf dieser Funktion hat keine Wirkung sowohl auf den Treiber als auch auf den Servocontroller. Sie bekommen im Normalfall ständig den Fehlercode 0 „Fehlerfrei“ zurück.

Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 2
Siehe auch: Kapitel 3.2.17

3.3.48 Funktion 48: Aktivieren/Deaktivieren des Sicherheitskreises

Zweck	Ein- oder Ausschalten des Sicherheitskreises von Servocontrollern	
Aufrufparameter	ax = 48	Funktionsnummer
	bx	Unterfunktionen
	cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Mit Hilfe dieser Funktion können Sie einen definierten Wert an dem PIN 43 des RIBBON-Steckers ausgeben. Falls die Hardware Ihrer Servocontroller nichts anders zulässt, sollten Sie dieses optoisolierte Signal benutzen, um den Sicherheitskreis Ihrer Servocontroller bei der Referenzfahrt oder beim Herausfahren aus einem aktiven Hardware-Endschalter zu überbrücken (siehe Kapitel 2.5 und 2.7).
Das Überbrücken des Sicherheitskreises verhindert das Ausschalten des Servocontrollers bei einem aktiven Hardware-Endschalter. In Abhängigkeit von dem Wert in bx wird der Sicherheitskreis des Servocontrollers aktiviert oder deaktiviert. Die Bedeutung des Werts im Register bx ist:

bx	Unterfunktionen	Wert am PIN 43 des RIBBON-Steckers	Bedeutung
0	Aktivieren des Sicherheitskreises	0	Sicherheitskreis nicht überbrückt
1	Deaktivieren des Sicherheitskreises	1	Sicherheitskreis überbrückt
sonst	Deaktivieren des Sicherheitskreises	1	Sicherheitskreis überbrückt

Sie müssen Ihren Servocontroller so aufbauen, dass ein LOW-Wert am PIN 43 den Sicherheitskreis aktiviert und dass ein HIGH-Wert am PIN 43 den Sicherheitskreis deaktiviert. Beim Starten des Treibers ist der Sicherheitskreis immer aktiviert. Das bedeutet, dass ein LOW-Wert am PIN 43 liegt. Der Zustand, ob der Sicherheitskreis aktiviert oder deaktiviert ist, wird vom Treiber intern in einem Flag abgespeichert.
Die Reset-Funktion hat keinen Einfluss auf diesen Flag. Diesen Flag können Sie nur mit Hilfe dieser Funktion setzen oder zurücksetzen. Es ist möglich, den Zustand dieses Flags jederzeit mit der Funktion 5 abzufragen.
Beachten Sie, dass viele Treiberfunktionen gesperrt sind, falls der Sicherheitskreis überbrückt ist. Aus Sicherheitsgründen sollten Sie so schnell wie möglich den Sicherheitskreis wieder aktivieren.
Aus unserer Sicht sollten Sie diese Funktion bei der Referenzfahrt und bei einer Teach-In-Bewegung benutzen d. h. direkt vor der Referenzfahrt sowie vor dem Einschalten des Teach-In-Modus sollten Sie mit dieser Funktion den Sicherheitskreis überbrücken. Nach der Referenzfahrt sowie nach dem Ausschalten des Teach-In-Modus sollte der Sicherheitskreis sofort wieder aktiviert werden. Während ein Bewegungssegment im Hintergrund aktiv ist, ist der Aufruf dieser Funktion verboten.

Ausführbarkeit während der Bewegung: Nein
Mögliche Fehlercodes: 0; 2; 4
Siehe auch: Kapitel 2.5; 2.7; 2.8; 3.2.15; 3.2.17; 3.3.6; 3.3.9

3.3.49 Funktion 49: Abfragen des Kontrollbytes

Zweck	Abfragen der einzelnen Hardware-Fehlersignale des Kontrollbytes	
Aufrufparameter	ax = 49	Funktionsnummer
	bx, cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx	Kontrollbyte-Status
	cx, dx	undefiniert

Kommentar:
Beim Aufruf dieser Funktion bekommen Sie einen Wert in bx zurück. Wobei das höherwertige Byte von bx, das Register bh, immer gleich Null ist.
Das niederwertige Byte in bl zeigt den Fehlerzustand des Kontrollbytes, ein von Ihnen frei definiertes Eingabeport. Über das Kontrollbyte können bis zu 8 Hardware-Fehlersignale dem Treiber zurückgeführt werden.
Das Kontrollbyte wird regelmäßig in einem Zeitintervall von weniger als 10 ms eingelesen. Der eingelesene Wert wird mit einem von Ihnen als fehlerfrei definierten Wert verglichen. Falls ein Fehlersignal aktiv ist, wird das entsprechende Bit in dem Kontrollbyte gleich 1 gesetzt. Der Wert Null bedeutet, dass es fehlerfrei ist.

Mit dem Programm PARKON können Sie nicht nur die Adresse des Kontrollbytes und den fehlerfreien Wert definieren. Sie können sogar noch definieren, welches Bit des Kontrollbytes benutzt wird. Jedes unbenutzte Bit liefert immer einen Wert gleich Null zurück, d. h. bei einem unbenutzten Bit ist es immer fehlerfrei.
Beachten Sie, dass der bei dieser Funktion zurückgelieferte Wert nicht der wirkliche Wert des als Kontrollbyte definierten Eingabeports ist. Dieser Wert ist nur ein logischer Wert. In einem fehlerfreien Fall von allen hardwaremäßigen Fehlersignalen sind alle Bits des Registers bx gleich Null. Falls ein Fehlersignal aktiv ist, wird das entsprechende Bit gleich 1 gesetzt. Ein interner Flag wird gesetzt und bleibt solange gesetzt, bis die Reset-Funktion aufgerufen wird. Den Zustand dieses Flags können Sie jederzeit mit der Funktion 5 abfragen. Der Wert vom Bit 0 des Registers bh ist der Flagzustand.

Im Unterschied zu diesem Flag zeigt Ihnen das Bit 7 des Registers bl beim Aufruf der Funktion 5 an, ob mindestens eines der Hardware-Fehlersignale momentan aktiv ist oder nicht. Falls ein Hardwarefehler vorhanden ist, ist der Aufruf dieser Funktion die einzige Möglichkeit, um die Fehlerquelle zu lokalisieren.

Die häufigste Anwendung dieses Kontrollbytes liegt in der Überwachung bestimmter Hardwarefehler wie z. B. Stromausfall, Bruch der Encoder-Leitung, etc. Beim Vorhandensein eines Hardwarefehlers wird ein aktives Bewegungssegment oder eine Referenzfahrt sofort unterbrochen. So kann keine unkontrollierte Bewegung entstehen.

Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 2
Siehe auch: Kapitel 3.2.15; 3.2.17; 3.3.5

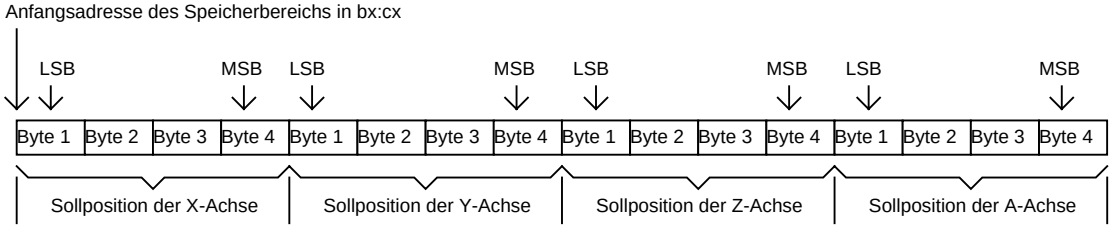
3.3.50 Funktion 50: Abfragen der Soll-Positionen der Achsen im Bezug auf den Werkstück-Nullpunkt

Zweck	Abfragen der aktuellen Soll-Positionen der Achsen im Bezug auf den Werkstück-Nullpunkt	
Aufrufparameter	ax = 50	Funktionsnummer
	bx	Segmentadresse der Soll-Positionen
	cx	Offsetadresse der Soll-Positionen
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Weil die Servomotoren in einem geschlossenen Regelkreis arbeiten, gibt es öfter eine gewisse Abweichung zwischen der Ist- und der Soll-Position. Das Abfragen der Ist-Positionen erfolgt mit Hilfe der Funktion 18. Mit dieser Funktion können Sie die Soll-Positionen der Achsen im Bezug auf den momentanen Werkstück-Nullpunkt abfragen.

Um diese Funktion nutzen zu können, müssen Sie zuerst einen Speicherbereich in Ihrem Anwenderprogramm reservieren. Die Anfangsadresse dieses Speicherbereichs legen Sie beim Aufruf dieser Funktion in bx:cx. Mit Hilfe dieser Anfangsadresse schreibt der Treiber die Achs-Soll-Positionen in den von Ihnen reservierten Speicherbereich (siehe Bild 3.38).

Für jede Achse braucht der Treiber 4 Byte Speicher. Den Wert in diesen 4 Bytes müssen Sie als eine ganze Zweier-Komplementär-Zahl interpretieren. Beachten Sie, dass es zwar für Ihre zukünftigen Anwendungen sinnvoll ist, beim Aufruf dieser Funktion 16 Byte Speicher für den Datenaustausch zu reservieren, dies muss aber nicht sein. Wenn Ihre Anlage z. B. nur 3 Achsen hat, brauchen Sie nur 12 Byte im Speicher für die 3 Achsen X, Y und Z zu reservieren.



LSB = Last Significant Byte (Niederwertigste Byte)
MSB= Most Significant Byte (Höchstwertigste Byte)

Bild 3.38: Belegung des Speichers mit den aktuellen Soll-Positionen der Achsen

In Abhängigkeit von der Anlagenstruktur ist die Einheit der Soll-Positionen entweder Mikrometer [μm] oder Bogensekunde [''].

Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 2; 9
Siehe auch: Kapitel 3.2.9; 3.2.13; 3.2.15; 3.2.17; 3.3.51

3.3.51 Funktion 51: Abfragen der Soll-Positionen der Achsen im Bezug auf den Referenzpunkt

Zweck	Abfragen der aktuellen Soll-Positionen der Achsen im Bezug auf dem Referenzpunkt	
Aufrufparameter	ax = 51	Funktionsnummer
	bx	Segmentadresse der Soll-Positionen
	cx	Offsetadresse der Soll-Positionen
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

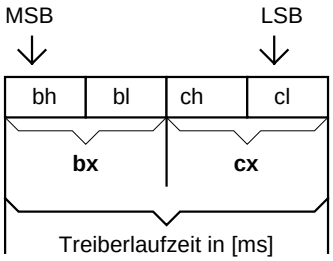
Kommentar:
Genau wie bei der Funktion 50 können Sie hier die Soll-Positionen der Achsen abfragen. Nur die Bezugspunkte der Soll-Positionen unterscheiden sich. Mit dieser Funktion fragen Sie die Soll-Positionen im Bezug auf den Referenzpunkt ab. Falls Sie keinen Werkstück-Nullpunkt definieren, sind der Referenzpunkt und der Werkstück-Nullpunkt identisch, d. h. in diesem Fall liefern die beiden Funktionen 50 und 51 dieselben Ergebnisse.

Die Vorgehensweise beim Aufruf dieser Funktion ist genauso wie bei der Funktion 50, d. h. zuerst müssen Sie einen Speicherbereich in Ihrem Anwenderprogramm reservieren. Die Anfangsadresse dieses Speicherbereichs legen Sie beim Aufruf dieser Funktion in bx:cx fest. Mit Hilfe dieser Anfangsadresse schreibt der Treiber die Achs-Soll-Positionen in den von Ihnen reservierten Speicherbereich (siehe Bild 3.38).
In Abhängigkeit von der Anlagenstruktur ist die Einheit der Soll-Positionen entweder Mikrometer [μm] oder Bogensekunde [''].
Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 2; 9
Siehe auch: Kapitel 3.2.9; 3.2.13; 3.2.15; 3.2.17; 3.3.50

3.3.52 Funktion 52: Abfragen der Treiberlaufzeit

Zweck	Abfragen der Treiberlaufzeit	
Aufrufparameter	ax = 52	Funktionsnummer
	bx, cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx:cx	Treiberlaufzeit in Millisekunde
	dx	undefiniert

Kommentar:
Intern speichert der Treiber die Zeit, die seit seiner Installation verstrichen ist. Beim Aufruf dieser Funktion bekommen Sie die verstrichene Zeit in Millisekunde zurück. Die Genauigkeit der Zeitangabe liegt im Mikrosekunden-Bereich. Dadurch sind Sie in der Lage, eine sehr genaue Zeitbasis zu realisieren. Die verstrichene Zeit wird in bx:cx zurückgeliefert (siehe Bild 3.39).



LSB = Last Significant Byte (Niederwertigste Byte)
MSB= Most Significant Byte (Höchstwertigste Byte)

Bild 3.39: Die Treiberlaufzeit in bx:cx

Sie müssen den Wert in bx:cx als eine ganze Zweier-Komplementär-Zahl interpretieren.

Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 2
Siehe auch: Kapitel 3.2.9; 3.2.13; 3.3.7; 3.2.15; 3.3.16

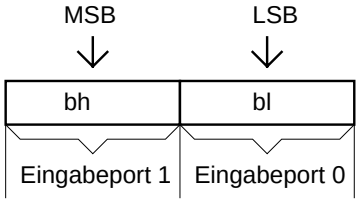
3.3.53 Funktion 53: Lesen der Überwachungseingabeports

Zweck	Lesen der beiden Eingabeports zur Überwachung	
Aufrufparameter	ax = 53	Funktionsnummer
	bx, cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx	Inhalt der Überwachungseingabeports
	cx, dx	undefiniert

Kommentar:
Die Ein- und Ausgabeports sind für das Steuern der Peripherien wie z. B. Werkzeugwechsler, Kühlmittelpumpe, Abdeckhauben, usw. Es kommt öfter vor, dass die Anwender noch zusätzliche Peripherien zur Maschine aufrüsten. Um eine klare Trennung bei der Bedienung zwischen den maschineneigenen Peripherien und den anwendereigenen Peripherien zu erreichen, haben wir noch zwei Ein- und zwei Ausgabeports eingeführt. Das sind die Ports für die Überwachung der Maschinen. Diese Ports sollen nur vom Anwenderprogramm benutzt werden.
Die Bedienung dieser Überwachungsports erfolgt mit den Funktionen 53 bis 56. Die Kanalein- und Kanalausgabeports, die mit den Funktionen 36 bis 42 benutzt werden können, sollen für die Anwender reserviert werden.

Diese klare Trennung verhindert, dass die Anwender irrtümlich die maschineneigenen Peripherien bedienen.

Diese Funktion ermöglicht Ihnen das Lesen der beiden Eingabeports für die Überwachung. Die Adressen dieser beiden Ports können Sie mit dem Programm PARKON definieren. Über das entsprechende Maskebyte können Sie sogar die Benutzung der einzelnen Bits festlegen. Die hier nicht benutzten Bits können Sie bei den Kanaleingabeports benutzen.
Der beim Aufruf dieser Funktion zurückgelieferte Wert liegt im Register bx, wobei der Wert im Register bl dem ersten Eingabeport und der Wert im Register bh dem zweiten Eingabeport entspricht (siehe Bild 3.40).



LSB = Last Significant Byte (Niederwertigste Byte)
MSB= Most Significant Byte (Höchstwertigste Byte)

Bild 3.40: Inhalte der Eingabeports für die Überwachung

Beachten Sie unbedingt, dass die zurückgelieferten Werte logische Werte sind. Mit dem Programm PARKON können Sie für jedes einzelne Bit definieren, ob der LOW-Wert oder der HIGH-Wert aktiv ist. Sie bekommen den HIGH-Wert in einem Bit zurück, falls der von Ihnen als aktiv definierte Wert an dem Bit liegt. Umgekehrt bekommen Sie den LOW-Wert zurück. Für ein von Ihnen als unbenutzt definiertes Bit bekommen Sie auch immer den LOW-Wert zurück. Das folgende Beispiel soll Ihnen verdeutlichen, was hier gemeint ist:

	Port 1	Port 0
Maske	00000000	11110000
Aktivwert	11101101	11001010
Realwert	00001111	01101111
Inhalt von bx	00000000	01010000
	(**bh**)	(**bl**)

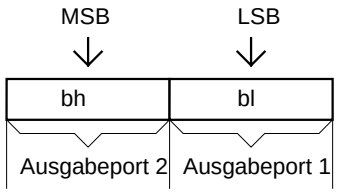
Das zweite Port wird nicht benutzt, weil alle Maskebits gleich 0 sind. Der zurückgelieferte Wert in bh ist immer gleich 0. Die ersten 4 Bits des Ports 0 werden nicht benutzt. Deswegen sind die ersten 4 Bits von bl auch gleich 0. Das Bit 4 hat sowohl den Aktivwert als auch den Realwert (der tatsächlich am Bit liegende Pegel) gleich 0. Der Aktivwert und der Realwert vom Bit 6 sind gleich 1. Daher bekommen Sie in bl den Wert 1 in den entsprechenden Bits. Für das Bit 5 und Bit 7 sind der Aktivwert und der Realwert unterschiedlich. Folglich sind die entsprechenden Bits in bl gleich 0.

Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 2
Siehe auch: Kapitel 3.2.15; 3.2.17

3.3.54 Funktion 54: Ausgabe an den Überwachungsausgabeports

Zweck	Beschreiben die beiden Ausgabeports für die Überwachung	
Aufrufparameter	ax = 54	Funktionsnummer
	bx	Ausgabewert
	cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Ähnlich wie bei den Eingabeports können Sie bei den beiden Ausgabeports für die Überwachung auch die Maskebytes und die Aktivwerte definieren. Die auszugebenden Werte werden beim Aufruf dieser Funktion in das Register bx eingelegt, wobei das Register bl für das erste Port und das Register bh für das zweite Port bestimmt ist (siehe Bild 3.41).



LSB = Last Significant Byte (Niederwertigste Byte)
MSB= Most Significant Byte (Höchstwertigste Byte)

Bild 3.41: Auszugebende Werte für die Überwachungsausgabeports

Die auszugebenden Werte in bx sind logische Werte, d. h. vor der tatsächlichen Ausgabe werden diese Werte zuerst mit den Maskebytes und den Aktivwerten kombiniert. Ein Bitwert gleich 1 wird ausgegeben, falls der Aktivwert und der in bx stehende Wert des entsprechenden Bits gleich sind. Es ist unabhängig davon, ob sie gleich 0 oder 1 sind. Im anderen Fall wird der Wert gleich 0 an das entsprechende Bit ausgegeben.
Ein Bit des Maskebytes gleich 0 bedeutet, dass das entsprechende Bit der Ausgabeports nicht benutzt ist. An unbenutzten Bits werden generell die von Ihnen definierten Initialisierungswerte ausgegeben, wobei die Initialisierungswerte wieder nur logische Werte sind. Bei der Ausgabe an unbenutzten Bits gibt es folgende Ausnahmen:

- Falls alle Bits eines Ausgabeports durch das entsprechende Maskebyte als nicht benutzt definiert sind, wird dieses Port von dem Treiber gar nicht angesprochen, d. h. es erfolgt keine Ausgabe an diesem Port.

- Falls die als unbenutzt definierten Bits eines Überwachungsausgabeports im „Mischbetrieb“ bei den Kanalausgabeports (Funktionen 38 bis 42) benutzt sind, werden die durch Funktionen 38, 39 und 42 festgelegten Werte an diesen Bits ausgegeben. Somit ist die gemeinsame Benutzung eines Ausgabeports sowohl bei dem „Kanal“ als bei der Überwachung möglich. Beachten Sie, dass die Benutzung durch die Überwachung immer eine höhere Priorität hat.

An dem folgenden Beispiel wollen wir das Gesagte verdeutlichen.

	Port 1	Port 0
Maske	00000000	11110000
Initialisierungswert	00001111	11110001
Aktivwert	11101101	11001011
Inhalt von bx	00000000	01010101
	(**bh**)	(**bl**)
Realwert	----------	01100101

An dem zweiten Port wird nicht ausgegeben, weil alle Bits des Maskebytes gleich 0 sind, d. h. alle Bits sind nicht benutzt. Die niederwertigsten 4 Bits des ersten Ports sind nicht benutzt. Deswegen werden die Initialisierungswerte hier ausgegeben. Die Initialisierungswerte sind auch nur logische Werte. Daher werden sie vor der Ausgabe zuerst mit den entsprechenden Aktivwerten kombiniert. Bei den höherwertigsten 4 Bits werden die Bits von bl mit den entsprechenden Bits des Aktivwertes ebenfalls verknüpft. Dadurch entsteht der auszugebende Realwert.

Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 2
Siehe auch: Kapitel 3.2.15; 3.2.17; 3.3.2

3.3.55 Funktion 55: Lesen der Überwachungsausgabeports

Zweck	Lesen der an den Überwachungsausgabeports liegenden Werte	
Aufrufparameter	ax = 55	Funktionsnummer
	bx, cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx	Inhalt der Ausgabeports
	cx, dx	undefiniert

Kommentar:
Jederzeit können Sie mit Hilfe dieser Funktion den Inhalt der für die Überwachung bestimmten Ausgabeports einlesen. Das Ergebnis bekommen Sie im Register bx zurück, wobei der Inhalt des ersten Ports in bl und der des zweiten in bh steht.

Die eingelesenen Werte sind logische Werte. Vor der Rückkehr werden die tatsächlichen Werte an Ausgabeports mit den Aktivwerten und den Maskebytes kombiniert. Ein Bitwert gleich 1 wird zurückgegeben, falls der Aktivwert und der in bx stehende Wert des entsprechenden Bits gleich sind. Es ist unabhängig davon, ob sie gleich 0 oder 1 sind. Im anderen Fall wird der Wert gleich 0 an das entsprechende Bit ausgegeben.

Ein Bit der Maskebyte gleich 0 bedeutet, dass das entsprechende Bit der Ausgabeports nicht benutzt ist. An unbenutzten Bits werden generell die von Ihnen definierten Initialisierungswerte zurückgegeben, d. h. falls alle Bits als unbenutzt definiert sind, bekommen Sie in bx die kompletten Initialisierungswerte zurück. Eine Fehlermeldung, dass die Überwachungsports nicht definiert sind, gibt es nicht.

Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 2
Siehe auch: Kapitel 3.2.15; 3.2.17

3.3.56 Funktion 56: Initialisieren der Überwachungsausgabeports

Zweck	Initialisieren der Überwachungsausgabeports	
Aufrufparameter	ax = 56	Funktionsnummer
	bx, cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Für die Überwachungsports können Sie mit dem Programm PARKON entsprechende Initialisierungswerte definieren. Diese Anfangswerte werden beim Treiberstart und beim Aufruf dieser Funktion an den Überwachungsports ausgegeben.

Beachten Sie aber auch, dass die von Ihnen definierten Initialisierungswerte auch nur logische Werte sind. Wenn Sie die Funktion 54 mit den Initialisierungswerten als Aufrufparameter im Register bx benutzen, wird die gleiche Wirkung wie der Aufruf dieser Funktion erzielt.

Um die Überwachungsports auf ihren Anfangswerten zurückzusetzen bietet sich diese Funktion an. Die Reset-Funktion können Sie dafür nicht benutzen, weil sie keinen Einfluss auf den Überwachungsports hat.

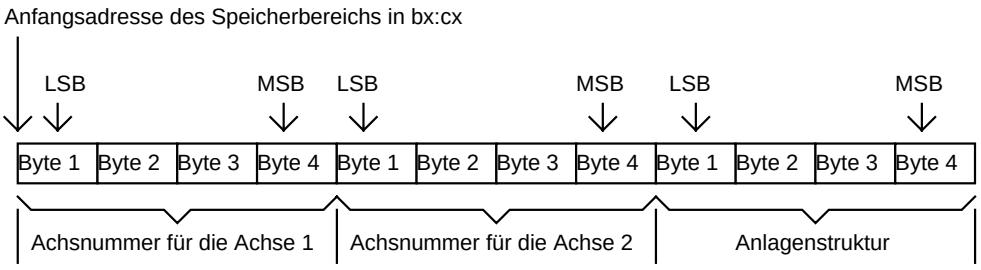
Ausführbarkeit während der Bewegung: Ja
Mögliche Fehlercodes: 0; 2
Siehe auch: Kapitel 3.2.15; 3.2.17; 3.3.2

3.3.57 Funktion 57: Umschalten der Achsen

Zweck	Sofwaremäßiges Umtauschen der Achsen	
Aufrufparameter	ax = 57	Funktionsnummer
	bx	Segmentadresse der Achsumschaltungsparameter
	cx	Offsetadresse der Achsumschaltungsparameter
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Mit unserer Treibersoftware können Sie den Interpolationsbetrieb (Linear-/ Kreisinterpolation) nur innerhalb der 3 Achsen X, Y und Z realisieren. Abgesehen von der Helixinterpolation, wo alle 4 Achsen benötigt werden, wird die A-Achse eigentlich nur synchron gesteuert bzw. mitgeschleppt. Die Zuordnung zwischen der PC-Einsteckkarte und den mechanischen Achsen ist physikalisch festgelegt, d. h. eine Interpolation zwischen den beliebigen mechanischen Achsen ist im Normalfall gar nicht möglich. Um diesen Nachteil zu überwinden, haben wir diese Funktion eingeführt.

Mit dieser Funktion können Sie die Achsbenennung softwaremäßig ändern. Ein hardwaremäßiger Eingriff ist gar nicht notwendig. Auf diese Weise ist es sehr einfach, die Interpolation zwischen den beliebigen Achsen zu realisieren.
Um diese Funktion nutzen zu können, müssen Sie zuerst einen Speicherbereich in Ihrem Anwenderprogramm reservieren. Die Parameter in diesem Speicherbereich teilen dem Treiber mit, welche Achsen umgetauscht werden sollen und welche Anlagestruktur Ihre Anlage nach der Umtauschaktion hat (siehe Bild 42).



LSB = Last Significant Byte (Niederwertigste Byte)
MSB= Most Significant Byte (Höchstwertigste Byte)

Bild 3.42: Parameter für den Achsumtausch

Die Anfangsadresse dieses Speicherbereichs legen Sie beim Funktionsaufruf in bx:cx fest. Aus den ersten 4 Bytes und den zweiten 4 Bytes entnimmt der Treiber, welche Achsen umgetauscht werden sollen. Die Zuordnung zwischen den Achsnummern und den Anlageachsen ist die folgende:

Achsnummer	Anlagenachse
1	X-Achse
2	Y-Achse
3	Z-Achse
4	A-Achse.

Über die letzten 4 Bytes können Sie dem Treiber die nach dem Achsumtausch neu entstehende Anlagestruktur mitteilen. Die Zuordnung zwischen dem Wert in diesen Bytes und der Anlagenstruktur ist:

Wert	Anlagenstruktur
1	X-TTT-Struktur
2	XY-TTT-Struktur
3	XYZ-TTT-Struktur
4	KEIN-TTT-Struktur

Am folgenden Beispiel wollen wir das Gesagte verdeutlichen. Wenn Sie diese Funktion mit den Parameter:

Achsnummer für Achse 1	= 1
Achsnummer für Achse 2	= 4
Anlagenstruktur	= 2

aufrufen, werden die X- und die A-Achse umgetauscht, d. h. ab diesem Zeitpunkt wird die physikalische A-Achse als X-Achse und umgekehrt behandelt. Jeder Zugriff von der Oberfläche auf die physikalische X-Achse wird von dem Treiber direkt auf die physikalische A-Achse umgeleitet und umgekehrt.
Das führt unter anderem dazu, dass der Treiber eine Interpolation mit den Achsen X, Y und Z als eine Interpolation mit den physikalischen Achsen A, Y und Z ausführt. Ein Referenzfahrt der X-Achse wird an der physikalischen A-Achse durchgeführt. Alle Abfragen wie Positionen, Endlageschalterstatus, usw. werden auch entsprechend umgetauscht. Für die interne Verwaltung im Treiber ist die Angabe der Anlagenstruktur unbedingt notwendig. Diese Angabe soll korrekt zu der tatsächlichen Anlagenstruktur sein.
Beachten Sie aber, dass nicht mehr die Zuordnung der physikalischen Achsen X, Y und Z sondern die Zuordnung der physikalischen Achsen A, Y und Z entscheidend für die Anlagenstruktur ist.
Wenn Sie noch einmal diese Funktion mit den Parameter

Achsnummer für Achse 1	= 3
Achsnummer für Achse 2	= 1
Anlagenstruktur	= 3

aufrufen, werden die Achse Z (Achsnummer 3, physikalische Z-Achse) und die Achse X (Achsnummer 1, physikalische A-Achse nach dem ersten

Achsumtausch) umgetauscht. Die neue Anlagenstruktur ist die XYZ-TTT-Struktur. Direkt nach dem Start des Treibers oder nach einem Reset werden die Achsbezeichnungen wieder korrekt zu den physikalischen Achsen zugeordnet. Es ist unabhängig davon, wie oft Sie diese Funktion inzwischen aufgerufen haben.

Ausführbarkeit während der Bewegung: Nein
Mögliche Fehlercodes: 0; 2; 4; 9; 14; 29
Siehe auch: Kapitel 3.2.5; 3.2.15; 3.2.17

3.3.58 Funktion 58: Setzen den Radius für das Bearbeiten auf einer Zylinderoberfläche

Zweck	Definieren des Zylinderradius	
Aufrufparameter	ax = 58	Funktionsnummer
	bx	Segmentadresse der Zylinderparameter
	cx	Offsetadresse der Zylinderparameter
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
In der Praxis kommt die Bearbeitung auf einer Zylinderoberfläche sehr oft vor. Bis zu einer bestimmten Tiefe ist die Bearbeitung auf einer Zylinderoberfläche identisch mit der Bearbeitung auf einer Ebene. Am Bild 3.43 können Sie es am Beispiel eines Zylinders erkennen, deren Drehachse die X-Achse ist.

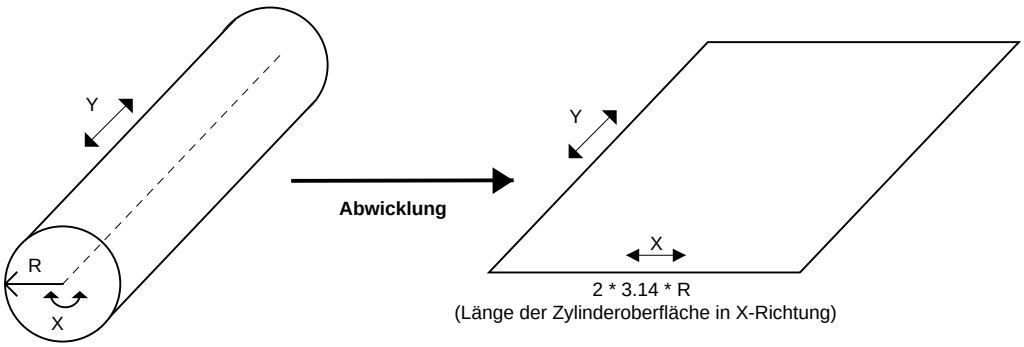
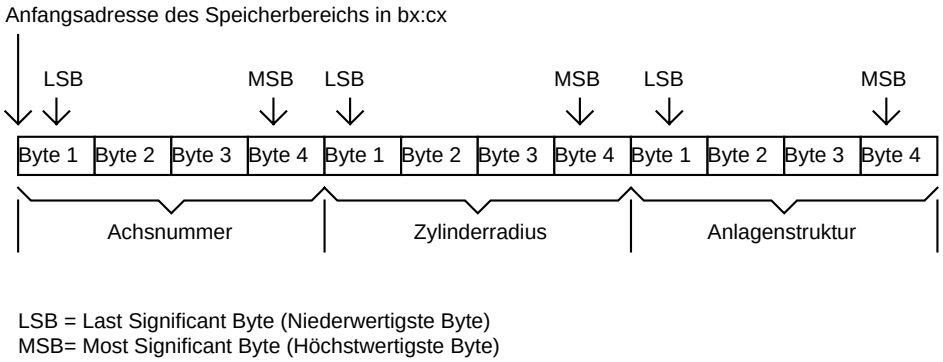


Bild 3.43: Abwicklung einer Zylinderoberfläche

Die Bewegung in der X-Achse auf der Zylinderoberfläche entsteht durch eine Drehbewegung und ist abhängig vom Zylinderradius. In Abhängigkeit vom Radius R muss die Drehgeschwindigkeit auch entsprechend geändert werden, falls Sie eine konstante Bearbeitungsgeschwindigkeit und einen bestimmten Längenmaßstab haben wollen. Der Verwaltungsaufwand für die unterschiedlichen Radien ist nicht klein, wenn Sie alles selbst machen wollen. Aus diesem Grund haben wir diese Funktion eingeführt.

Um diese Funktion nutzen zu können, müssen Sie zuerst einen Speicherbereich in Ihrem Anwenderprogramm reservieren. In diesem Speicherbereich werden die notwendigen Parameter abgelegt (siehe Bild 3.44).



LSB = Last Significant Byte (Niederwertigste Byte)
MSB= Most Significant Byte (Höchstwertigste Byte)

Bild 3.44: Belegung des Speichers mit den Parametern für die Bearbeitung auf einer Zylinderoberfläche

Beim Aufruf dieser Funktion wird die Anfangsadresse dieses Speicherbereichs in bx:cx abgelegt. Aus den ersten 4 Bytes entnimmt der Treiber, für welche Achse Ihrer Anlage der Aufruf gilt. Die Zuordnung zwischen der Achsnummer und der Achse ist die folgende:

Achsnummer	Anlagenachse
1	X-Achse
2	Y-Achse
3	Z-Achse
4	A-Achse

In den nächsten 4 Bytes wird der Zylinderradius definiert. Der Treiber interpretiert den Wert in diesen Bytes als eine ganze Zweier-Komplementär-Zahl mit der Einheit Mikrometer.
Über den letzten 4 Bytes können Sie dem Treiber die neu entstehende Anlagestruktur mitteilen. Es wird noch erläutert, warum Sie eine neue Anlagenstruktur definieren müssen. Die Zuordnung zwischen dem Wert in diesen Bytes und der Anlagenstruktur ist:

Wert	Anlagenstruktur
1	X-TTT-Struktur
2	XY-TTT-Struktur
3	XYZ-TTT-Struktur
4	KEIN-TTT-Struktur

Am Beispiel eines Zylinders mit der Drehachse als X-Achse wollen wir Ihnen diese Funktion weiter erläutern (siehe Bild 3.43). Die Achsnummer muss logischerweise gleich 1 sein. Nachdem der Treiber den Radius kennt, wird der ganze Verwaltungsaufwand von ihm übernommen. Als Anwender brauchen Sie nichts weiter zu tun.

Für Sie ist die Bewegung in X-Richtung eine Linearbewegung. Entsprechend des Radius' ändert der Treiber die Drehgeschwindigkeit so, dass die Bearbeitung auf der Zylinderoberfläche genau wie die Bearbeitung auf einer Ebene ist, sowohl im Hinblick auf die Geschwindigkeit als auch im Hinblick auf den Längenmaßstab. Für eine Bewegung in X-Richtung übergeben Sie dem Treiber eine Bewegungslänge auf der Zylinderoberfläche mit der Einheit Mikrometer. Weil der Treiber den Zylinderradius kennt, wird der Drehwinkel für die X-Achse selbständig berechnet. Sie brauchen sich hier absolut keine Gedanken zu machen.

Falls Sie die Position der X-Achse abfragen, bekommen Sie auch keinen Drehwinkel zurück. Die Bewegungslänge auf der Zylinderoberfläche wird zurückgeliefert. Alle Bewegungsfunktionen, wie die Linear- oder Kreisbewegungsfunktionen, können Sie ohne Beschränkung verwenden. Selbst die Bahnbearbeitung ist auch möglich. Nach einem erfolgreichen Aufruf dieser Funktion ist die Drehachse eine Linearachse für Sie, d. h. wir haben hier eine Art von Achsumwandlung. Deswegen ist die Angabe der neuen Anlagenstruktur notwendig (siehe Bild 3.44).

Um die durch diesen Aufruf entstehende X-Linearachse in die X-Drehachse zurückzuwandeln, können Sie auch diese Funktion benutzen.
Beachten Sie, dass Sie den Zylinderradius gleich Null setzen müssen.
Ein negativer Wert des Zylinderradius' wird nicht akzeptiert. Die Reset-Funktion ist eine zweite Möglichkeit um die Achse zurückzusetzen. Beim Aufruf der Reset-Funktion werden alle umgewandelten Achsen zurückgesetzt.

An diesem Beispiel haben wir die Drehachse als X-Achse gewählt. Aber genau so gut kann die Drehachse die Y- oder die Z- oder die A-Achse sein. Mit der Funktion 57 für das Achsumschalten können Sie jederzeit auch die Achsen umbenennen.
Diese Funktion können Sie auf jede Drehachse und beliebig oft anwenden.
Die Anwendung dieser Funktion auf eine Linearachse ist nicht gestattet.
Beachten Sie außerdem, dass die Bearbeitung nur möglich ist, wenn die Drehachse des Zylinders im Zusammenhang mit den beiden anderen Achsen exakt ein kartesisches Koordinatensystem bildet. Eine andere Zuordnung ist mit dem Treiber momentan nicht machbar.

Ausführbarkeit während der Bewegung: Nein
Mögliche Fehlercodes: 0; 2; 3; 4; 7; 8; 9; 14; 15; 19; 21; 22; 29; 30
Siehe auch: Kapitel 3.2.5; 3.2.15; 3.2.17

3.3.59 Funktion 59: Benutzung der letzten Achse im Spindel-Modus

Aufrufparameter	ax = 59	Funktionsnummer
	bx	Unterfunktion
	cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Diese Funktion ermöglicht Ihnen, die letzte Achse aus dem Achsverbund zu trennen bzw. in den Achsverbund wieder hinzuzufügen. Dadurch haben Sie die Möglichkeit, eine der Achsen als eine separate Achse zu benutzen, um z. B. eine Spindel zu betreiben.
Die Zuordnung zwischen dem Wert im Register bx und den einzelnen Unterfunktionen ist folgende:

bx	Unterfunktion
0	Die getrennte Achse wird wieder hinzugefügt
1	Die letzte Achse wird aus dem Achsverbund getrennt
Sonst	Die letzte Achse wird aus dem Achsverbund getrennt

Die Anzahl der Achsen im System wird im Programm PARKON definiert.
Nach dem Start oder einem Reset ist die letzte Achse wie folgt festgelegt:

Achsanzahl	Achsen	letzte Achse
1	X	X
2	X, Y	Y
3	X, Y, Z	Z
4	X, Y, Z, A	A

Mit Hilfe der Funktion 57 können die Achsen noch umgetauscht werden. Nach dem Achsumtausch sind die physikalischen Achsen und die logischen Achsen nicht mehr identisch. Beachten Sie, dass sich die Achsnummerierung immer auf die logischen Achsen bezieht.

Um dies zu verdeutlichen, wollen wir das Ganze am einem Beispiel erläutern. Die Anlage hat 3 Achsen (X, Y und Z). Nach dem Start bzw. einem Reset ist die Z-Achse die letzte Achse (sowohl physikalisch als auch logisch). Mit dieser Funktion können Sie die physikalische Z-Achse aus dem Achsverbund trennen. Falls die X- und die Z-Achse mit der Funktion 57 umgetauscht sind, ist die physikalische Z-Achse die logische X-Achse und die physikalische X-Achse die logische Z-Achse. Mit dieser Funktion können Sie die logische Z-Achse (d. h. die physikalische X-Achse) aus dem Achsverbund trennen.
Nachdem die letzte Achse getrennt ist, reduziert sich die Achsanzahl automatisch um 1. Ab diesem Zeitpunkt ist die letzte Achse für die meisten Funktionen nicht mehr sichtbar. Über die Funktionen 60 ... 66 können Sie diese Achse wie eine autonome Achse bedienen, die mit den anderen Achsen nichts mehr zu tun hat.

Beachten Sie, dass die Funktion 57 nicht mehr aufgerufen werden darf, solange die Achstrennung aktiv ist - und die Achstrennung ist nur möglich bei Anlagen mit mindestens 2 Achsen. Eine eventuelle Achstrennung können Sie entweder durch einen Aufruf dieser Funktion mit dem Wert 0 im bx oder einen Aufruf der Reset-Funktion aufheben.

Ausführbarkeit während der Bewegung: Nein
Mögliche Fehlercodes: 0; 2; 3; 4; 7; 8; 14; 15; 19; 21; 22; 32
Siehe auch: Kapitel 3.3.60 ... 3.3.66

3.3.60 Funktion 60: Festlegen des Benutzungsortes der Spindelachse

Aufrufparameter	ax = 60	Funktionsnummer
	bx	Unterfunktion
	cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Mit dieser Funktion können Sie festlegen, dass die Spindelachse während der Bahn benutzt wird. Innerhalb einer Bahn können Sie dann einen entsprechenden Befehl benutzen, um die Spindelgeschwindigkeit zu ändern, ohne die Funktion 62 separat aufrufen zu müssen (siehe Anhang B).
Die Funktionen 62, 63 und 64 können nicht mehr separat aufgerufen werden. Falls es einmal festgelegt wurde, dass die Spindelachse nicht in der Bahn benutzt werden soll, können die Funktionen 62, 63 und 64 immer benutzt werden. Es ist unabhängig davon, ob die Bahn aktiv ist oder nicht. Ein in der Bahn stehender Befehl zur Änderung der Spindelgeschwindigkeit wird während der Bahnbearbeitung einfach übersprungen, d. h. es gibt keine Fehlermeldung.
Beachten Sie, dass Sie die Funktion 61 immer benutzen können, um die Spindelgeschwindigkeit zu ändern, egal ob die Spindelachse in der Bahn benutzt werden soll oder nicht. Die Zuordnung zwischen dem Wert im Register bx und den einzelnen Unterfunktionen ist folgende:

bx	Unterfunktion
0	Die Spindel wird während der Bahn nicht benutzt
1	Die Spindel wird während der Bahn benutzt
Sonst	Die Spindel wird während der Bahn benutzt

Nach dem Start des Treibers oder nach einem Aufruf der Reset-Funktion lässt sich die Spindelachse nicht in der Bahn benutzen. Ein Aufruf dieser Funktion mit dem Wert 1 im Register bx ist notwendig, falls Sie die Spindelachse während des Bahnfahrens benutzen wollen.

Ausführbarkeit während der Bewegung: Nein
Mögliche Fehlercodes: 0; 2; 3; 4; 7; 8; 14; 15; 19; 21; 22; 31; 32; 34
Siehe auch: Kapitel 3.3.59; 3.3.61 ... 3.3.66; Anhang B

3.3.61 Funktion 61: Setzen des Änderungsfaktors der Spindelgeschwindigkeit

Aufrufparameter	ax = 61	Funktionsnummer
	bx	Änderungsfaktor der Spindelgeschwindigkeit
	cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Die Geschwindigkeit einer Spindelachse ist das Produkt aus der mit Hilfe der Funktion 62 gesetzten Soll-Geschwindigkeit und dem Änderungsfaktor, der mit dieser Funktion jederzeit neu festgelegt werden kann. Sie haben also mit diesem Änderungsfaktor die Möglichkeit, die Spindelgeschwindigkeit zu ändern, ohne die Funktion 62 benutzen zu müssen. Falls Sie mit der Funktion 60 festgelegt haben, dass die Spindelachse nur in der Bahn benutzt werden darf, ist diese Funktion die einzige Möglichkeit, um die Spindelgeschwindigkeit zu ändern.
Die Art und Weise, wie dieser Faktor intern im Treiber gehandhabt wird und wie Sie diese Funktion benutzen können, ist ähnlich wie beim Änderungsfaktor der Werkzeuggeschwindigkeit (siehe Funktion 21).

Ausführbarkeit während der Bewegung: Ja

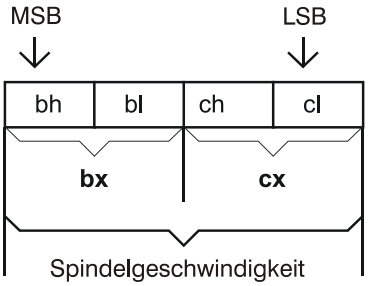
Mögliche Fehlercodes: 0; 2; 3; 4; 7; 8; 31
Siehe auch: Kapitel 3.3.21; 3.3.62

3.3.62 Funktion 62: Setzen einer neuen Spindelgeschwindigkeit

Aufrufparameter	ax = 62	Funktionsnummer
	bx:cx	Spindelgeschwindigkeit
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Mit dieser Funktion haben Sie die Möglichkeit, die Spindelachse in den Zustand der endlosen Bewegung zu setzen, falls die Geschwindigkeit der Spindelachse ungleich 0 ist. Nach dem Treiberstart bzw. nach einem Aufruf der Reset-Funktion ist die Spindelgeschwindigkeit immer gleich 0. Die tatsächliche Soll-Geschwindigkeit der Spindelachse ist gleich dem Produkt der hier gesetzten Geschwindigkeit und dem Änderungsfaktor, der sich mit der Funktion 61 ändern lässt.
Beachten Sie, dass die Spindelachse entweder im Geschwindigkeits-Modus oder im Positions-Modus arbeiten kann. Der Positions-Modus ist mit der

Funktion 63 zu erreichen. Im Positions-Modus bewegt sich die Spindelachse unabhängig von allen anderen Achsen bis zu der gewünschten Zielposition. Solange diese Zielposition noch nicht erreicht ist, ist der Aufruf dieser Funktion nicht erlaubt, d. h. das Umschalten in den Geschwindigkeits-Modus ist nicht gestattet. Diese Funktion dürfen Sie auch nicht benutzen, falls die Spindelachse mit der Funktion 60 für die Benutzung in der Bahn festgelegt wurde oder falls sich die Spindelachse im Hand-Modus befindet.
Falls die Spindelachse für die Benutzung in der Bahn bestimmt wurde, kann die Spindelgeschwindigkeit entweder durch einen Aufruf der Funktion 61 oder durch einen entsprechenden Befehl in der Bahndatei verändert werden (siehe Anhang B).
Die neue Spindelgeschwindigkeit, die Sie dem Treiber übergeben wollen, steht in bx:cx (siehe Bild 3.45).



LSB = Last Significant Byte (Niederwertigste Byte)
MSB = Most Significant Byte (Höchstwertigste Byte)

Bild 3.45: Die neue Spindelgeschwindigkeit in bx:cx

Ein erfolgreicher Aufruf dieser Funktion führt sofort zu einer Änderung der Spindelgeschwindigkeit. Der Treiber interpretiert die Zahl in bx:cx als eine ganze Zweier-Komplementär-Zahl. In Abhängigkeit davon, ob die Spindelachse als eine Linearachse oder eine Drehachse definiert wurde, ist die Einheit der Bahngeschwindigkeit entweder Mikrometer/Sekunde [μm/s] oder Bogensekunde/Sekunde ["/s].
Im Unterschied zu der Funktion 20, mit der Sie die Segmentgeschwindigkeit neu definieren können, kann die Spindelgeschwindigkeit hier einen negativen oder einen positiven Wert haben. Entsprechend bewegt sich die Spindelachse in die negative oder in die positive Richtung.
Falls die Spindelachse im Geschwindigkeits-Modus und in Bewegung ist, können Sie nur durch einen Aufruf der Reset-Funktion oder durch einen Aufruf dieser Funktion mit der Spindelgeschwindigkeit gleich 0 sofort anhalten.

Im Bezug auf die Referenzfahrt beachten Sie folgendes. Wenn Sie einmal die letzte Achse aus dem Achsverbund trennen, ändert sich der Status des Referenzflags nicht, solange sich die Spindelachse noch nicht im Geschwindigkeits-Modus mit einer Geschwindigkeit ungleich 0 bewegt.
Wenn Sie einmal diese Funktion mit einer Geschwindigkeit ungleich 0 aufrufen, wird der Referenzflag dieser Spindelachse zurückgesetzt, d. h. wenn Sie

irgendwann die Achstrennung aufheben, müssen Sie die Referenzfahrt für diese Achse neu ausführen. Wenn sich die Spindelachse nur im Positions-Modus bewegt, ändert sich der Status des Referenzflags dieser Achse nicht.

Ausführbarkeit während der Bewegung: Ja, der neue Wert wird sofort akzeptiert.

Mögliche Fehlercodes: 0; 2; 3; 4; 7; 8; 31; 32; 33; 34;
Siehe auch: Kapitel 3.3.21; 3.3.61; Anhang B

3.3.63 Funktion 63: Positionieren der Spindelachse

Aufrufparameter	ax = 63	Funktionsnummer
	bx	Segmentadresse der Bewegungsparameter
	cx	Offsetadresse der Bewegungsparameter
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Mit dieser Funktion können Sie die Spindelachse zu einer neuen Zielposition bewegen. Falls die Referenzfahrt der Spindelachse noch nicht ausgeführt ist, wird die aktuelle Position nach dem Treiberstart als Nullpunkt angenommen, d. h. Sie können die Spindelachse immer bewegen, ohne die Referenzfahrt ausführen zu müssen. Falls sich die Spindelachse im Geschwindigkeits-Modus bewegt oder im Hand-Modus befindet oder für die Benutzung in der Bahn bestimmt ist, ist ein Aufruf dieser Funktion 63 nicht erlaubt. Falls sich die Spindelachse im Positions-Modus bewegt, ist ein Aufruf der Funktion 62 nicht erlaubt.
Beachten Sie, dass die absolute Position der Spindelachse unbekannt ist, falls sich die Spindelachse einmal im Geschwindigkeits-Modus befindet. Sie können die absolute Position aber jederzeit mit der Funktion 66 abfragen. Weil das interne Positionsregister des Chips LM628 32 bit beträgt, kann es im Geschwindigkeits-Modus sehr oft dazu führen, dass das Positionsregister überlaufen ist. D. h. der mit der Funktion 66 zurückgelieferte Positionswert muss nicht immer gleich dem im Geschwindigkeits-Modus zurückgelegten Weg sein.
Um diese Funktion zu benutzen, müssen Sie einen Speicherbereich in Ihrem Programm reservieren. Die Belegung des Speicherbereichs ist in Bild 3.46 zu sehen. In den ersten 4 Bytes steht die Bewegungsgeschwindigkeit, mit der die Spindelachse sich zur Soll-Position bewegen soll. Die Soll-Position steht in den letzten 4 Bytes. Beim Aufruf dieser Funktion müssen Sie die Anfangsadresse des Speicherbereichs in bx:cx einlegen.

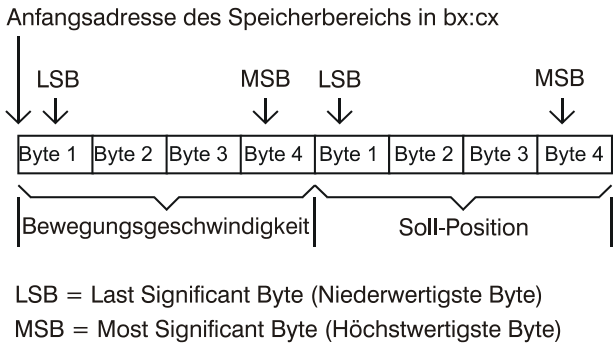


Bild 3.46: Die Bewegnungsparameter für das Positionieren der Spindelachse

Der Treiber interpretiert die übergegebenen Werte als eine ganze Zweier-Komplementär-Zahl. In Abhängigkeit davon, ob die Spindelachse als eine Linearachse oder eine Drehachse definiert wurde, ist die Einheit der Position entweder Mikrometer/Sekunde [μm/s] oder Bogensekunde/Sekunde ["/s]. Achten Sie bitte, dass die Bewegungsgeschwindigkeit positiv sein muss. Sonst bekommen Sie den Fehlercode 17 zurück.

Ausführbarkeit während der Bewegung: Ja

Mögliche Fehlercodes: 0; 2; 3; 4; 7; 8; 31; 32; 33; 34;
Siehe auch: Kapitel 3.3.60; 3.3.62; 3.3.64; 3.3.65; 3.3.66

3.3.64 Funktion 64: Ein- oder Ausschalten des Hand-Modus der Spindelachse

Aufrufparameter	ax = 64	Funktionsnummer
	bx	Unterfunktion
	cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
In der Praxis ist es oft notwendig, dass Sie die Spindelachse mit der Hand bewegen können. Mit Hilfe dieser Funktion kann der Hand-Modus der Spindelachse ein- oder ausgeschaltet werden. Falls der Hand-Modus aktiv ist, ist der Positionsregler der Spindelachse ausgeschaltet. Zur Leistungsendstufe ist 0 Volt ausgegeben, obwohl die Position der Spindelachse weiterhin überwacht wird. Die Spindelachse lässt sich aber nur mit der Hand bewegen, falls Sie gleichzeitig die Leistungsendstufe für die Spindelachse über einen Ausgabenport in den Disable-Zustand setzen. Das Setzen der Leistungsendstufe in den Disable-Zustand müssen Sie selber erledigen. Solange der Hand-Modus aktiv ist, können Sie die Funktionen 62 und 63 nicht benutzen.

Die Zuordnung zwischen dem Wert im Register bx und den einzelnen Unterfunktionen ist folgende:

bx	Unterfunktion
0	Ausschalten des Hand-Modus der Spindelachse
1	Einschalten des Hand-Modus der Spindelachse
Sonst	Einschalten des Hand-Modus der Spindelachse

Verwechseln Sie diese Funktion nicht mit der Funktion 8.
Mit Hilfe dieser Funktion können Sie nur den Hand-Modus der Spindelachse ein- oder ausschalten und es gilt umgekehrt, dass der Hand-Modus der Spindelachse mit der Funktion 8 nicht ändern lässt.

Ausführbarkeit während der Bewegung: Ja

Mögliche Fehlercodes: 31; 32; 33
Siehe auch: Kapitel 3.3.8; 3.3.62; 3.3.63; 3.3.65

3.3.65 Funktion 65: Abfragen des Status der Spindelachse

Aufrufparameter	ax = 65	Funktionsnummer
	bx, cx, dx	undefiniert
Ergebnis	al	Fehlercode
	ah	Status der Spindelachse
	bx, cx, dx	undefiniert

Kommentar:
Mit dieser Funktion können Sie den Status der Spindelachse abfragen.
Das Ergebnis der Abfrage befindet sich im Register ah. Die Nummerierung der Bits vom Register ah ist im Bild 3.19 zu sehen. Die Bedeutung der einzelnen Bits wird im Folgenden beschrieben.

Register ah:

Bit 0: Benutzungs-Status des Spindel-Modus
- = 0 Die letzte Achse befindet sich nicht im Spindel-Modus
- = 1 Die letzte Achse befindet sich im Spindel-Modus

Bit 1: Benutzungs-Ort der Spindelachse
- = 0 Die Spindelachse ist nicht für die Benutzung in der Bahn bestimmt
- = 1 Die Spindelachse ist für die Benutzung in der Bahn bestimmt

Bit 2,3: Submodi der Spindelachse
Bit 2 = 0 Die Spindelachse steht im Geschwindigkeits-Modus mit
Bit 3 = 0 einer Geschwindigkeit gleich 0, d. h. die Spindelachse bewegt sich nicht. Das Bit 4 ist gleich 0.

Bit 2 = 1 Die Spindelachse steht im Geschwindigkeits-Modus mit
Bit 3 = 0 einer Geschwindigkeit ungleich 0, d. h. die Spindelachse ist momentan in einer endlosen Bewegung. Das Bit 4 ist gleich 1.
Bit 2 = 0 Die Spindelachse steht im Positions-Modus, d. h. die
Bit 3 = 1 Spindelachse wurde zuletzt durch die Funktion 63 beauftragt, sich zu einer absoluten Zielposition zu bewegen. Das Bit 4 zeigt an, ob die Bewegung momentan aktiv (Bit 4 = 1) ist oder nicht (Bit 4 = 0). Eine nicht aktive Bewegung bedeutet, dass die Zielposition schon erreicht ist.

Bit 4: Bewegungs-Status der Spindelachse
= 0 Die Spindelachse ist nicht in Bewegung
= 1 Die Spindelachse ist in Bewegung

Bit 5: Hand-Modus-Status
= 0 Der Hand-Modus der Spindelachse ist nicht aktiv
= 1 Der Hand-Modus der Spindelachse ist aktiv

Bit 6: Spindel-Dauer-Modus-Status
= 0 Der Treiber wurde nicht mit dem Optionsschalter /DSM gestartet
= 1 Der Treiber wurde mit dem Optionsschalter /DSM gestartet.

Bit 7: Reservierte Bits

Ausführbarkeit während der Bewegung: Ja

Mögliche Fehlercodes: 0; 2
Siehe auch: Kapitel 3.3.5; 3.3.59 ... 3.3.64

3.3.66 Funktion 66: Abfragen der Bewegungsparameter der Spindelachse

Aufrufparameter	ax = 66	Funktionsnummer
	bx	Segmentadresse der Bewegungsparameter
	cx	Offsetadresse der Bewegungsparameter
	dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Diese Funktion ermöglicht es Ihnen, die Bewegungsparameter der Spindelachse abzufragen. Es sind die Ist-Geschwindigkeit, die Ist-Position sowie die Soll-Position der Spindelachse. Die Soll-Position ist die Position, die der Chip LM628 aufgrund der Rampengenerierung berechnet und ausgibt. Die Ist-Position ist die aktuelle Position, die logischerweise der Soll-Position hinterher läuft.

Um diese Funktion nutzen zu können, müssen Sie zuerst einen Speicherbereich in Ihrem Anwenderprogramm reservieren. Die Anfangsadresse dieses Speicherbereichs legen Sie beim Aufruf dieser Funktion in bx:cx fest. Mit Hilfe dieser Anfangsadresse schreibt der Treiber die Bewegungsparameter der Spindelachsen in den von Ihnen reservierten Speicherbereich (siehe Bild 3.47).
Für jeden Parameter braucht der Treiber 4 Byte Speicher. Den Wert in diesen 4 Bytes müssen Sie als eine ganze Zweier-Komplementär-Zahl interpretieren.

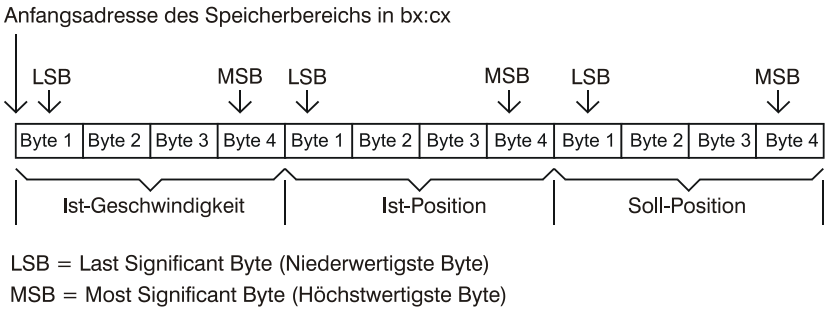


Bild 3.47: Belegung des Speichers mit Bewegungsparametern der Spindelachse

Die Positionen, die Sie zurück bekommen, sind Absolut-Positionen mit dem Referenzpunkt als Bezugpunkt. In Abhängigkeit davon, ob die Spindelachse eine Linearachse oder eine Drehachse ist, hat die Ist-Geschwindigkeit die Einheit Mikrometer/Sekunde [μm/s] oder Bogensekunde/Sekunde ["/s] und die Position die Einheit Mikrometer [μm] oder Bogensekunde ["].

Ausführbarkeit während der Bewegung: Ja

Mögliche Fehlercodes: 0; 2; 31
Siehe auch: 3.3.61; 3.3.62; 3.3.63

3.3.67 Funktion 67: Ein- oder Ausschalten des Handrad-Modus

Aufrufparameter	ax = 67	Funktionsnummer
	bx	Unterfunktionen
	cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Um die Achsen manuell zu bewegen, wird das Handrad in der Praxis sehr häufig benutzt. Deswegen haben wir diese Funktion eingeführt, um das Integrieren des Handrads in die Steuerung zu erleichtern.
Von den Bewegungsfunktionen 24 ... 31 und 33 können Sie nur noch die Relativbewegungsfunktionen 24 oder 26 benutzen, falls der Handrad-Modus aktiv ist. Im Bewegungsverhalten gibt es keinen Unterschied, welche Funktion Sie benutzen. In diesem Modus werden die Achsen nicht interpoliert sondern

unabhängig voneinander bewegt. Jede Achse versucht autonom, ihre vorgegebene Zielposition so schnell wie möglich zu erreichen.

Das Integrieren des Handrads in die Steuerung geschieht folgendermaßen:
Das Handrad mit der zugehörigen Elektronik liefert beim Drehen ständig Impulse, die die Bewegung nachbilden. Sie müssen diese Impulse z. B. mit Hilfe einer Zählkarte zählen. In regelmäßigen Zeitabständen müssen Sie die Anzahl der Impulse abfragen, in äquivalente Bewegungslänge umwandeln und zum Treiber schicken. Die Achsen versuchen, der regelmäßig geschickten Positionen nachzukommen. Dadurch entsteht eine Nachbildung der Handradbewegung auf den Achsen.
Beachten Sie, dass die mit der Funktion 24 bzw. 26 zum Treiber geschickten Positionen relative Positionen sind und dass Sie gleichzeitig mehr als eine Achse im Handrad-Modus bewegen können. Außerdem müssen Sie beachten, dass Sie ständig neue Relativpositionen schicken können, ohne warten zu müssen, bis die vorherige Bewegung schon fertig ist.
Das ist der große Unterschied zu einem normalen Aufruf der Funktion 24 oder 26, bei dem der Handrad-Modus nicht aktiv ist. Hier müssen Sie warten, bis die vorherige Bewegung zu Ende ist.
Die Zuordnung zwischen dem Wert im Register bx und den einzelnen Unterfunktionen ist folgende:

bx	Unterfunktion
0	Ausschalten des Handrad-Modus
1	Einschalten des Handrad-Modus
Sonst	Einschalten des Handrad-Modus

Nach dem Treiberstart oder nach einem Aufruf der Reset-Funktion ist der Handrad-Modus nicht aktiv. Der Handrad-Modus und der Hand-Modus dürfen nicht gleichzeitig aktiv sein. Das Ausschalten des Handrad-Modus ist nicht gestattet, falls eine Bewegung noch aktiv ist, d. h. Sie müssen warten, bis die Bewegung zu Ende ist oder die Funktion 68 benutzen, um die Bewegung zu stoppen.

Ausführbarkeit während der Bewegung: Nein

Mögliche Fehlercodes: 0; 2; 4; 10; 36
Siehe auch: Kapitel 3.3.8; 3.3.24; 3.3.26; 3.3.68; 3.3.69

3.3.68 Funktion 68: Stop einer Handrad-Bewegung

Aufrufparameter	ax = 68	Funktionsnummer
	bx, cx, dx	undefiniert
Ergebnis	al	Fehlercode
	bx, cx, dx	undefiniert

Kommentar:
Mit Hilfe dieser Funktion können Sie eine mit der Funktion 24 oder 26 gestartete Bewegung im Handrad-Modus sofort unterbrechen. Diese Funktion können Sie nicht aufrufen, falls der Handrad-Modus nicht aktiv ist.

Ausführbarkeit während der Bewegung: Ja, aber nur bei aktivem Handrad-Modus

Mögliche Fehlercodes: 0; 2; 35
Siehe auch: Kapitel 3.3.24; 3.3.26; 3.3.67; 3.3.69

3.3.69 Funktion 69: Abfragen der Parameter des Handrad-Modus

Aufrufparameter	ax = 69	Funktionsnummer
	bx, cx, dx	undefiniert
Ergebnis	al	Fehlercode
	ah	Parameter des Handrad-Modus
	bx, cx, dx	undefiniert

Kommentar:
Mit dieser Funktion können Sie den Status des Handrad-Modus abfragen. Das Ergebnis der Abfrage befindet sich im Register ah. Die Nummerierung der Bits vom Register ah ist im Bild 3.19 zu sehen. Die Bedeutung der einzelnen Bits wird im Folgenden beschrieben.

Register ah:

Bit 0:	Status des Handrad-Modus
= 0	Der Handrad-Modus ist nicht aktiv
= 1	Der Handrad-Modus ist aktiv

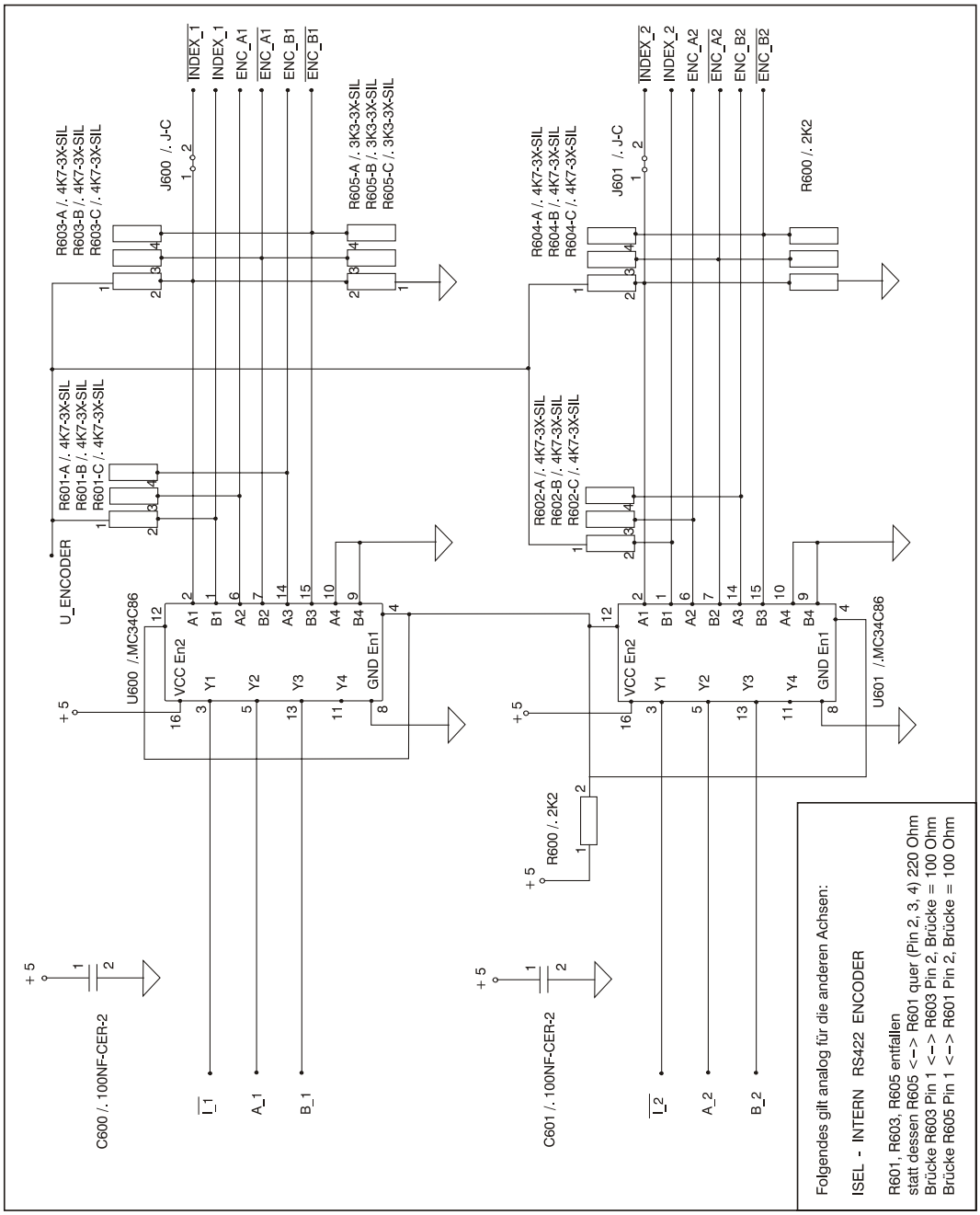
Bit 1:	Status der Handrad-Bewegung
= 0	Keine Bewegung ist aktiv im Handrad-Modus
= 1	Eine Bewegung ist aktiv im Handrad-Modus

Bit 2 ... 7: Reservierte Bits

Ausführbarkeit während der Bewegung: Ja

Mögliche Fehlercodes: 0; 2
Siehe auch: Kapitel 3.3.67; 3.3.68

Anhang A: Encoderbeschaltung der *isel*-Servomotor-Steuerkarte UPMV 4/12



Anhang B: Übersicht des Formats zur Ansteuerung von *isel*-Maschinen

NCP-Dateikennung **IMF_PBLxxxxxxxxxx**
Angabe einer Kennung für eine gültige NCP-Quelldatei.
Die Angabe dieser Kennung ist nur erforderlich, falls mit einer Servo-Steuerung Bahnbewegungen realisiert werden sollen. Die Kennung steht am Anfang der ersten Zeile der NCP-Quelldatei.

Groß-/ Kleinschreibung
beliebig
keine Unterscheidung zwischen Groß- und Kleinschreibung.

Achsbezeichner **X Y Z A B C**
Zulässige Achsbezeichner sind X, Y und Z sowie A für die 4. Achse, B für die 5. Achse usw.

Kommentar **;**
Einleitung eines Kommentars. Der Kommentar gilt bis zum Ende der aktuellen Zeile.

Trennzeichen
Zwischen den Kommandos und den einzelnen Parametern sind sogenannte „white spaces“ als Trennzeichen zulässig (das sind Formatierungszeichen wie Leerzeichen, TABs usw.).

Zeilennummer **Nxxxxxx**
Angabe einer Zeilennummer des aktuellen NC-Satzes.
Für die spätere Verwendung des Bahndatengenerators einer Servo-Steuerung ist die Angabe von Zeilennummern in der NCP-Quelldatei notwendig. Der Bahndatengenerator einer Microstep-Steuerung kann Dateien mit und ohne Zeilennummern bearbeiten.

Referenzfahrt **REF X (Y Z A ...)**
Ausführen einer Referenzfahrt der angegebenen Achsen.
Die Reihenfolge der Angabe ist beliebig. Die Referenzfahrt wird in der Reihenfolge Z Y X A B ... ausgeführt.
Um die Reihenfolge zu ändern, fügen Sie ein Whitespace-Trennzeichen zwischen Achsbezeichnern ein.
Beispiel

```
REF X        ;Ausführen der Referenzfahrt X
REF XYZ      ;Referenzfahrt in der Reihenfolge Z Y X
REF X Y Z    ;Referenzfahrt in der Reihenfolge X Y Z
REF X YZA    ;Zuerst Referenzfahrt X, dann in der
             ;Reihenfolge Z Y A
```

SW-Endschalter*)

LIMIT Xneg,pos Yneg,pos Zneg,pos Aneg,pos

Definieren der Software-Endschalter für den nachfolgenden Verfahrbetrieb. Nach dem Achsbezeichner folgt der negative sowie der positive Wert für den Software-Endschalter.
Die Angabe der Positionen erfolgt in Mikrometer [µm].

Normal-geschwindigkeit

VEL xxx

Setzen der Normalgeschwindigkeit auf den Wert xxx.
Die Normalgeschwindigkeit ist die Segment-Verfahrgeschwindigkeit, die bei Verwendung eines der Befehle MOVEABS, MOVEREL, FASTABS, FASTREL, CWABS, CWREL, CCWABS, CCWREL, CWHLXABS, CWHLXREL, CCWHLXABS oder CCWHLXREL benutzt wird. Die Einheit des übergebenen Geschwindigkeitswertes ist [μm/s].
Beispiel

```
VEL 5000          ;setzt die Normalgeschwindigkeit auf 5 mm/s
```

Eilgeschwindigkeit

FASTVEL xxx

Setzen der Eilgeschwindigkeit auf den Wert xxx.
Die Eilgeschwindigkeit ist die Segment-Verfahrgeschwindigkeit, die bei Verwendung eines der Befehle FASTABS oder FASTREL benutzt wird. Die Einheit des übergebenen Geschwindigkeitswertes ist [μm/s].
Beispiel

```
FASTVEL 50000     ;setzt die Eilgeschwindigkeit auf 50 mm/s
```

Absolutbewegung

MOVEABS X... Y... Z... A...

Absolute Linearbewegung mit Normalgeschwindigkeit (Geschwindigkeit, die durch Verwendung des Befehls VEL gesetzt wird).
Diese Funktion führt eine Bewegung von bis zu vier Achsen aus. Bei einer Konfiguration der Achsen als XYZ_TTT-Struktur werden die drei Achsen X, Y und Z interpoliert; die Achse A wird synchron nachgeführt. Durch den Achsbezeichner (X, Y, ...) wird die Zielkoordinate der entsprechenden Achse zugewiesen. Die Zielposition der Bewegung wird in Absolutkoordinaten angegeben (bezogen auf den aktuell gesetzten Werkstücknullpunkt).
Die Angabe der Zielposition für jede Achse ist modal, d. h. wenn für eine oder mehrere Achsen kein Wert übergeben wird, wird die aktuelle Position beibehalten.
Die Einheit der Zielposition ist Mikrometer [μm] für Linearachsen und Bogensekunden [''] für Drehachsen (1° = 60‘ = 3600'').
Beispiel

```
VEL 10000                       ;Geschw. 10 mm/s
MOVEABS X20000 Y15000 Z-5000    ;Achsen XYZ an neue Position
MOVEABS Y15000                  ;nur Achse Y an neue Position
                                ;andere Achsen bleiben stehen
```

Abs. Eilbewegung **FASTABS X... Y... Z... A...**

Bewegung zu einer Absolutposition in Eilgeschwindigkeit (das ist die Geschwindigkeit, die durch Verwendung des Befehls FASTVEL gesetzt wird).
Die Bedeutung der nachstehenden Parameter ist die gleiche wie bei der Absolutbewegung mit Normalgeschwindigkeit.

Relativbewegung **MOVEREL X... Y... Z... A...**

Relative Linearbewegung mit Normalgeschwindigkeit (das ist die Geschwindigkeit, die durch Verwendung des Befehls VEL gesetzt wird).
Diese Funktion führt eine Bewegung von bis zu vier Achsen aus. Bei einer Konfiguration der Achsen als XYZ_TTT-Struktur werden die drei Achsen X, Y und Z interpoliert; die Achse A wird synchron nachgeführt. Durch den Achsbezeichner (X, Y, ...) wird die Zielkoordinate der entsprechenden Achse zugewiesen. Die Zielposition der Bewegung wird in Absolutkoordinaten angegeben (bezogen auf die aktuelle Position der Achsen).
Die Angabe der Zielposition für jede Achse ist modal, d. h. wenn für eine oder mehrere Achsen kein Wert übergeben wird, wird für diese Achse(n) automatisch der Verfahrweg 0 angenommen.
Die Einheit der Zielposition ist Mikrometer [µm] für Linearachsen und Bogensekunden ['’] für Drehachsen (1° = 60‘ = 3600’’).
Beispiel

```
VEL 5000                          ;Geschwindigkeit 5 mm/s
MOVEREL X20000 Y15000 Z-5000      ;Bewegung von 3 Achsen
MOVEREL Z-5000                    ;nur Achse Z 5 mm abwärts
```

Rel. Eilbewegung **FASTREL X... Y... Z... A...**

Relative Bewegung in Eilgeschwindigkeit (das ist die Geschwindigkeit, die durch Verwendung des Befehls FASTVEL gesetzt wird)
Die Bedeutung der nachstehenden Parameter ist die gleiche wie bei der Relativen Normalbewegung.
Beispiel

```
VEL 5000                          ;Normalgeschwind. 5 mm/s
FASTVEL 35000                     ;Eilgeschwindigkeit 35 mm/s
MOVEREL Z5000          ;Achse Z 5 mm anheben
FASTREL X100000 Y100000           ;Positionierung in XY
```

Kreisebene

PLANE XY (XZ, YZ)

Festlegen der Bearbeitungsebene für die Kreisinterpolation bei Verwendung eines der Befehle CWABS, CCWABS, CWREL oder CCWREL.

Dieser Befehl gilt nur für die Bearbeitungsebene des Kreises, er kann nicht zur Festlegung einer Arbeitsebene der Linearbewegungen verwendet werden. Das Verhalten des Befehls ist modal, d. h. wenn die Kreisebene einmal gesetzt ist, bleibt diese Einstellung bis zum nächsten Aufruf des Befehls PLANE erhalten.

Beispiel

```
PLANE XY                    ;für die Kreisinterpolation werden
                            ;die Achsen X und Y benutzt
CWREL I20000 J0 X0 Y0 Z0 A0 ;Kreisinterpolation ausführen
PLANE YZ                    ;die Achsen Y/Z werden benutzt
CWREL I20000 J0 X0 Y0 Z0 A0 ;mit dem gleichen Kommando
                            ;führen Y und Z den Kreis aus
```

*K*reis Absolut

CWABS/CCWABS I... J... X... Y.... Z.... A....

Ausführen einer Kreisinterpolation Kreisinterpolation in einer der zulässigen Ebenen XY, YZ oder XZ unter Verwendung von Absolutkoordinaten.

Die Angabe der Koordinaten erfolgt absolut, d. h. alle Positionsangaben beziehen sich auf den gesetzten Werkstücknullpunkt.

Die Festlegung der Kreisrichtung erfolgt mit dem Kommando CWABS (im Uhrzeigersinn) oder CCWABS (Gegen-Uhrzeigersinn).

Die Parameter I und J geben die Koordinaten des Kreismittelpunktes an; I und J sind Abhängigkeit von der jeweils aktiven Kreisebene.

Bearbeitungsebene	Bedeutung
XY-Ebene	I entspricht einer X-, J einer Y-Koordinate
XZ-Ebene	I entspricht einer X-, J einer Z-Koordinate
YZ-Ebene	I entspricht einer Y-, J einer Z-Koordinate

Die Parameter X, Y, Z, und A geben die jeweiligen absoluten Endpositionen an.

Falls eine Achse ihre aktuelle Position behalten soll, kann die Angabe der Endposition für diese Achse entfallen. Falls Angaben für Achsen gemacht werden, die für die Kreisinterpolation nicht benötigt werden, werden diese ignoriert.

Die Angabe der Positionen erfolgt in Mikrometer. Als Fahrgeschwindigkeit wird die Normalgeschwindigkeit verwendet (das ist die Geschwindigkeit, die durch Verwendung des Befehls VEL gesetzt wird).

Beispiel

```
PLANE XY                              ;Festlegen der Interpolationsebene
MOVEABS X50000 Y50000 Z-2500          ;Startpunkt anfahren
CWABS I75000 J50000 X50000 Y50000 Z-2500  ;Kreis r = 25 mm
MOVEABS X50000 Y50000 Z-2500          ;dieser Befehl erzeugt
CWABS I75000                          ;die gleiche Bewegung,
                                      ;aber nur die absolut
                                      ;notwendigen Parameter
                                      ;werden übergeben
```

Kreis Relativ

CWREL/CCWREL I... J... X... Y.... Z.... A....

Dieser Befehl führt eine Kreisinterpolation unter Verwendung von Relativkoordinaten aus.
Die Verwendung der Kreisparameter erfolgt sinngemäß wie bei der Ausführung mit der Angabe von Absolutkoordinaten mit dem Unterschied, dass die Parameter relative Positionen beschreiben. Die Angabe der Parameter erfolgt modal, d. h. wenn eine relative Endposition 0 ist, kann diese Angabe entfallen. Falls Parameter X, Y, Z, A, B und C auftauchen, die nicht benötigt werden, werden diese ignoriert.

Beispiel

```
PLANE XY                ;Festlegen der Interpolationsebene
MOVEABS X50000 Y50000 Z-2500    ;Startpunkt anfahren
CWREL I20000 J0 X0 Y0 Z0 A0     ;Vollkreis mit Radius 20 mm
MOVEABS X50000 Y50000 Z-2500    ;derselbe Kreis, aber ohne
CWREL I20000            ;die nicht erforderlichen Parameter
```

Zeitverzögerung*)

WAIT xxx

Ausführen einer Zeitverzögerung während der Bearbeitung.
Die Angabe der Zeit erfolgt in Millisekunden [ms].
Während der Ausführung des Befehls kann die Zeitverzögerung je nach Art des verwendeten Steuerungsprogramms übersprungen bzw. abgebrochen werden.

Beispiel `WAIT 2000 ;2 Sekunden warten`

WS-Null Setzen

WPZERO

Die Position, an dem das Werkzeug im Augenblick steht, wird als neuer Werkstücknullpunkt gesetzt. Der alte Werkstücknullpunkt wird gelöscht. Es werden keine Parameter benötigt.

Beispiel

```
REF XYZ                 ;Referenzfahrt
MOVEABS X30000 Y25000   ;An eine Absolutposition fahren
WPZERO                  ;und diese Position als neuen
                        ;Werkstücknullpunkt setzen
```

WS-Null Absolut **WPZEROABS X.... Y.... Z.... A....**

Die übergebenen Koordinaten werden als neuer Werkstücknullpunkt gesetzt. Die Koordinaten sind Absolutwerte und beziehen sich auf den Maschinennullpunkt (das ist die Position, die normalerweise durch eine Referenzfahrt ermittelt wird). Der alte Werkstücknullpunkt wird gelöscht. Die Angabe der Positionen erfolgt in Mikrometer [µm]. Falls die Position des Werkstücknullpunktes für eine oder mehrere Achsen unverändert bleiben soll, genügt es, die entsprechenden Parameter auszulassen. Für diese Achse wird die bisher benutzte Koordinate beibehalten.

Beispiel

```
REF XYZ                          ;Referenzfahrt von XYZ
WPZEROABS X5000 Y5000 Z-5000 ;Werkstücknullpunkt an (X5000,
                                 ;Y5000, Z-5000) setzen
WPZEROABS X35000 Y20000          ;jetzt an Position (X35000,
                                 ;Y20000, Z-5000) setzen
```

Clear WS-Nullpkt **WPCLEAR**

Löschen des aktuell gesetzten Werkstücknullpunktes. Der Maschinennullpunkt (Position nach Referenzfahrt) wird als neuer Werkstücknullpunkt angenommen. Diese Funktion erfordert keine Parameter.

Beispiel *WPCLEAR ;Werkstücknullpunkt löschen*

Laden WS-Nullpkt **WPREGn X.... Y.... Z.... A....**

Laden einer Werkstücknullpunktposition in das Werkstücknullpunktregister n (zum späteren Gebrauch). Der Parameter n ist der Index des benutzten Registers.

Die übergebenen Koordinaten sind Absolutwerte und beziehen sich auf den Maschinennullpunkt (Position nach Referenzfahrt).

Die einzelnen Werkstücknullpunkte können mit der Funktion WPREGnACT aktiviert werden, wobei 'n' den Index des gewünschten Registers angibt.

In Abhängigkeit vom benutzten Steuerungsprogramm können zwei oder mehr Werkstücknullpunkte verwaltet werden:

n = 2 falls Sie PRO-PAL oder PRO-DIN als Steuerungsprogramm benutzen

n = 8 falls Sie REMOTE für Servomotoranlagen oder REMOTE für Schrittmotoranlagen benutzen

Beispiel

```
WPREG1 X100000 Y50000 Z-50000    ;Register 1 mit Pos. laden
WPREG2 X100000 Y100000 Z-50000   ;Register 2 mit Pos. laden
WPREG1ACT                        ;Register 1 aktivieren
MOVEABS X0 Y0 Z0                 ;an die Pos. (0,0,0) fahren
WPREG2ACT                        ;Register 2 aktivieren
MOVEABS X0 Y0 Z0                 ;an die Position (0,0,0)
                ;fahren, die Position Y ist nun um 50 mm verschoben
```

WS-Nullpkt-Reg aktivieren

WPREGnACT

Diese Funktion aktiviert den Werkstücknullpunkt, der zuvor in das Register ‘n’ geladen wurde. Der Parameter ‘n’ ist der Index des gewünschten Registers, er ist im Unterschied zu den meisten anderen Befehlen Bestandteil des Kommandos. Es dürfen also keine Leerzeichen vor oder hinter n stehen.

Beispiel siehe Befehl WPREGn

WS-Null teachen

WPTEACH

Während der Bearbeitung wird eine Teachbox geöffnet, mit deren Hilfe Sie einen neuen Werkstücknullpunkt anteachen können. Mit OK wird die aktuelle Position als neuer Werkstücknullpunkt übernommen; das Beenden mit ESC behält den alten Werkstücknullpunkt bei.

Dieser Befehl steht nur zur Verfügung, wenn Sie das Programm REMOTE für Servomotoranlagen oder REMOTE für Schrittmotoranlagen benutzen.

Beispiel

```
WPCLEAR                 ;aktuellen Werkstücknullpunkt löschen
MOVEABS X0 Y0 Z-0       ;an dieser Stelle wird ein neuer
                        ;Werkstücknullpunkt benötigt
WPTEACH                 ;Öffnen der Teachbox
MOVEABS X10000 Y20000   ;Weiter mit der Bearbeitung
MOVEABS Z-15000
```

Port setzen

SETPORT An=v

Setzen eines Ausgabeports.

Die Zuweisung einer physikalischen Adresse zu einer logischen Adresse wird im Einstellprogramm des Treibers vorgenommen. Das Einrichten der logischen Adresse ist für die Benutzung dieser Funktion erforderlich.

Der Parameter ‘n’ (Ziffer zwischen 1 und 4) gibt die logische Adresse eines Ausgabeports an. Der Parameter ‘v’ ist der auszugebende Wert. Das Format des Ausgabewertes richtet sich nach dem angehängten Buchstaben.

Beispiel

```
SETPORT A1=100101B      ;setzt den Binärwert 00100101
SETPORT A1=42D          ;setzt den Dezimalwert 42
SETPORT A1=F2H          ;setzt den Hexadezimalwert F2
```

Bit setzen

SETBIT An.b

Setzen eines einzelnen Bits eines Ausgabeports.
Die Zuweisung einer physikalischen Adresse zu einer logischen Adresse wird im Einstellprogramm des Treibers vorgenommen. Das Einrichten der logischen Adresse ist für die Benutzung dieser Funktion erforderlich.
Der Parameter ‘n’ (Ziffer zwischen 1 und 4) gibt die logische Adresse des Ausgabeports an, der Parameter ‘b’ (Ziffer zwischen 1 und 8) kennzeichnet die Bitnummer des zu setzenden Bits.

Beispiel

```
SETBIT A1.4          ;setzt Bit 4 von Ausgabeport 1 auf „1“
SETBIT A2.1          ;setzt Bit 1 von Ausgabeport 2 auf „1“
```

Bit löschen

RESBIT An.b

Löschen eines einzelnen Bits eines Ausgabeports.
Die Zuweisung einer physikalischen Adresse zu einer logischen Adresse wird im Einstellprogramm des Treibers vorgenommen. Das Einrichten der logischen Adresse ist für die Benutzung dieser Funktion erforderlich.
Der Parameter n (eine Ziffer zwischen 1 und 4) gibt die logische Adresse des Ausgabeports an, der Parameter b (eine Ziffer zwischen 1 und 8) kennzeichnet die Bitnummer des zu löschenden Bits.

Beispiel

```
RESBIT A1.4          ;setzt Bit 4 von Ausgabeport 1 auf „0“
RESBIT A2.1          ;setzt Bit 1 von Ausgabeport 2 auf „0“
```

Warten auf Bitwert

WAITBIT Ep.n==v

Warten auf einen Bitwert.
Die Zuweisung einer physikalischen Adresse zu einer logischen Adresse wird im Einstellprogramm des Treibers vorgenommen. Das Einrichten der logischen Adresse ist für die Benutzung dieser Funktion erforderlich.
Das Steuerungsprogramm stoppt die Bearbeitung und wartet, bis der definierte Bitwert am angegebenen Eingangskanal auftritt. Der Parameter ‘E’ gibt die logische Portadresse (Ziffer zwischen 1 und 4) sowie Bitnummer (Ziffer zwischen 1 und 8) des zu prüfenden Eingangskanals an. Der Parameter ‘v’ hinter dem Vergleichsoperator ist der logische Pegel, auf den gewartet werden soll.
Für Test- oder Kontrollzwecke ist es möglich, diese Funktion abzubrechen bzw. zu überspringen.

Beispiel

```
MOVEABS X0 Y0          ;Bearbeitung ...
WAITBIT E1.4==1        ;Warten bis Bit 4 an Port 1 high wird
MOVEABS X20000 Y5000   ;weiter mit der Bearbeitung ...
```

Warten auf Bitmuster

WAITPORT Ep==v

Warten auf Bitmuster an einem Eingangsport
Die Zuweisung einer physikalischen Adresse zu einer logischen Adresse wird im Einstellprogramm des Treibers vorgenommen. Das Einrichten der logischen Adresse ist für die Benutzung dieser Funktion erforderlich.
Das Steuerungsprogramm stoppt die Bearbeitung und wartet, bis das definierte Bitmuster am angegebenen Eingangsport auftritt. Der Parameter 'E' gibt die logische Portadresse an (Ziffer zwischen 1 und 4). Der Parameter 'v' hinter dem Vergleichsoperator ist der 8-Bit-Wert, auf den gewartet werden soll.
Für Test- oder Kontrollzwecke ist es möglich, diese Funktion abzubrechen bzw. zu überspringen.
Beispiel

```
  MOVEABS Z0                  ;Bearbeitung ...
  MOVEABS X0 Y0               ;Bearbeitung ...
  WAITPORT E1==00100101B      ;warte auf Binärwert 00100101
oder
  WAITPORT E1==37D            ;warte auf Dezimalwert 37
oder
   WAITPORT E1==25D           ;warte auf Hexadezimalwert 25
  MOVEABS X20000 Y5000;weiter mit der Bearbeitung ...
```

Beginn Bahnbearbeitung

PATH

Startet die Bearbeitung eines Bahndatenfeldes. Das Starten und Beenden der Bearbeitung eines Bahndatenfeldes sind intern benutzte Funktionen für einen besonderen Arbeits-Modus des Steuerungsprogrammes.
Ein Bahndatenfeld bzw. eine Datei mit Bahndatenfeldern wird durch ein spezielles Hilfsprogramm von *iselautomation* erstellt, einem sogenannten Bahndatengenerator. Anwender sollten auf keinen Fall eigene Bahndatenfelder generieren.

Ende Bahnbearbeitung

PATHEND

Kennzeichnet das Ende eines Bahndatenfeldes. Das Starten und Beenden der Bearbeitung eines Bahndatenfeldes sind intern benutzte Funktionen für einen besonderen Arbeits-Modus des Steuerungsprogrammes.
Ein Bahndatenfeld bzw. eine Datei mit Bahndatenfeldern wird durch ein spezielles Hilfsprogramm von *iselautomation*, einen sogenannten Bahndatengenerator, erstellt. Anwender sollten auf keinen Fall eigene Bahndatenfelder generieren.

Werkzeugwechsel **GETTOOL x**

Ruft das Hilfsprogramm für den automatischen Werkzeugwechsel auf. Der Parameter x gibt die Nummer des neuen Werkzeuges an. Bevor Sie diesen Befehl benutzen können, müssen Sie das Werkzeugwechselprogramm (ITC) selbst einrichten sowie die Benutzung des Werkzeugwechselprogramms in der Steuerungsoberfläche (REMOTE) festlegen.

Beispiel

```
MOVEABS Z5000                ;Bearbeitung ...
FASTABS X5000 Y5000          ;Bearbeitung ...
COOLANT OFF                  ;Kühlmittelpumpe aus
SPINDLE OFF                  ;Spindel aus
GETTOOL 3                    ;Neues Werkzeug 3 holen
SPINDLE ON                   ;Spindel wieder einschalten
MOVEABS Z10000               ;Sicherheitshöhe
FASTABS X150000 Y200000      ;Positionieren ...
COOLANT ON                   ;Kühlmittelpumpe ein
```

Definition Bohrzyklus **DRILLDEFC**<1> **P**<2> **D**<3> **T**<4> **V**<5> **F**<6> **O**<7> **I**<8> **R**<9> **L**<10> **S**<11>

Ein Bohrzyklus wird definiert, ohne die Bohrung unmittelbar auszuführen. Dieser Befehl wird benutzt, um alle Bohrparameter für den nachfolgenden Befehl DRILL zu setzen.

Die Handhabung der Bohrparameter ist modal, d. h. wenn ein Parameter einmal gesetzt ist, bleibt diese Einstellung bis zum nächsten Aufruf des Befehls sowie Parameters gültig.

Die Angabe der Bohrparameter kann in einer oder mehreren Zeilen erfolgen, z. B. um generelle Einstellungen zu Beginn der NC-Datei vorzunehmen und veränderliche Einstellungen vor Aufruf des jeweiligen Bohrbefehls.

Parameter	benutzt in Zyklus	Voreinstellwert	Bedeutung
C		1	Angabe des Bohrzyklus: 1 Einfaches Bohren 2 Tieflochbohren bzw. Ausräumen 3 Spanbrechen
P	1,2,3	0	Referenzebene, auf die sich alle weiteren Parameter des Bohrzyklus' beziehen. Die Referenzebene bezieht sich auf den definierten Werkstücknullpunkt, der Wert der Z-Koordinate gibt den Abstand dazu an. Die Angabe erfolgt in Mikrometern [µm]. Vorzeichen: Positiv, falls Z-Position oberhalb des Werkstücknullpunktes.
D	1,2,3	0	Tiefe des Bohrlochs in [mm], relativ zur Referenzebene. Vorzeichen: Positiv, wenn Zustellung nach unten.
T	1,2,3	0	Verzögerungszeit für Freischnitt bei Erreichen des Lochendes, Angabe in Millisekunden [ms].
V	1,2,3	1000	Vorschubgeschwindigkeit für Bohren, Angabe in [mm/s]. Als Geschwindigkeit für Eilbewegungen wird die zuletzt durch den Zwischenformat-Befehl FASTVEL eingestellte Eilgeschwindigkeit bzw. die Default-Eilgeschwindigkeit des Steuerungsprogramms benutzt.
F	2,3	0	Erste inkrementelle Zustelltiefe für Ausräumen und Spanbrechen, Angabe in [mm]. Vorzeichen: Positiv, wenn Zustellung nach unten.
O	2,3	0	Alle weiteren inkrementellen Zustelltiefen für Ausräumen und Spanbrechen, Angabe in [mm]. Vorzeichen: Positiv, wenn Zustellung nach unten.
I	2,3	0	Abnahme der inkrementellen Zustelltiefen pro Arbeitsgang. Vorzeichen: Positiv, wenn inkr. Zustelltiefe verringert wird.
R	3	0	Inkrementeller Rückzug für Spanbrechen oder Differenzhöhe für das Wiederanfahren der letzten Bohrtiefe im Eilgang bei Tieflochbohren. Angabe in [mm]. Vorzeichen: Positiv, da Rückzugwert angegeben wird.
L	1,2,3	0	Rückzugshöhe bezüglich der definierten Referenzebene nach dem Ende des Bohrens. Angabe in [mm]. Vorzeichen: Positiv, wenn Z-Position oberhalb des Werkstücknullpunktes.
S	1,2,3	0	Relative Sicherheitshöhe, die nach dem Ende des Bohrens relativ zur Referenzebene eingestellt wird. Angabe in [mm]. Vorzeichen: Positiv, wenn Z-Position oberhalb des Werkstücknullpunktes.

Beispiel siehe Befehl DRILL

Loch Bohren

DRILL X... Y...

Bohrung an der Position (X, Y).
Die Parameter des Bohrzyklus' und der auszuführende Bohrzyklus müssen zuvor mit dem Befehl DRILLDEF festgelegt worden sein.
Die Angabe der Bohrposition erfolgt mit den Parametern X und Y, die Einheit der Position ist Mikrometer [µm].

Beispiel

```
DRILLDEF P2000      ;Referenzebene 2 mm über dem
                    ;definierten Werkstücknullpunkt Z
DRILLDEF D20000     ;Bohrtiefe 20 mm relativ zur Referenzebene
DRILLDEF T1000      ;Zeitverzögerung am Lochende 1 Sekunde
DRILLDEF V10000     ;Vorschubgeschw. für Bohren 10 mm/s
DRILLDEF F7000      ;Erste Zustelltiefe der Bohrung ist 7 mm
DRILLDEF O5000      ;Alle weiteren Zustelltiefen sind 5 mm und
DRILLDEF I1000      ;nehmen pro Zustellung um 1 mm ab
DRILLDEF R1000      ;Der inkrementelle Rückzug beträgt 1 mm
DRILLDEF L1000      ;Rückzugshöhe relativ zur Referenzebene
DRILLDEF S3000      ;Sicherheitshöhe relativ zur Referenzebene
DRILLDEF C1           ;Bohrzyklus 1, Einfaches Bohren, aktivieren
DRILL X10000 Y30000 ;An Position X = 10 mm, Y = 30 mm bohren
DRILLDEF C2           ;Bohrzyklus 2, Tieflochbohren, aktivieren
DRILL X30000 Y30000 ;An Position X = 30 mm, Y = 30 mm bohren
DRILLDEF C3           ;Bohrzyklus 3, Spanbrechen, aktivieren
DRILL X50000 Y30000 ;An Position X = 50 mm, Y = 30 mm bohren
```

Zylinderradius

CYL X(Y,Z,A)radius X(Y,Z,A)

Angabe des Radius' eines in einer Drehachse aufgespannten Zylinders
Falls Sie eine Zwischenformatdatei für die Bearbeitung eines zylindrischen Drehkörpers gestalten wollen, können Sie diesen Befehl benutzen. Der Treiber rechnet dann die linearen Geschwindigkeiten auf der Zylinderoberfläche in entsprechende Drehgeschwindigkeiten um.
Der erste Parameter gibt die Maschinenachse an, die als Drehachse konfiguriert ist. Nachfolgend angegeben ist der Radius des eingespannten Zylinders, die Angabe erfolgt in Millimetern.
Der zweite Parameter gibt an, welche der drei Achsen X, Y oder Z eines kartesischen Koordinatensystems durch die Drehachse ersetzt wird.
Zu beachten: Auf einer Zylinderoberfläche kann keine Kreis- oder Helixinterpoaltion stattfinden.
In die Originalkonfiguration zurückschalten können Sie mit der Übergabe des Radius' 0, z. B.: CYL A0.

Beispiel

```
CYL A5000 A           ;Zylinderradius 5 mm, Drehachse ist Achse A
```

Bearbeitung anhalten

HALT

Anhalten der laufenden Bearbeitung und Umschalten in den Bearbeitungs-Modus „Einzelschritt“.
Die Bearbeitung kann schrittweise oder kontinuierlich fortgesetzt werden, nachdem die an dieser Stelle notwendigen Aktionen von Benutzerseite ausgeführt wurden.
In der NC-Datei kann an entsprechender Stelle ein Kommentar stehen, der die erforderlichen Aktionen auf Seite des Benutzers beschreibt.

Beispiel

```
SPINDLE OFF                 ;Spindel ausschalten
MOVEABS Z5000               ;Bearbeitung ...
FASTABS X5000 Y5000         ;Bearbeitung ...
                            ;Kommentar mit Anweisungen für den
                            ;Bediener, z. B.:
HALT                        ;Drehen Sie das Werkstücks um 90° nach
                            ;links und setzen Sie die Bearbeitung fort
MOVEABS Z10000              ;Sicherheitshöhe
FASTABS X15000 Y20000       ;Positionieren ...
SPINDLE ON TIME2000         ;Spindel wieder einschalten
```

Rampen*)

ACCEL X.... Y... Z.... A....

Einstellen der Beschleunigungswerte für die Achsen.
Die Angabe des Wertes erfolgt in Prozent. Der Wert der Beschleunigung kann in einem Bereich von 5 % ... 100 % liegen.
Der maximale Beschleunigungswert (100 %) wurde in der Initialisierungsdatei eingestellt.

Spindel ein/aus

SPINDLE CW/CCW RPMxxx RPSxxx TIMExxx

Ein-/Ausschalten der Antriebsspindel. CW bzw. CCW bestimmen die Drehrichtung der Spindel, der Parameter RPM (RPS) die Drehzahl in Umdrehungen/Minute [U/s].
Eine Programmverzögerung für den Anlauf der Spindel auf Nenndrehzahl kann durch den Parameter TIME angegeben werden.
Die Angabe der Zeit erfolgt in Millisekunden [ms].

Kühlmittel ein/aus

COOLANT ON/OFF

Ein-/Ausschalten der Kühlmittelpumpe.
Die Festlegung der Art der Kühlmittelpumpensteuerung erfolgt im Interpreterteil des NC-Steuerungsprogramms.

Def. Werkzeug

TOOLDEF Tx Ly Rz

Bekanntgabe der verwendeten bzw. bestückten Werkzeuge.
(Für den späteren Gebrauch)

Init WZWechsel

INITTOOL
Initialisieren des angeschlossenen Werkzeugwechslers/des Werkzeugwechselteils der CNC-Software. Durchführen von WZ-Wechsel-Initialisierung, Feststellen der aktuellen Konfiguration, Referenzläufe etc.
(Für den späteren Gebrauch)

Programmende

PROGEND der NC-Datei
Markiert das Ende der NC-Datei. Abhängig von den Einstellungen im Steuerungsprogramm wartet der Interpreter entweder auf Anweisungen zum Neustart der Bearbeitung oder das Ausgabefenster wird automatisch geschlossen und die Haupteingabemaske des Steuerungsprogramms wieder angezeigt.
Beispiel

```
SPINDLE OFF      ;Spindel ausschalten
COOLANT OFF      ;Kühlmittelpumpe ausschalten
REF XYZ          ;Achsen an definierte Positionen fahren
PROGEND          ;Programmende kennzeichnen
```

Spindel-Modus der letzten Achse ein/aus

SEPAXIS ON/OFF
Der Spindel-Modus für die letzte Achse wird ein- oder ausgeschaltet. Es bedeutet, dass die letzte Achse aus dem Achsverbund getrennt oder wieder eingefügt wird.

Änderung der Geschwindigkeit der letzten Achse im Spindel-Modus

SPINDELVEL xxx
Die Geschwindigkeit der im Spindel-Modus stehenden letzten Achse wird während der Bahnbearbeitung gesetzt.

Bit setzen während der Bahn

PATHSETBIT x
Setzen eines Ausgabe-Bit mit dem Wert 1. Das Programm PARKON ermöglich das Definieren von 4 logischen Ausgabeports. Daraus entstehen 32 logische Ausgabekanäle, die von 0 ... 31 nummeriert sind. Die Nummer hinter dem Befehl ist die Kanalnummer (siehe Abs. 3.3.36 und 3.3.38).

Bit rücksetzen während der Bahn

PATHRESBIT x
Setzen eines Ausgabe-Bit mit dem Wert 0. Das Programm PARKON ermöglicht das Definieren von 4 logischen Ausgabeports. Daraus entstehen 32 logische Ausgabekanäle, die von 0 ... 31 nummeriert sind. Die Nummer hinter dem Befehl ist die Kanalnummer (siehe Abs. 3.3.36 und 3.3.38).

Beispiel für eine NCP-Quelldatei

Die Beispieldatei wurde mit einem HP/GL-Konverter erstellt und zeigt das Ausfräsen von drei Rechtecken.

```
N000001 IMF_PBL_V1.0 - HPREMOTE V1.32 - PP FILE
N000002 ;***********************************************
N000003 ; 3RECTS.NCP Fri Mar 01 12:04:29 1996
N000004 ;***********************************************
N000005 VEL 5000
N000006 FASTVEL 50000
N000007 MOVEABS Z-3000
N000008 VEL 8000
N000009 FASTVEL 50000
N000010 FASTABS X53375 Y0 Z-3000
N000011 MOVEABS Z5000
N000012 VEL 12000
N000013 MOVEREL Y69625
N000014 MOVEREL X67625
N000015 MOVEREL Y-69625
N000016 MOVEREL X-67625
N000017 VEL 10000
N000018 MOVEABS Z-3000
N000019 VEL 5000
N000020 FASTVEL 50000
N000021 FASTABS X30625 Y54000 Z-3000
N000022 MOVEABS Z7000
N000023 VEL 12000
N000024 MOVEREL Y76250
N000025 MOVEREL X72875
N000026 MOVEREL Y-76250
N000027 MOVEREL X-72875
N000028 VEL 10000
N000029 MOVEABS Z-3000
N000030 VEL 5000
N000031 FASTVEL 50000
N000032 FASTABS X0 Y16875 Z-3000
N000033 MOVEABS Z1000
N000034 MOVEREL Y65875
N000035 MOVEREL X61250
N000036 MOVEREL Y-65875
N000037 MOVEREL X-61250
N000038 MOVEABS Z-3000
N000039 PROGEND
```

PARKON - Konfigurationsprogramm

1 Einleitung

Sie können mit dem Konfigurationsprogramm PARKON.EXE eine sogenannte Initialisierungsdatei erzeugen und alle systemspezifischen Parameter Ihrer Anlage in diese Datei eingeben. Die Initialisierungsdatei wird vom Treiber bei dessen Start eingelesen. Aus Sicherheitsgründen ist die Initialisierungsdatei schreibgeschützt. Somit ist eine unbeabsichtigte Änderung der Datei nahezu unmöglich.
Eine Änderung der Parameter sollten Sie generell über das Konfigurationsprogramm vornehmen, weil die Initialisierungsdatei intern ein eigenes Format hat. Eine kleine Änderung des internen Dateiformats kann schon dazu führen, dass der Treiber beim Start die Datei nicht mehr lesen kann.

Beim Neustarten des Konfigurationsprogramms müssen Sie zuerst die Struktur Ihrer Anlage definieren, um gesperrte Eingabefelder freizubekommen.
Diese Vorgehensweise ist deshalb notwendig, weil die Struktur der Anlage (TTT_Struktur oder nicht, Achsenanzahl, Achsenart) darüber entscheidet, welche Einheiten die Position, die Geschwindigkeit sowie die Beschleunigung haben und welche Eingabefelder zugelassen oder gesperrt sind.
Die Bedienung des Konfigurationsprogramms ist sehr einfach nicht nur wegen des SAA-Standards (**S**ystem **A**pplication **A**rchitektur) sondern auch wegen der On-Line-Hilfe.
Im Statusfenster können Sie immer sofort erkennen, an welcher Datei Sie momentan arbeiten. In den folgenden Abschnitten werden wir Ihnen die einzelnen Menüs und Untermenüs genau erläutern.

2 Das Hauptmenü *Datei*

In diesem Hauptmenü können Sie eine Initialisierungsdatei erzeugen, öffnen, konvertieren, speichern und drucken. Das Konfigurationsprogramm wird ebenfalls in diesem Hauptmenü beendet.

Untermenü *Neu*
Damit können Sie eine neue Initialisierungsdatei erzeugen. Alle Initialisierungsdateien haben die Namenserweiterung 'INI'. Nach der Auswahl dieses Untermenüs fragt das Programm, ob Sie eine eventuell vorhandene Datei speichern wollen, falls diese alte Datei geändert wurde.
Ein Teil des Hauptmenüs *Hardware* und das ganze Hauptmenü *Software* sind gesperrt. Sie müssen die Struktur Ihrer Anlage zuerst eingeben, um diese Menüs frei zu bekommen.

Untermenü *Öffnen*

Eine bereits vorhandene Initialisierungsdatei wird geöffnet. Falls diese Datei ein falsches Format hat, wird Formatfehler angezeigt. Ein Dateinamenfehler wird angezeigt, wenn Sie eine Datei mit einer anderen Namenserweiterung als 'INI' öffnen wollen. Bevor eine Datei geöffnet wird, fordert das Konfigurationsprogramm Sie auf, eine in Bearbeitung befindliche Datei zu speichern, falls diese Datei von Ihnen vorher geändert wurde.
Nachdem die Datei erfolgreich geöffnet wurde, erscheinen der Dateiname mit dem Verzeichnispfad sowie das letzte Änderungsdatum der Datei im Statusfenster. Ein von Ihnen eingegebener Kommentar zur Identifikation der Datei wird in diesem Satusfenster auch angezeigt (siehe Untermenü *Bemerkung* im Hauptmenü *Hardware*).

Untermenü *Konvertieren*

Eine Initialisierungsdatei der Version 3.00 kann hier in eine Initialisierungsdatei der aktuellen Version umgewandelt werden. Einige Parameter der alten Version werden weggelassen. Viele neue Einstellmöglichkeiten kommen hinzu. Deswegen ist es notwendig, über den entsprechenden Menüs die umgewandelte Datei zu vervollständigen. Falls Sie es nicht tun, werden immer Standardwerte genommen. Die Unterschiede zwischen der Version 3.00 und der Version 3.10 sind hier aufgelistet.

- Im Untermenü *Ref_Schalter* des Hauptmenüs *Hardware* der Version 3.10 ist das Feld „Watch-Dog“ nicht mehr vorhanden. Das Überbrücken der Endlageschalter bei der Referenzfahrt wird bei der Version 3.10 nicht mehr durch das Watch-Dog-Signal sondern durch eine separate Treiberfunktion realisiert
  (siehe Handbuch *isel*-Servosteuerkarte UPMV 4/12).

- Anstelle des Felds „Watch-Dog“ haben Sie das Feld „KeyCode“, mit dem Sie eine Tastaturtaste für die Referenzfahrtunterbrechung definieren können.

- Bei der früheren Treiberversion dient der Schlüsselschalter zum Überbrücken der Hardware-Endlageschalter. Damit ist es möglich, die Leistungsendstufen bei Hardware-Endlagerschalterfehlern einzuschalten und die Achsen aus den Endlageschaltern herauszufahren. Bei der neuen Treiberversion wird das Überbrücken der Hardware-Endlageschalter durch eine separate Treiberfunktion realisiert (siehe Handbuch *isel*-Servosteuerkarte UPMV 4/12).

- Die Funktionalität des Schlüsselschalters wird nicht mehr unterstützt. Deswegen gibt es das Untermenü Schlüsselschalter nicht. Beim Konvertieren werden die Parameter des Schlüsselschalters einfach weggelassen.
- Das Untermenü *Überwachungsport* im Hauptmenü *Hardware* sowie das Untermenü *V_Achsfaktor* im Hauptmenü *Software* sind komplett neu

hinzugekommen. Alle Einstellparameter in diesen beiden Menüs müssen neu gesetzt werden, falls Sie die Standardwerte nicht haben möchten.

- Beim Konvertieren der Initialisierungsdatei werden die Reduktionsfaktoren für Kreis- und Bahnfahren generell auf 1 gesetzt. Durch das Einführen der V-Achsverstärkungsfaktoren ist es notwendig, die Reduktionsfaktoren auf die Standardwerte 1 zu setzen. Die Reduktionsfaktoren können Sie hinterher immer noch ändern, falls Sie das für nötig halten.

- Im Untermenü *Regler* des Hauptmenüs *Software* können Sie bei der Version 3.10 definieren, ob die Positionsregler Ihrer Anlage bei einem Nachlauffehler automatisch abgeschaltet werden sollen oder nicht. Diese Maßnahmen ist wichtig für die Strombegrenzung.

Untermenü *Speichern*
Die momentan in Bearbeitung befindliche Datei wird unter ihrem alten Namen gespeichert. Falls es sich um eine neue Datei handelt, müssen Sie zuerst noch das Verzeichnis auswählen und den Dateinamen eingeben. Danach werden der Dateiname mit dem Verzeichnis und das aktuelle Datum im Statusfenster angezeigt. Wenn der von Ihnen eingegebene Dateiname nicht die Erweiterung 'INI' hat, wird ein Dateinamenfehler angezeigt.

Untermenü *Speichern als ...*
Die momentan in Bearbeitung befindliche Datei wird unter einem neuen Namen gespeichert. Sie müssen natürlich zuerst das Verzeichnis auswählen und den neuen Dateiname angeben. Ein Dateiname mit einer anderen Erweiterung als 'INI' wird nicht akzeptiert und ein Dateinamenfehler angezeigt. Nachdem die Datei erfolgreich gespeichert wurde, erscheint im Statusfenster der neue Dateiname mit dem Verzeichnis sowie das aktuelle Datum.

Untermenü *Drucken*
Die momentan in Bearbeitung befindliche Datei wird gedruckt. Auf der Druckliste mit sehr deutlichen Kommentaren und Parameternamen können Sie problemlos alle eingegebenen Daten kontrollieren. Ein eventuell vorhandener Druckerfehler wird sofort angezeigt.

Untermenü *Beenden*
Hiermit beenden Sie das Konfigurationsprogramm. Es fordert Sie auf, die in Bearbeitung befindliche Datei zu speichern, falls diese Datei geändert wurde.

3 Das Hauptmenü *Hardware*

Alle Parameter, die mit der Hardware bzw. der Mechanik zusammenhängen, können in diesem Hauptmenü eingegeben werden.

Untermenü *Bemerkung*
Hier haben Sie die Möglichkeit, bis zu 30 ASCII-Zeichen einzugeben.
Diese Zeichenkette erscheint sofort im Statusfenster. Sie wird außerdem intern in der Datei abgelegt. Jedes Mal, wenn Sie die Datei öffnen, wird sie zusammen mit dem Dateinamen und dem letzten Änderungsdatum im Statusfenster angezeigt.
Mit der Eingabe einer eigenen Zeichenkette können Sie die verschiedenen Initialisierungsdateien besser voneinander unterscheiden.

Untermenü *Ressourcen*
Hier können Sie festlegen, wie viel Prozent der Rechenleistung von Ihrem PC der Treiber beanspruchen darf. Beim Starten kontrolliert der Treiber selbst, ob er genug Rechenzeit bekommt. Falls nicht werden folgende Fehlermeldungen ausgegeben:

> “Rechner ist zu langsam. Treiber wurde nicht installiert.“

Oder

> "Warnung:
> Rechner ist langsam. Installation des Treibers erfolgt mit der Taste J."

Sie müssen den Prozentsatz erhöhen. Der Grenzwert von 75 % sollte aber nicht überschritten werden, weil Ihr Anwenderprogramm selbst auch Rechenzeit benötigt. Falls es trotzdem nicht geht, müssen Sie einen Co-Prozessor in ihren Rechner einbauen. Das Fehlen eines Co-Prozessors ist in den meisten Fällen der Grund für diese Fehlermeldung.

Untermenü *Interrupt*
Hier können Sie den Software- und Hardware-Interrupt definieren.
Über den Software-Interrupt können Sie mit dem Treiber kommunizieren.
Die Eingabe des Software-Interrupts muss in Hexadezimalzahlen erfolgen.
Die Auswahl der Nummer des Software-Interrupts verlangt von Ihnen höchste Sorgfalt. Der Interrupt darf noch nicht von einem anderen Programm benutzt werden.
Für uns ist es natürlich unmöglich, Ihnen zu sagen, welcher Interrupt momentan in Ihrem Rechner noch frei ist, weil dies von der konkreten Konfiguration des Rechners abhängt. Um zu bestimmen, welche Interrupts momentan noch frei sind, sollten Sie in der Interrupt-Tabelle Ihres Rechners nachsehen. Die Interrupt-Tabelle liegt im Speicherbereich von 0000:0000 bis

0000:03FF. Es gibt 100h Interrupt-Nummern. Jede Interrupt-Nummer belegt in der Tabelle eine 4 Byte Sprungadresse. Die Nummer 0 belegt den Speicherbereich 0000:0000 bis 0000:0003 und die Nummer 100h belegt den Bereich 0000:03FC bis 0000:03FF. Wenn der Speicherbereich eines Interrupts nur mit 00h geladen ist, ist dieser Interrupt noch frei.
Die meisten Interrupts sind für das Betriebsystem reserviert. Die restlichen stehen allen Anwenderprogrammen frei zur Verfügung. Nach unserer Erfahrung sind die Interrupts 60h ... 66h sowie 78h ... 7Fh meistens frei. Die Benutzung eines Interrupts in diesen Bereichen ist empfehlenswert. Es gibt natürlich einige Programme, die die Interrupt-Tabelle für Sie analysieren und die freien Interrupt-Nummern anzeigen wie z. B. das sehr populäre Programm NORTON COMMANDER von Symantec oder das Programm PC Tools von Central Point.

Intern braucht der Treiber einen Hardware-Interrupt. Sie haben die Auswahl zwischen dem Hardware-Interrupt IRQ10 und dem Hardware-Interrupt IRQ11. Die Interrupts IRQ10 und IRQ11 sind normalerweise für die seriellen Schnittstellen COM3 und COM4 reserviert. Die meistens PC's benutzen aber nur die seriellen Schnittstellen COM1 und COM2, d. h. in den meisten Fällen sind IRQ10 und IRQ11 noch frei. Wenn diese beiden Interrupts in Ihrem PC bereits besetzt sein sollten, müssen Sie, erst einen frei machen, um den Treiber nutzen zu können.
Die Definition des Hardware-Interrupts muss mit der hardwarmäßigen Einstellung des Interrupts des entsprechenden Jumpers auf der PC-Einsteckkarte übereinstimmen (siehe Handbuch *isel*-Servosteuerkarte UPMV 4/12), da sonst gar nichts läuft.
Nachdem Sie alle Einstellungen für den Hardware-Interrupt erledigt haben, brauchen Sie sich darum nicht weiter zu kümmern. Dies ist dann eine interne Sache des Treibers. Für Sie ist dann nur noch der oben erwähnte Software-Interrupt von Bedeutung.

Untermenü *Basisadresse*
Hier geben Sie die Basisadresse, die Sie auf den entsprechenden Jumpern der PC-Einsteckkarte eingestellt haben, ein (siehe Handbuch *isel*-Servosteuerkarte UPMV 4/12). Beachten Sie, dass die Adresse in Hexadezimalzahlen eingegeben werden muss.
Aus dieser Basisadresse berechnet der Treiber intern alle für ihn notwendigen Adressen, wie z. B. die Adressen der Achscontroller. Wenn die hier eingegebene Adresse und die hardwarermäßig eingestellte Adresse nicht übereinstimmen, erscheint beim Starten des Treibers folgende Fehlermeldung auf dem Monitor:

"Keine bzw. defekte PC-Einsteckkarte oder falsche Konfiguration"

Der Treiber wird dann nicht installiert.

Untermenü *Struktur*

In diesem Menü definieren Sie die Achsenanzahl, die Art der einzelnen Achsen und die Struktur Ihrer Anlage. Nur nachdem Sie diese Parameter eingegeben haben, werden die beim Programmstart gesperrten Menüs für Sie zugänglich. Die hier eingegebenen Parameter entscheiden, welche Einheiten die Position, die Geschwindigkeit sowie die Beschleunigung haben werden und welche Parameter überflüssig bzw. notwendig für den Treiber sind. Die entsprechenden Menüs werden gesperrt oder freigegeben.
Sie sehen schon, wie wichtig die richtige Eingabe der verlangten Parameter in diesem Menü ist.

Die Achsenanzahl brauchen wir nicht weiter zu erläutern. Zu beachten ist die Beziehung zwischen der Achsenanzahl und der Zuordnung der einzelnen Achsen. Wenn die Achsenanzahl gleich 1 ist, spricht der Treiber nur die X-Achse an. Eine Achsenanzahl gleich 2 bedeutet, dass Ihre Anlage die X- und die Y-Achse hat. Bei einer Achsenanzahl gleich 3 hat Ihre Anlage die X-, die Y- und die Z-Achse. Die vierte Achse, d. h. die A-Achse, ist nur dann aktiv, wenn Sie die Achsenanzahl gleich 4 definiert haben. Alle Eingabefelder der Achsen, die vom Treiber nicht angesprochen werden, werden vom Konfigurationsprogramm gesperrt.
Bei jeder benutzten Achse müssen Sie noch angeben, ob diese Achse eine Linear- oder eine Drehachse ist. Bei einer Linearachse wird die Längeneinheit Mikrometer [μm] benutzt. Bei einer Drehachse wird die Winkeleinheit Bogensekunde ["] benutzt.

Die Achsenanzahl, die Art der einzelnen Achsen sowie die Zuordnung der Achsen entscheiden über die Struktur Ihrer Anlage.

- Ihre Anlage hat eine XYZ_TTT-Struktur, wenn Ihre Anlage mindestens 3 Achsen hat. Es sind die X- , die Y- und die Z-Achse. Diese Achsen müssen Linearachsen sein. Gleichzeitig müssen sie ein kartesisches Koordinatensystem im Raum bilden. Die A-Achse hat hier nichts zu sagen.

- Eine XY_TTT-Struktur liegt vor, wenn Ihre Anlage mindestens zwei Achsen hat. Diese beiden Achsen sind die X- und die Y-Achse. Sie müssen Linearachsen sein und bilden ein kartesisches Koordinatensystem in der XY-Ebene. Die beiden anderen Achsen Z und A haben hier keinen Einfluss.

- Eine X_TTT-Struktur liegt vor, falls Ihre Anlage mindestens eine Achse hat. Es ist die X-Achse. Diese Achse muss eine Linearachse sein. Alle anderen drei Achsen haben hier nichts zu sagen.

- Jede andere Struktur, die keine der drei oben genannten Strukturen ist, gehört zur Kategorie "KEIN_TTT_Struktur".

Die XY_TTT- oder die XYZ_TTT-Struktur ist eine notwendige Bedingung für die Kreis- und Helixinterpolation sowie für das Bahnfahren. Sie können diese Interpolationsart vom Treiber nur dann verlangen, wenn eine der beiden genannten Strukturen vorliegt und wenn Sie dies dem Treiber über dieses Menü mitteilen.

Untermenü *Richtung*

Für jede Achse hat die PC-Einsteckkarte bereits die positive und die negative Richtung festgelegt. In diesem Menü können Sie festlegen, ob diese vordefinierten Richtungen beibehalten oder geändert werden sollen.
Die Beibehaltung der vordefinierten Richtungen erfolgt durch die Auswahl der Standardeinstellung. Wenn Sie die Standardeinstellung nicht auswählen, werden die positive und die negative Richtung umgetauscht. Die Einstellung in diesem Menü sieht so aus:

[X] Standard	--->	Standardeinstellung
[ ] Standard	--->	Keine Standardeinstellung

Nachdem Sie die Achsrichtungen definiert haben, sind nur noch diese neuen Richtungen für Sie interessant. Alle Angaben, die Bezug auf die Achsrichtungen nehmen, beziehen sich dann auf die von Ihnen neu definierten Richtungen.
Hier denken wir in erster Linie an die Eingabe der positiven bzw. der negativen Positionen einer Achse, an die Definition der positiven bzw. der negativen Hardware- bzw. Software-Endschalter oder an die Referenzschalter, die auf der positiven oder auf der negativen Seite der Achsen liegen können.

Untermenü *Schalter_Aktiv*

Die Referenz- sowie die Hardware-Endschalter sind über entsprechende Eingabeports mit dem Treiber verbunden. Anhand des Signalpegels (High oder Low), der an diesen Ports anliegt, weiß der Treiber, ob der oder die Schalter momentan angefahren sind oder nicht.
In diesem Menü definieren Sie, ob der High- oder der Low-Pegel der aktive Pegel sein soll. Der aktive Pegel ist der an dem entsprechenden Eingabeport liegende Pegel, wenn der Schalter angefahren ist. Hier haben Sie die Auswahl zwischen dem Low-Pegel und dem High-Pegel. Der definierte aktive Pegel gilt dann für alle Referenzschalter und für alle Hardwareendschalter Ihrer Anlage.

Untermenü *Endschalter*

In diesem Menü definieren Sie die Portadressen und die Bitnummern, mit denen die Endschalter Ihrer Anlage verbunden sind. Alle Adressen sind als Hexadezimalzahlen anzugeben. Die Nummerierung der Bits in einem Byte fängt bei 0 an. Das höchstwertigste Bit hat die Nummer 7. Für jede Achse können Sie bis zu zwei Hardware-Endschalter definieren.

Zudem können Sie jeden Schalter sperren lassen, falls Sie diesen Endschalter in Ihrer Anlage nicht brauchen. Das Sperren und Freilassen sieht so aus:

[X] Ein/Aus --->	Benutzen des Endschalters
[ ] Ein/Aus --->	Kein Benutzen des Endschalters

Der positive bzw. der negative Hardware-Endschalter liegt auf der positiven bzw. auf der negativen Richtung der Achse. Vergessen Sie nicht, dass Sie selbst die positive bzw. die negative Richtung im Untermenü *Richtung* definiert haben.

Untermenü *Ref_Schalter*
Ähnlich wie bei den Hardware-Endschaltern definieren Sie hier die Portadressen und die Bitnummern, mit denen die Referenzschalter verbunden sind. Wir haben zwei Modi der Referenzfahrt definiert. Das sind der Standard- und der Nicht-Standard-Modus (siehe Handbuch *isel*-Servosteuerkarte UPMV 4/12).
Im Normalfall wird der Standard-Modus benutzt. Der Nicht-Standard-Modus findet seine Anwendung bei einer endlosen Drehachse mit einem Magnetschalter als Referenzschalter. Die Auswahl des Referenz-Modus sieht so aus:

[X] Modus --->	Standard-Modus der Referenzfahrt
[ ] Modus --->	Nicht-Standard-Modus der Referenzfahrt

Weiterhin müssen Sie angeben, ob die Referenzschalter auf der positiven oder negativen Seite der dazugehörigen Achse liegen. Standardmäßig liegen die Referenzschalter immer auf der negativen Seite. Damit bewegen sich die Achsen in positiver Richtung beim Herausfahren aus dem Schalter. Sie können den Referenzschalter natürlich auch auf die positiven Seite der Achsen bauen. Sie müssen dann aber in diesem Menü eingeben, dass Sie etwas nicht standardmäßig gemacht haben. Die Eingabe sieht so aus:

[X] Richtung --->	Referenzschalter auf der negativen Seite
[ ] Richtung --->	Referenzschalter auf der positiven Seite

Ähnlich wie bei den Hardware-Endschaltern sollten Sie beachten, dass Sie selbst die positive bzw. die negative Richtung im Untermenü *Richtung* definiert haben. Die Richtungen, die die PC-Einsteckkarte für jede Achse festgelegt hat, sind hier uninteressant.

Im Standard-Modus fährt die Achse zuerst mit einer konstanten Geschwindigkeit zum Referenzschalter, bis dieser Schalter betätigt wird.
Mit einer wesentlich kleineren Geschwindigkeit fährt die Achse wieder aus dem Schalter heraus. Der Punkt, in dem der Schalter seinen Pegel ändert, wird üblicherweise als der Referenzpunkt bzw. als der Maschinennullpunkt definiert.

In solchen Fällen, in denen Sie einen der Hardware-Endlageschalter als Referenzschalter benutzen, kann es beim Nullpunktanfahren wegen der Überschwingung zu einem ungewollten Hardware-Endlagerschalter-Fehler führen. Deswegen haben wir die Option "Referenzabstand" eingeführt.
So haben Sie die Möglichkeit, den Referenzpunkt um den definierten Abstand weg vom Schalter zu verschieben. Diese Option hat keine Wirkung beim Referenzfahrt-Modus "Nicht-Standard-Modus".

Sie können einen Hardware-Endlageschalter als Referenzschalter definieren, indem Sie die Portadresse und die Bitnummer des Hardware-Endschalters als die Portadresse und die Bitnummer des Referenzschalters definieren. Falls es aus Sicherheitsgründen notwendig ist, die Leistungsendstufen bei Hardware-Endlageschalter-Fehlern auszuschalten, müssen Sie bei der Treiberversion 3.00 die Watch-Dog-Option benutzen, um die Referenzfahrt ausführen zu können (siehe Handbuch *isel*-Servosteuerkarte UPMV 4/12).
Ab der Version 3.10 ist diese Option nicht mehr vorhanden. Um das Abschalten der Leistungsendstufen bei der Referenzfahrt zu verhindern, müssen Sie eine separate Treiberfunktion benutzen.

Die Referenzfahrt ist eine langwierige Geschichte. Falls aus irgendwelchen Gründen der Wunsch besteht, die Referenzfahrt zu unterbrechen, können Sie ab der Version 3.10 eine Taste definieren, mit der Sie die Referenzfahrt problemlos unterbrechen können.
Das Definieren der Taste erfolgt durch die Eingabe des Scan-Codes der Taste in das Feld „KeyCode“. Um den Scan-Code der entsprechenden Taste zu ermitteln, benutzen Sie das Einstellprogramm PAREIN.EXE. Standardmäßig ist der Wert 01 hier eingetragen. Das ist der Scan-Code für die ESC-Taste.
Der Unterschied zwischen dem Referenzschalter und Hardware-Endlageschaltern einer Achse besteht darin, dass jede benutzte Achse einen Referenzschalter benötigt, um einen Bezug für alle Positionsangaben zu haben. Aus diesem Grund haben Sie hier nicht die Möglichkeit, die Benutzung der Referenzschalter ein oder auszuschalten.

Untermenü *V_Regler*
Hier geben Sie ein, ob die Geschwindigkeitsregler für die Achsregelung vorhanden sind oder nicht. Jede Achse Ihrer Anlage hat einen Positionsregler, der durch den Chip LM628 bzw. LM629 realisiert wird. Die Geschwindigkeitsregler sind optional. Man kann mit ihnen eine wesentlich größere Dynamik der Anlage erreichen.

Untermenü *Überwachungsport*

In der Kategorie „Kontrollbyte“ kann ein Eingabeport definiert werden. Über diesen Eingabeport können Sie bis zu acht Hardware-Signale dem Treiber zurückführen. Somit können Sie dem Treiber bis zu acht externe Fehlerquellen wie z. B. Stromausfall, Kabelbruch, Encoder defekt, usw. mitteilen. Im Fehlerfall wird die momentane Bewegung sofort unterbrochen. Entsprechende Sicherheitsmaßnahmen werden vom Treiber ausgeführt, um eine mögliche Gefahr zu vermeiden. Die Benutzung der einzelnen Bits dieses Kontrollbyte-Eingabeports wird im Feld „Maske“ festgelegt. Ein Bitwert gleich 0 in diesem Feld bedeutet, dass das entsprechende Bit des Kontrollbyte-Eingabeports nicht benutzt und d. h. vom Treiber nicht bewertet wird. Falls ein Bit des Kontrollbyte-Eingabeports benutzt werden soll, muss das entsprechende Bit in der Maske gleich 1 gesetzt werden. Im Feld „Wert“ definieren Sie die fehlerfreien Werte für die entsprechende Bits des Kontrollbyte-Eingabeports.

Der Kontrollebyte-Eingabeport wird regelmäßig vom Treiber eingelesen. Einen Eingabewert ungleich als der von Ihnen definierte fehlerfreie Wert interpretiert der Treiber als ein Fehlerfall. Die durch die Maske als unbenutzt definierten Bits werden logischerweise vom Treiber nicht bewertet.

Der wichtigste Anwendungsfall des Kontrollbyte-Eingabeports ist die Überwachung des Stromausfalls. Über ein entsprechendes Bit dieses Eingabeports wird dem Treiber mitgeteilt, ob der Strom da ist oder nicht. Auch wir, im Hause *isel*, benutzen diese Möglichkeit, um den Strom zu überwachen.
Hardwaremäßig bedingt ist es bei unseren Anlagen aber nicht möglich, eindeutig zwischen einem tatsächlichen Stromausfall und einem aktiven Hand-Modus (stromloser Zustand der Leistungsendstufen) zu unterscheiden.
In diesen beiden Fällen ist das entsprechende Hardware-Signal aktiv.
Über einen internen Flag weiß der Treiber, wann der Hand-Modus aktiv ist.
In diesem Fall interpretiert der Treiber das aktive Signal nicht als Fehlerfall, selbst dann nicht, wenn der Strom tatsächlich ausgefallen ist.
Beim Deaktivieren des Hand-Modus benötigt der Controller aufgrund der Trägheit der Relais eine gewisse Zeit, bis der Strom wieder da ist.
Diese Verzögerungszeit der Relais müssen Sie dem Treiber mitteilen, damit der Treiber in diesem Fall (das Deaktivieren des Hand-Modus) während der Verzögerungszeit das aktive Hardware-Signal nicht als einen Stromausfall interpretiert.

All diese Besonderheiten werden in der Kategorie „Hand-Modus“ zusammengefasst. In das Feld „Bit“ müssen Sie die für die Überwachung des Stromausfalls zuständige Bitnummer des Kontrollbyte-Eingabeports eintragen. Beachten Sie, dass die Bits von 0 bis 7 nummeriert sind. Die Verzögerungszeit der Relais ist im Feld „Zeit“ definiert.

Wenn Sie Ihren eigenen Controller bauen und Ihre Hardware in der Lage ist, eindeutig zwischen dem Hand-Modus und dem Stromausfall zu unterscheiden, müssen Sie über das Feld „Ein/Aus“ dem Treiber mitteilen, dass Sie die oben genannten Sonderbehandlung beim Hand-Modus nicht haben möchten.
Bei der Benutzung der *isel*-Controller ist die Sonderbehandlung beim Hand-Modus ein muss. Die Eingabe in diesem Feld sieht so aus:

Ein/Aus [X] ---> Sonderbehandlung beim Hand-Modus
Ein/Aus [ ] ---> Keine Sonderbehandlung beim Hand-Modus

Die Ein- und Ausgabeports werden für die Steuerung der Peripherien benötigt. Um eine klare Trennung zwischen den maschineneigenen Peripherien wie z. B. Werkzeugwechsler, Abdeckhauben, Kühlmittelpumpe, etc. und den vom Anwender zusätzlich aufgerüsteten Peripherien zu bekommen, haben wir die Ein- und Ausgabeports in zwei Typen unterteilt.
Zum ersten Typ gehören die Ein- und Ausgabeports für die Überwachung. Diese Ports werden für die maschineneigenen Peripherien benutzt und in diesem Untermenü konfiguriert.
Zum zweiten Typ gehören die Ein- und Ausgabeports, die im Untermenü *Ein*/*Ausgabeport* definiert werden können.
In der Kategorie „Eingabe“ können Sie bis zu zwei Eingabeports für die Überwachung definieren. Ins Feld „Port“ werden die Portadressen in Hexadezimal-Format eingetragen. Über das Feld „Maske“ besteht die Möglichkeit, die Benutzung der einzelnen Bits ein- oder auszuschalten.
Ein Bitwert in der Maske gleich 1 bedeutet, dass das entsprechende Bit der Eingabeports benutzt wird.
Falls Sie ein bestimmtes Bit der Eingabeports nicht benutzen wollen, müssen Sie das entsprechende Bit in der Maske gleich 0 setzen. Falls alle Bits der Maske gleich 0 sind, werden die Eingabeports vom Treiber gar nicht abgefragt.
Über das Feld „Aktiv“ werden die Aktivpegels für die einzelnen Bits der Eingabeports definiert. Somit können Sie in Ihrem Anwenderprogramm auf der logischen Ebene arbeiten (siehe Handbuch *isel*-Servosteuerkarte UPMV 4/12).
Beim Aufruf der entsprechenden Treiberfunktionen bekommen Sie den Bitwert 1, falls der tatsächliche Bitwert und der als aktiv definierte Bitwert identisch sind. In den anderen Fällen bekommen Sie den Bitwert 0.

Ähnlich wie in der Kategorie „Eingabe“ können Sie in der Kategorie „Ausgabe“ auch bis zu zwei Ausgabeports für die Überwachung definieren.
Die Portadressen stehen im Feld „Port“. Die bitweise Benutzung der Ausgabeports kann über das Feld „Maske“ definiert werden. Wenn Sie die Ausgabeports nicht benutzen wollen, müssen Sie alle Bits im Feld „Maske“ gleich 0 setzen. Diese Ports werden dann nicht mehr vom Treiber angesprochen.

Die Aktivpegel können Sie im Feld „Aktiv“ festlegen, um auf der logischen Ebene arbeiten zu können. Die Initialisierungswerte, die der Treiber beim Start an den entsprechenden Ports ausgibt, sind im Feld „Init“ einzutragen. Beachten Sie, dass die Initialisierungswerte auch logische Werte sind (siehe Handbuch *isel*-Servosteuerkarte UPMV 4/12).
Die bitweise Benutzung der Ein- und Ausgabeports ermöglicht Ihnen den Mischbetrieb der Ein- und Ausgabeports. Ein Port, deren Bits nur teilweise hier für die Überwachung benutzt wird, kann auch über den Kanalein- und Kanalausgabeports angesprochen werden (siehe Untermenü *Ein*/*Ausgabeport*).

Untermenü *Ein*/*Ausgabeport*
In diesem Menü können Sie bis zu 4 Ein- und 4 Ausgabeports definieren, die wir Kanalein- und Kanalausgabeports nennen. Das bitweise bzw. byteweise Ansprechen dieser Ports erfolgt dann durch logische Nummer.
Im Handbuch für die *isel*-Servosteuerkarte UPMV 4/12 können Sie sich über die Vorteile informieren, die Ports durch logische Nummern anzusprechen.
Die Eingabeports werden mit den Nummer E0, E1, E2 und E3 versehen.
Die Ausgabeports werden mit den Nummern A0, A1, A2 und A3 nummeriert.
Bei den Ausgabeports müssen Sie noch die Anfangswerte der Ports definieren. Beim Starten des Treibers werden die Ausgabeports mit diesen Anfangswerten initialisiert. Die Benutzung der Ein- und Ausgabeports ist optional.
Die Definition der Benutzung jedes einzelnen Ports sieht so aus:

Ein/Aus
[X] ---> benutzt,
[ ] ---> nicht benutzt.

Untermenü *Bew_Ausgabeport*
In diesem Menü haben Sie die Möglichkeit, einen Port zu definieren, über den eine geschwindigkeitsabhängige Ausgabe während der Bearbeitungsbewegung erfolgt. Die Begrenzung des Ausgabebereichs erfolgt durch die Angabe des Minimal- und des Maximalwertes. Diese Angaben sind in Hexadezimalzahlen einzugeben.
Über die Verzögerungszeit besteht die Möglichkeit, die Ausgabe wegen der Nachlaufzeit der angeschlossene Peripherie anzupassen (siehe Handbuch *isel*-Servosteuerkarte UPMV 4/12).
Die Benutzung dieser Möglichkeiten des Treibers ist optional. Deswegen müssen Sie die Benutzung auch angeben, falls Sie diese Möglichkeit nutzen wollen.

Untermenü *Encoderlinie*
In diesem Menü geben Sie die Linienzahl des benutzten Encoders der jeweiligen Achse ein. Die Linienzahl eines Encoders wird als die Anzahl der von diesem Encoder pro Umdrehung gelieferten Inkremente definiert. Sie sollten beachten, dass die interne Vervierfachung durch den Achscontroller hier nichts zu sagen hat, d. h. wenn der Encoder 1000 Impulse pro Umdrehung liefert, geben Sie hier wirklich 1000 und nicht 4000 (4 * 1 000) ein.
Der Treiber benutzt diese Information zusammen mit dem Übertragungsfaktor der dazugehörigen Achse (siehe unten), um einen internen Übersetzungsfaktor zu berechnen.

Untermenü *Übersetzung*
Der Treiber benutzt die Informationen über die Linienzahl der Encoder zusammen mit der von Ihnen in diesem Menü eingegebenen Informationen, um interne Übersetzungsfaktoren zu berechnen. Der Achsübersetzungsfaktor wird bei einer Linearachse als die Bewegungslänge in Mikrometer pro eine Motorumdrehung und bei einer Drehachse als der Drehwinkel in Bogensekunde pro eine Motorumdrehung definiert.

4 Das Hauptmenü *Software*

Die Parameter, die in direktem Zusammenhang mit der Bewegung stehen und Grenzwerte bzw. Standardwerte sind, sowie die Reglerparameter können Sie in diesem Hauptmenü eingeben.
Achten Sie besonders auf die Einheiten der Parameter in den Eingabefeldern. Die Einheiten werden entsprechend Ihrer Auswahl im Untermenü *Struktur* vom Konfigurationsprogramm festgelegt. Deswegen ist dieses Hauptmenü gesperrt, bis Sie die Parameter im Untermenü Struktur eingegeben haben.

Untermenü *A_Max*
In diesem Menü geben Sie die maximalen Beschleunigungen der Achsen ein. Diese Beschleunigungswerte werden vom Treiber bei der Rampengenerierung für die einzelnen Achsen benutzt.
Wenn die von Ihnen eingegebenen Werte größer sind als die Werte, die von den jeweiligen Motoren erbracht werden können, werden Sie ein ungleichmäßiges Bewegungsverhalten der Achsen bekommen. Je kleiner die eingegebenen Werte sind, desto sanfter und träger sind die Achsbewegungen. Beachten Sie bei der Dimensionierung Beschleunigungswerte, dass die Motoren einen Teil ihres Beschleunigungsvermögens für die Überwindung der Last während der Bearbeitung aufbringen müssen, d. h. entsprechend der Bearbeitungslast müssen Sie die hier einzugebenden Werte kleiner wählen als die Maximalwerte, die von den Motoren im Leerlauf erbracht werden können.

Untermenü *A_Referenz*
Hier geben Sie für jede Achse einen Verkleinerungsfaktor der Beschleunigung ein. Die Referenzfahrt wird mit einer kleineren Beschleunigung als bei einer normalen Bewegung durchgeführt. Bei einem Verkleinerungsfaktor von 10 ist

die Beschleunigung während der Referenzfahrt gleich 1/10 der Beschleunigung während einer normalen Bewegung, d. h. die Bewegung während der Referenzfahrt ist wesentlich sanfter. Dadurch werden die Referenzschalter mechanisch geschont.

Untermenü *A_Sättigung*

Die Leistungsendstufen benötigen eine bestimmte Versorgungsspannung, um die benötigten Motorströme einzuprägen. Während der Bewegung erzeugen die Motorspulen aufgrund der Spulen-Induktivität eine Gegenspannung (EMK). Je schneller sich der Motor dreht, desto größer ist die Gegenspannung. Dadurch reduziert sich der Spannungsanteil für das Stromeinprägen im Motor. Weil die Versorgungsspannung in der Praxis nicht unendlich groß sein kann, verläuft die Beschleunigungskurve im oberen Drehzahlbereich flacher als im unteren Drehbereich, d. h. im oberen Drehbereich haben die Motoren einen kleineren Beschleunigungswert als im unteren Drehbereich.
Mit Hilfe dieses Sättigungsfaktors kann der Servo-Treiber den Standardwert der Beschleunigung entsprechend verkleinern, falls der Motor bzw. die Achse sich sehr schnell bewegen sollen, d. h. der Sättigungsfaktor ist nicht anders als ein Verkleinerungsfaktor. Der Beschleunigungswert im oberen Drehzahlbereich wird um diesen Faktor verkleinert.
Der Standardwert ist 10. Falls Sie nicht so genau wissen, wie dieser Faktor gesetzt werden soll, nehmen Sie einfach den Standardwert.

Untermenü *V_Max*

Die maximalen Geschwindigkeiten der einzelnen Achsen werden hier eingegeben. Die hier eingegebenen Werte sind sehr wichtige Grenzwerte für die Interpolation. Wenn die hier eingegebenen Werte größer als die tatsächlichen Maximalgeschwindigkeiten der Achsen sind, kann dies dazu führen, dass die Interpolation nicht mehr richtig funktioniert.

Untermenü *V_Segment*

Hier geben Sie den Standardwert der Geschwindigkeit während der vektoriellen Bearbeitung ein. Dieser Wert wird vom Treiber direkt nach seinem Start oder nach einem Reset als die gewünschte Bearbeitungs-Geschwindigkeit angenommen bis ein neuer Wert durch den entsprechenden Funktionsaufruf gesetzt wird. Im Handbuch für die *isel*-Servosteuerkarte UPMV 4/12 haben wir Ihnen gezeigt, wie man die Werkzeuggeschwindigkeit aus den Achsgeschwindigkeiten (und umgekehrt) berechnen kann.

Untermenü *V_Bahn*

Hier geben Sie den Standardwert der Geschwindigkeit während der 3D-Bahnbearbeitung ein. Dieser Wert wird vom Treiber direkt nach seinem Start oder nach einem Reset als die gewünschte Bahnbearbeitungs-Geschwindigkeit angenommen bis ein neuer Wert durch den entsprechenden Funktionsaufruf gesetzt wird.

Untermenü *V_Teach-In*

Durch dieses Untermenü ist die Eingabe des Standardwertes der Teach-In-Geschwindigkeit möglich. Dieser Wert wird vom Treiber direkt nach seinem Start oder nach einem Reset als die gewünschte Teach-In-Geschwindigkeit angenommen bis ein neuer Wert durch den entsprechenden Funktionsaufruf gesetzt wird.
Die verschiedenen Geschwindigkeiten und wie man sie berechnen kann, können Sie im Handbuch für die *isel*-Servosteuerkarte UPMV 4/12 nachsehen.

Untermenü *V_Eil*

Der Standardwert für die Eilgeschwindigkeit wird hier eingegeben.
Der Standardwert wird vom Treiber nach seinem Start oder einem Reset angenommen bis ein neuer Wert durch den entsprechenden Funktionsaufruf gesetzt wird. Wie man diese Geschwindigkeit berechnen kann, können Sie im Handbuch für die *isel*-Servosteuerkarte UPMV 4/12 nachlesen.

Untermenü *V_Referenz*

Die Anfangsgeschwindigkeit, mit der sich die Achse während der Referenzfahrt zum Referenzschalter bewegt, und die Endgeschwindigkeit, mit der sich die Achse während der Referenzfahrt aus dem Referenzschalter heraus bewegt, können Sie hier definieren. Um eine schnelle Referenzfahrt und eine hohe Wiederholgenauigkeit zu ermöglichen, muss die Anfangsgeschwindigkeit wesentlich größer als die Endgeschwindigkeit sein.

Untermenü *V Achsfaktor*

Der V-Verstärkungsfaktor ist der dynamische Kennwert einer Achse. Je größer dieser Wert ist, desto besser ist die Dynamik der Achse. Auf der einen Seite hängt die Achsdynamik von der Hardware und Mechanik ab auf der anderen Seite können Sie die Achsdynamik durch eine bewusste Änderung der Reglerparameter beeinflussen.
Der V-Verstärkungsfaktor ist experimentell mit Hilfe des Programms PAREIN.EXE zu ermitteln. Aufgrund der Anlagensymmetrie, die einen großen Einfluss auf die Bearbeitungsgenauigkeit hat, ist es wünschenswert, dass alle Anlagenachsen die gleichen V-Verstärkungsfaktoren haben.

Untermenü *V Reduktion*

Über das Feld Reduktionsfaktor geben Sie anlagenspezifische Faktoren ein, die mathematisch nicht fassbare Einflüsse wie z. B. Reibung, Spiel etc. berücksichtigen. Solche Einflüsse wirken sich negativ auf die Bahngenauigkeit aus.
Um diese negativen Einflüsse zu fassen, haben wir den Bahn- und den Umkehrreduktionsfaktor eingeführt. Über diese Faktoren können Sie die Übergangsgeschwindigkeit zwischen zwei aufeinander folgende Bewegungselemente variieren.
Beachten Sie, dass die Übergangsgeschwindigkeit nicht direkt proportional zu den Reduktionsfaktoren ist. Die Reduktionsfaktoren sind durch Probieren an der jeweiligen Anlage zu bestimmen.

Im Bahnbetrieb gilt: Je größer die Reduktionsfaktoren sind, desto kleiner ist die Geschwindigkeit beim Übergang zwischen zwei Segmenten. Das führt zu einer höheren Genauigkeit.
Der Unterschied zwischen dem Bahnreduktionsfaktor und dem Umkehrreduktionsfaktor besteht darin, dass der Bahnreduktionsfaktor bei jedem Segmentübergang aktiv ist. Der Umkehrreduktionsfaktor wirkt noch zusätzlich zu dem Bahnreduktionsfaktor, falls mindestens eine Achse ihre Bewegungsrichtung beim Segmentübergang wechselt.
Beim Kreisfahren gilt: Je größer der Reduktionsfaktor ist, desto kleiner ist die Bearbeitungsgeschwindigkeit.

Untermenü *Totzeit*
Jede Achse einer Anlage hat naturgemäß immer eine Totzeit ungleich Null. Die Totzeit stellt nichts anders als die Verzögerungszeit der Achsen dar. Intern benutzt der Treiber die Totzeiten der Achsen und die von Ihnen angegebene Toleranz, um den Nachlauffehler der Anlage während des Fahrens zu überwachen. Eine Toleranz von 200 % ist sinnvoll. Im Handbuch für die *isel*-Servosteuerkarte UPMV 4/12 können Sie nachsehen, wie der Nachlauffehler der Anlage intern im Treiber berechnet wird.

Untermenü *Endschalter*
Hier definieren Sie die Standardwerte der Software-Endschalter in Bezug auf den Referenzpunkt. Diese Standardwerte werden vom Treiber nach seinem Start oder nach einem Reset angenommen, obwohl die Software-Endschalter erst nach einem Aufruf der entsprechenden Funktion aktiv werden.
Die Änderung der Software-Endschalter während der Laufzeit des Treibers erfolgt extra durch eine andere Funktion.

Untermenü *Regler*
In diesem Menü definieren Sie die Parameter für die PID-Regler der Achscontroller. Das sind die Standardwerte, die der Treiber nach seinem Start oder einem Reset den Achscontrollern übergibt. Eine Änderung dieser Parameter ist durch die entsprechende Funktion möglich.
Hier können Sie außerdem noch festlegen, ob die Regler bei einem Nachlauffehler abgeschaltet werden sollen.

Diese Maßnahme ist sehr sinnvoll, falls Ihre Leistungsendstufen nicht die Möglichkeit haben, den Strom hardwaremäßig zu begrenzen. In solchen Fällen führt das Abschalten der Regler dazu, dass der Strom vom Treiber bei einem Nachlauffehler softwaremäßig begrenzt wird, d. h. ein Nachlauffehler, bei dem der Strom wegen der großen Regelabweichung über die zulässige Grenze übersteigen kann, stellt aufgrund dieser Maßnahme keine echte Gefahr dar. Aber nach wie vor ist die hardwaremäßige Strombegrenzung direkt in den Leistungsendstufen immer die beste Methode.

Weitere Informationen finden Sie im Handbuch für die *isel*-Servosteuerkarte UPMV 4/12.

5 Das Hauptmenü *Info*

Hier können Sie nichts eingeben, sondern nur sehen, was für eine Treiberversion Sie haben müssen, wenn Sie mit diesem Konfigurationsprogramm die Initialisierungsdateien für den Treiber erzeugen wollen. Außerdem können Sie aus diesem Menü unsere Anschrift entnehmen, die für Sie in manchen kritischen Situationen nützlich sein kann.

6 Mögliche Fehler bei der Parameterkonfiguration

Beim Erstellen der Initialisierungsdatei für den Treiber sollten Sie unbedingt unser Programm PARKON.EXE benutzen. Mit Hilfe dieses Programms können Sie auf der einen Seite sehr einfach und übersichtlich die Datei konfigurieren. Auf der anderen Seite kontrolliert das Programm so gut wie möglich Ihre Angaben auf Unstimmigkeiten. Fehlerhafte Angaben werden sofort zurückgemeldet.
In diesem Abschnitt wollen wir Ihnen die typischen Fehler und deren Ursachen auflisten.

Dateischreibfehler
Dieser Fehler tritt nur beim Speichern der Initialisierungsdatei auf. Der Grund dafür liegt bei dem Datenträger, der entweder defekt, voll oder schreibgeschützt ist.

Dateinamenfehler
Diesen Fehler können Sie beim Öffnen, beim Konvertieren oder beim Speichern einer Initialisierungsdatei bekommen. Die Namen der Initialisierungs-dateien sind Namen nach der DOS-Konvention. Alle Buchstaben von „a“ bis „z“ werden akzeptiert. Es gibt keine Unterscheidung zwischen Groß- und Kleinbuchstaben. Die Zahlen von „0“ bis „9“ sowie die Zeichen „-“ und „_“ werden als normale Buchstaben akzeptiert. Außerdem müssen die Dateinamen die Erweiterung „INI“ haben.

Dateiformatfehler
Sie bekommen diesen Fehler zu sehen, wenn Sie versuchen, eine Datei zu öffnen, die keine Initialisierungsdatei ist oder die eine falsche Versionsnummer hat. Das Gleiche kann auch beim Konvertieren passieren. Wir betonen ausdrücklich, dass Sie nicht versuchen sollten, die Initialisierungsdatei zu ändern, ohne das Programm PARKON.EXE dazu zu benutzen. Eine Änderung per Hand kann dazu führen, dass die geänderte Datei nicht mehr eingelesen werden kann.

Druckerfehler
Diesen Fehler bekommen Sie nur, wenn Sie versuchen, die Initialisierungsdatei zu drucken. Wenn der Drucker nicht bereits ist oder wenn die Verbindung zu dem Drucker nicht vorhanden ist, bekommen Sie logischerweise diesen Fehler.

Adressenkonflikt
Über den Untermenüs Überwachungsport, Ein/Ausgabeport und Bew_Ausgabeport können Sie verschiedene Ein- und Ausgabeports für den Treiber definieren. Bei jeder Eingabe werden alle Portadressen, sowohl die neu eingegebenen Adressen als auch die bereits über den anderen Untermenüs definierten Adressen, miteinander verglichen. Falls zwei unterschiedlichen Ports die selben Adressen haben, bekommen Sie in der Regel diese Fehlermeldung zu sehen. Es gibt hier aber eine Ausnahme. Ein Port, den Sie im Untermenü Überwachungsport
definiert haben, kann weiterhin im Untermenü Ein/Ausgabeport eingeben werden, wenn nicht alle Bits dieses Ports schon als benutzt markiert sind.
Die Adressen der Ausgabeports werden nicht nur miteinander sondern auch noch mit dem Adressenbereich unserer PC-Einsteckkarte verglichen.
Ein Ausgabeport in diesem Adressenbereich ist streng verboten. Hier haben Sie nichts zu suchen.
Die Adressen unserer PC-Einsteckkarte liegt im Bereich (Basisadresse + 00h) ... (Basisadresse + 0Fh). Die Basisadresse wird im Untermenü Basisadresse definiert.

Fehlerhafte Strukturangabe
Dieser Fehler tritt nur im Untermenü Struktur auf. Es hängt mit der Angabe der Anlagenstruktur zusammen. Eine XYZ_TTT-Struktur muss mindestens 3 Achsen haben und die X-, die Y- und die Z-Achse müssen Linearachsen sein. Eine XY_TTT-Struktur benötigt mindestens 2 Achsen, wobei die X- und die Y-Achse Linearachsen sein müssen. Bei einer X_TTT-Struktur haben Sie mindestens eine Achse und die X-Achse ist eine Linearachse. Alle anderen Anordnungen führen zwangsläufig zu einer KEIN_TTT-Struktur. Bei dieser Struktur müssen Sie beachten, dass die X-Achse eine Drehachse sein muss. Sonst haben Sie mindestens eine X_TTT-Struktur vor sich.

Fehlerhafte Referenzmode
Für eine Drehachse können Sie entweder den Referenzfahrt_Standard-Modus oder den Referenzfahrt_Nicht_Standard-Modus definieren. Aber bei einer Linearachse ist nur der Referenzfahrt_Standard-Modus zulässig. Falls Sie bei einer Linearachse unbedingt etwas anders probieren wollen, bekommen Sie diese Fehlermeldung als Antwort.

PAREIN - Einstellprogramm

1 Einleitung

Die Inbetriebnahme einer Servoanlage ist sehr kompliziert und gewiss nicht jedermanns Sache. Ehe sich der Aha-Effekt einstellt, müssen viele Schwierigkeiten überwunden werden.
Um Ihnen das Leben leichter zu machen, haben wir ein sogenanntes Inbetriebnahmeprogramm mit dem Namen PAREIN.EXE entwickelt.
Das Inbetriebnahmeprogramm soll Ihnen bei der Inbetriebnahme Ihrer Anlage mit unserer PC-Einsteckkarte UPMV 4/12 helfen.

Mit diesem Programm können Sie kontrollieren, ob der Encoder sowie dessen Verbindung zur PC-Einsteckkarte in Ordnung ist. Es unterstützt Sie beim Einstellen der Offset-Spannungen auf der PC-Einsteckkarte sowie auf der Endstufe. Weiterhin können Sie ohne große Mühe prüfen, ob die End- sowie Referenzschalter Ihrer Anlage funktionstüchtig und richtig konfiguriert sind. Eventuelle Fehler lassen sich schnell erkennen und aufheben. Achsübersetzungsfaktoren, deren Bestimmung bei manchen Anlagen sehr kompliziert ist, können mit unserem Programm sehr schnell ermittelt werden. Außerdem gibt das Inbetriebnahmeprogramm Ihnen entsprechende Hilfsmittel, um die PID-Positionsregler auf der PC-Einsteckkarte sowie die eventuell vorhandenen Geschwindigkeitsregler schnell und einfach zu dimensionieren.

Aus der Anwendersicht ist es natürlich wünschenswert, alle Schritte der Inbetriebnahme zu automatisieren. Dieses Ziel ist aufgrund der verschiedenen Hardwareplattformen kaum realisierbar.
Das Inbetriebnahmeprogramm gehört zu dem Softwarepaket, das wir den Kunden anbieten und bei uns im Haus benutzen, um Servomotoren zu steuern. Zu diesem Softwarepaket gehören außerdem noch der residente Softwaretreiber ISELDRV.EXE für das Betriebsystem MS DOS und das Konfigurationsprogramm PARKON.EXE.
Mit diesem Softwarepaket und der PC-Einsteckkarte UPMV 4/12 ist es möglich, derzeit bis zu 4 Servoachsen anzusteuern. Unter anderem sind die 4-Achsen-Linearinterpolation und die Kreis- sowie die Helixinterpolation die Hauptmerkmale unseres Servosteuerungssystems.

Das Inbetriebnahmeprogramm arbeitet selbständig und achsorientiert.
Sie brauchen vor dem Starten des Inbetriebnahmeprogramms den Treiber ISELDRV.EXE nicht zu installieren. Falls der Treiber schon resident im Speicher installiert ist, müssen Sie ihn entfernen, falls Sie unseres Inbetriebnahmeprogramm benutzen wollen. Auf diese Weise wollen wir die Konflikte der beiden Programme vermeiden.

Die Bedienung dieses Programms ist wegen des SAA-Standards (**S**ystem **A**pplication **A**rchitectur) und der On-Line-Hilfe sehr einfach. In den folgenden Abschnitten wollen wir genau erläutern, wie Sie die Menüs benutzen können

2 Das Hauptmenü *Datei*

In diesem Hauptmenü können Sie eine Initialisierungsdatei öffnen, speichern oder drucken. Das Inbetriebnahmeprogramm lässt sich in diesem Hauptmenü beenden.

2.1 Das Untermenü *Öffnen*

Der Treiber ISELDRV.EXE braucht bei seinem Start eine Initialisierungsdatei, die alle für ihn notwendigen Parameter erhält. Das Erstellen der Initialisierungsdatei ist einzig und allein die Aufgabe des Konfigurationsprogramms PARKON.EXE. Das Inbetriebnahmeprogramm kann zwar eine bereits vorhandene Initialisierungsdatei ändern, aber keine neue erstellen.
In diesem Menü können Sie eine Initialisierungsdatei öffnen. Dabei werden alle eingestellten Parameter dieser Datei vom Inbetriebnahmeprogramm übernommen. Es gibt einige Parameter, die unbedingt richtig sein müssen, wenn Sie mit dem Inbetriebnahmeprogramm arbeiten wollen. Das sind die Basisadresse der PC-Einsteckkarte, die Struktur Ihrer Anlage (Achsenanzahl, Linear oder Drehachse) und die Encoder-Linie (Anzahl der gelieferten Inkremente pro Motorumdrehung).
Alle diese Parameter wurden mit Hilfe des Konfigurationsprogramms in die Initialisierungsdatei eingegeben und entscheiden über den weiteren Programmablauf. Erst nach dem Öffnen einer Initialisierungsdatei werden die restlichen Untermenüs dieses Menüs und das Menü *Anlage* zugelassen.
Im Statusfenster im unteren Teil des Bildschirms können Sie immer sofort erkennen, an welcher Datei Sie momentan arbeiten.

2.2 Das Untermenü *Speichern*

Die momentan in Bearbeitung befindliche Initialisierungsdatei wird unter dem alten Name gespeichert.

2.3 Das Untermenü *Speichern als ...*

Die momentan in Bearbeitung befindliche Initialisierungsdatei wird unter einem neuen Name gespeichert. Sie müssen natürlich zuerst das Verzeichnis auswählen und den neuen Dateinamen angeben.
Ein Dateiname mit einer anderen Erweiterung als 'INI' wird nicht akzeptiert und ein Dateinamenfehler wird angezeigt. Nachdem die Datei erfolgreich gespeichert wurde, erscheinen der neue Dateiname mit dem Verzeichnis sowie das aktuelle Datum im Statusfenster.

2.4 Das Untermenü *Drucken*

Die momentan in Bearbeitung befindliche Datei wird gedruckt.
Auf der Druckliste mit eindeutigen Kommentaren und Parameternamen können Sie problemlos alle eingegebenen Daten kontrollieren. Ein eventuell vorhandener Druckfehler wird sofort angezeigt.

2.5 Das Untermenü *Beenden*

Hiermit beenden Sie das Inbetriebnahmeprogramm.
Das Inbetriebnahmeprogramm fordert Sie auf, die in Bearbeitung befindliche Datei zu speichern, falls diese Datei geändert wurde.

3 Das Hauptmenü *Anlage*

In diesem Hauptmenü können Sie die PC-Einsteckkarte und alle anderen Funktionen, die nicht direkt mit einer Achse verbunden sind, auf ihre Funktionstüchtigkeit überprüfen.

3.1 Das Untermenü *Einsteckkarte*

Wenn Sie dieses Untermenü auswählen, wird die Funktionstüchtigkeit der PC-Einsteckkarte geprüft. Dabei wird zuerst kontrolliert, ob die Basisadresse korrekt ist und ob der Timer sowie die Achscontroller in Ordnung sind. Es wird auch getestet, ob die von Ihnen gewählte Hardware-Interruptnummer (IRQ10 oder IRQ11) verfügbar ist. Falls die PC-Einsteckkarte nicht in Ordnung ist, wird der entsprechende Fehler zurückgemeldet (siehe Abschnitt 6). Nur bei einer funktionstüchtigen PC-Einsteckkarte werden die restlichen Untermenüs in dem Hauptmenü *Anlage* freigelassen.

3.2 Das Untermenü *Überwachung*

Der Kontrollbyte_Eingabeport, alle Angaben im Bezug auf den Hand-Modus sowie die Ein- und Ausgabeports für die Überwachung können in diesem Untermenü geprüft und entsprechend modifiziert werden. Im Bild 3.1 können Sie das Arbeitsmenü sehen.

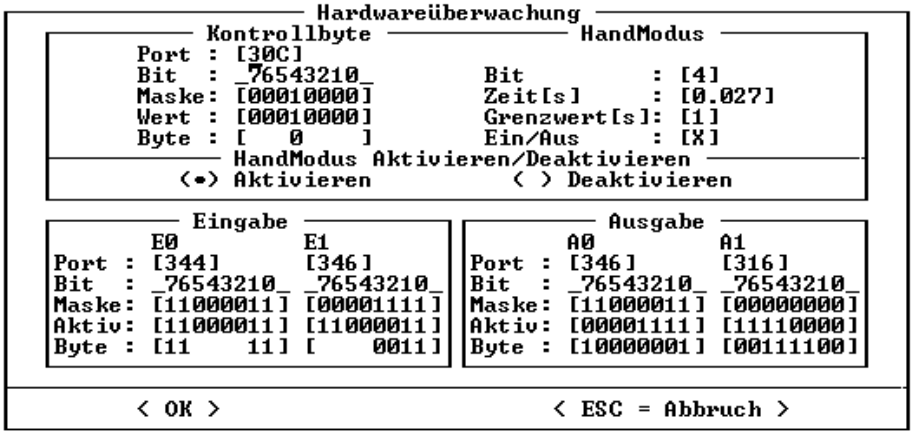


Bild 3.1: Arbeitsmenü für die Überwachungsports

Der Kontrollbyte_Eingabeport ist frei wählbar, d. h. Sie können jeden Eingabeport als den Kontrollbyte_Eingabeport definieren. Standardmäßig benutzen wir den Eingabeport mit der Adresse (Basisadresse + 0Ch) auf der PC-Einsteckkarte. Von diesem Port ist nur noch das Bit 4 frei. Wir benutzen dieses Bit in unserer Anlage, um den Strom des Servocontrollers zu überwachen. In diesem Fall muss die Nummer 4 ins Feld „Bit“ der Kategorie „HandModus“ eingetragen werden (siehe Handbuch PARKON.EXE). Über das Feld „Maske“ können Sie definieren, welche Bits des Kontrollbyte_Eingabeports Sie benutzen wollen. Ins Feld „Wert“ sind die fehlerfreien Bitwerte einzutragen. Der Inhalt des Kontrollbyte_Eingabeports wird ständig eingelesen und on-line bitweise im Feld „Byte“ dargestellt. Die nicht benutzten Bits werden hier aber nicht angezeigt.
Die Kategorie „HandModus“ und die Kategorie „HandModus Aktivieren/Deaktivieren“ sind eigentlich nur für *isel*-Anlagen gedacht. Beim Aktivieren des Hand-Modus schalten die Leistungsendstufen stromlos und beim Deaktivieren des Hand-Modus schalten Sie den Strom wieder ein. Beim Wechseln zwischen „Aktivieren“ und „Deaktivieren“ wird die Verzögerungszeit der Relais automatisch ermittelt und ins Feld „Zeit“ eingetragen.

Das Ermitteln der Verzögerungszeit ist eine ziemlich einfache Geschichte. Das Wechseln zwischen „Aktivieren“ und „Deaktivieren“ löst den Vorgang zur Zeitmessung aus. Dieser Messvorgang wird nur gestoppt, wenn sich der Zustand des Bits 4 des Kontrollbyte_Eingabeports ändert oder wenn die Messzeit den von Ihnen ins Feld „Grenzwert“ eingetragenen Maximalwert überschreitet. In dem zweiten Fall wird die Fehlermeldung

„Grenzwertzeit ist erreicht. Weiter mit beliebiger Taste“

auf dem Bildschirm gebracht. Dieser Fehler tritt dann auf, wenn Sie vergessen, den Servocontroller einzuschalten oder wenn die eingetragene Grenzwertzeit zu klein ist. Falls Sie den *isel*-Servocontroller nicht benutzen und die Besonderheiten beim Hand-Modus nicht gebrauchen können, müssen Sie die Benutzung über das Feld „Ein/Aus“ ausschalten.

Die Angabe sieht so aus:

[X] ---> Ein,
[ ] ---> Aus.

In der Kategorie „Eingabe“ können Sie jederzeit die Adressen der beiden Eingabeports für die Überwachung neu eingeben und die Benutzungsmaske sowie die Aktivwerte modifizieren. Die Portinhalte werden ständig eingelesen und mit den Aktivwerten kombiniert. Die logischen Werte der Portinhalte werden on-line angezeigt (siehe Handbuch PARKON.EXE). Als nicht benutzt definierte Bits werden nicht angezeigt.
Im Unterschied zu den Eingabeports können Sie die Adressen der Ausgabeports nicht ändern. Eine Änderung ist nur über das Programm PARKON.EXE möglich. Die Benutzungsmasken sowie die Aktivwerte der Ausgabeports können Sie ohne weiteres über das Feld „Maske“ und das Feld „Aktiv“ definieren. Über das Feld „Byte“ geben Sie die Werte ein, die sofort mit den Aktivwerten kombiniert und an den entsprechenden Ports ausgegeben werden. Auf Weise können Sie sehr einfach und schnell die Funktionalität der Ports kontrollieren.
Beachten Sie, dass Sie hier ähnlich wie bei den Eingabeports auch mit den logischen Werten arbeiten (siehe Handbuch PARKON.EXE).
Die korrekte Einstellung in diesem Untermenü spielt eine große Rolle bei der Inbetriebnahme der einzelnen Achsen. Deswegen wird das Hauptmenü *Achse* nur nach der Konfiguration in diesem Untermenü freigelassen.

3.3 Das Untermenü *Ref._Taste*

Dieses Untermenü unterstützt Sie bei der Bestimmung des Scan-Codes, der für die Unterbrechung der Referenzfahrt zuständigen Taste. Es geschieht hier alles automatisch. Sie brauchen nur die gewünschte Taste zu betätigen. Der Scan-Code wird für Sie ermittelt.
Beachten Sie, dass es verschiedene Typen von Tastaturen gibt. Unterschiedliche Tastaturtypen haben auch unterschiedliche Scan-Codes. Beim Wechseln der Tastatur ist der Scan-Code neu zu ermitteln.

3.4 Das Untermenü *Ein*/*Ausgabeport*

Der Treiber gibt den Anwendern die Möglichkeit, bis zu 4 Ein- und 4 Ausgabeports zu definieren. Das Ansprechen dieser Ports erfolgt nicht mehr über die physikalischen Adressen, sondern über die logischen Nummern. Parallel dazu besteht die Möglichkeit, eine geschwindigkeitsabhängige Ausgabe über einen 8-Bit-Port während der Bearbeitungsbewegung zu realisieren.
Alle diese Ein- und Ausgabeports können Sie hier in diesem Menü auf ihre Funktionstüchtigkeit überprüfen.
Das Bild 3.2 zeigt Ihnen das Arbeitsmenü für die Kontrolle von Ein- und Ausgabeports.

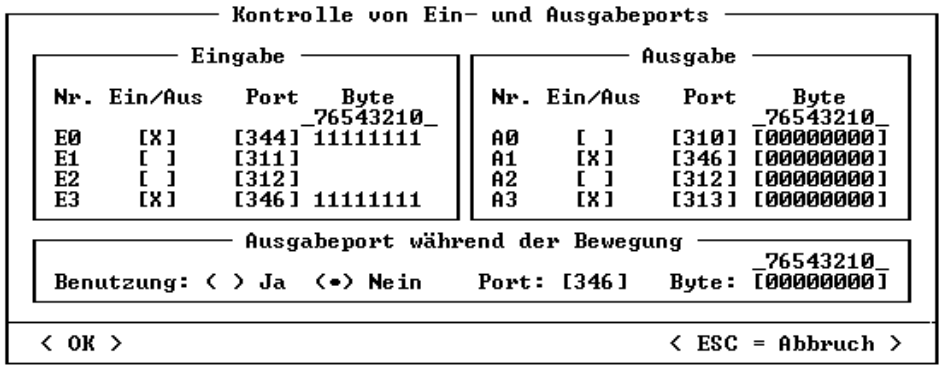


Bild 3.2: Arbeitsmenü für die Kontrolle von Ein- und Ausgabeports.

Zuerst müssen Sie angeben, ob Sie den jeweiligen Ein- und Ausgabeport benutzen wollen. Die Angabe sieht so aus:

Ein/Aus
[X] ---> Port wird benutzt,
[ ] ---> Port wird nicht benutzt.

Beim Port für die Ausgabe während der Bearbeitungsbewegung geschieht die Angabe durch den Klartext Ja oder Nein.
Alle physikalischen Adressen der Eingabeports können Sie jederzeit neu eingeben bzw. ändern.
Bei den Ausgabeports ist es anders. Hier können Sie die Adressen nicht ändern. Der Grund dafür ist ganz einfach. Um die Ausgabeports zu testen, werden die von Ihnen veränderbaren Werte im Feld „Byte“ ständig eingelesen und an den entsprechenden Ports ausgegeben. Falls Sie die Möglichkeit hätten, die Portadressen beliebig zu ändern, könnte es dazu führen, dass die Ausgabeoperationen zu den Portadressen erfolgen, die zu den anderen Hardwaren gehören. Es könnte sehr gefährlich werden. Deswegen haben wir hier die Änderung der Ausgabeportadressen nicht zugelassen.
Falls Sie die Adressen der Ausgabeports ändern wollen, müssen Sie das Konfigurationsprogramm PARKON.EXE benutzen.

Bei den als benutzt gekennzeichneten Eingabeports werden die an den jeweiligen Ports liegenden Werte ständig eingelesen und on-line angezeigt. Damit haben Sie die Möglichkeit, sehr schnell und einfach diese Eingabeports auf ihre Funktionstüchtigkeit zu kontrollieren.
Bei den Ausgabeports können Sie die Ausgabewerte im Eingabefeld *Byte* jederzeit ändern. Bei den als benutzt gekennzeichneten Ports werden die gerade eingegebenen Werte sofort an die jeweiligen Ports ausgegeben. Somit können Sie sehr einfach die jeweiligen Ausgabeports auf ihre Funktionalität kontrollieren.

Im Unterschied zu den Überwachungsports, die Sie im Untermenüs *Überwachung* prüfen können, arbeiten Sie hier nicht mit den logischen sondern mit den tatsächlichen Portinhalten, d. h. die Werte, die von den Eingabeports eingelesen und on-line angezeigt werden, sind tatsächliche Inhalte der jeweiligen Ports und bei den Ausgabeports werden die von Ihnen eingegebenen Werte ohne jede Änderung direkt an den jeweiligen Port ausgegeben.

4 Das Hauptmenü *Achse*

Die Hauptarbeit für die Inbetriebnahme einer Achse findet in diesem Menü statt. Dieses Hauptmenü wird nur dann zugelassen, nachdem Sie das Untermenü „Überwachung“ im Hauptmenü „Anlage“ ordentlich bearbeitet haben.
Sie können in diesem Hauptmenü den Offset der Einsteckkarte sowie den der Leistungsendstufe einstellen. Der Encoder sowie die End- und Referenzschalter lassen sich auf ihre Funktionstüchtigkeit prüfen. Andere Optionen, wie z. B. Überbrückung der Endlageschalter, Enable/Disable der Leistungsendstufen, usw. lassen sich hier auch prüfen.
In diesem Menü bekommen Sie außerdem noch die entsprechenden Hilfsmittel, um den Achsübertragungsfaktor, die Parameter für den optionalen Geschwin-digkeitsregler und den PID-Positionsregler sowie die Bewegungsrampen zu bestimmen.

Wir betonen, dass das Dimensionieren der Regler und das Ermitteln der Bewegungsrampen unter den realen Bedingungen stattfinden sollte. Wenn die Achse z. B. während der Bearbeitung mit bis zu 50 kg belastet werden soll, sollten Sie die Achse während des Inbetriebnahmevorgangs auch mit 50 kg belasten. Es gibt zwar immer einen gewissen Toleranzbereich (der Fachbegriff dafür: Robustheit), aber je praxisnäher die Bedingungen während des Inbetriebnahmevorgangs sind, desto besser ist die Reglungsqualität hinterher.
Sie sollten bedenken, dass der Servomotor zerstört werden kann, falls der Motorstrom über längere Zeit den vom Motorhersteller angegebenen Grenzwert übersteigt. Wegen der Regelung kann der Motorstrom aber sehr schnell seinen Grenzwert übersteigen, wenn z. B. die Achse auf einen Anschlag gefahren ist und nicht rechtzeitig zurückgefahren wird.
Aus diesem Grund sind wir hier so verfahren, dass der Controllerchip LM628 der Achse generell 2 Sekunden nach jeder Teach-In-Bewegung mit einem von uns festgelegten Reglerparametersatz eingeladen wird, der den Motorstrom begrenzt. Nachdem der neue Reglerparametersatz aktiv ist, werden Sie feststellen, dass die Achse keine Kraft mehr hat. Es ist aber kein Grund zur Sorge. Sie sollten auf Ihrer Seite trotzdem niemals eine Achse bis zum Anschlag fahren und dort stehen lassen. So schnell wie möglich sollten Sie die Achse aus dem Anschlag herausfahren.

Aufgrund der Anlagensicherheit wird der Strom bei *isel*-Servocontrollern generell ausgeschaltet, falls einer der Endlageschalter aktiv ist. In diesem Fall müssen Sie bei der Treiberversion 3.0 den Schlüsselschalter benutzen, um den Strom wieder einschalten zu können. Erst wenn der Strom da ist, können Sie die Achse weiterfahren. Bei der Treiberversion 3.1 gibt es die Option Schlüsselschalter nicht mehr. Um die Endlageschalter zu überbrücken, muss das Bit 5 des Ausgabeports (Basisadresse+0Ch) gesetzt werden. Das Setzen dieses Bits können Sie sehr einfach machen, weil wir Ihnen in jedem Arbeitsmenü die entsprechende Möglichkeit einräumen.
Das Überbrücken der Endlageschalter stellt aber eine gewisse Gefahr dar. Deswegen wird Sie die blinkende Warnung „HWE-Überbrücken“ in der unteren rechten Ecke des Bildschirms daran erinnern, dass die Endlagenschalter überbrückt sind. Das Stromabschalten und das Stromeinschalten bei aktiven Endlageschaltern ist ein Merkmal der *isel*-Servocontroller.
Wenn Sie unsere PC-Einsteckkarte mit Servocontrollern anderer Hersteller benutzen, brauchen Sie sich wahrscheinlich gar nicht darum zu kümmern. Das Gesagte hat keine Bedeutung für Sie.

4.1 Das Untermenü *Auswahl*

Hier haben Sie die Möglichkeit, eine der Achsen auszuwählen, die Sie in Betrieb nehmen wollen. Nach der Auswahl der wird das Untermenü *Reihenfolge* freigegeben.

4.2 Das Untermenü *Reihenfolge*

Hier entscheiden Sie über die Arbeitsweise des Hauptmenüs *Achse*. In der Betriebsart *Nacheinander* werden die Untermenüs nacheinander von oben nach unten freigegeben. Ein Untermenü wird nur dann aktiv, wenn die Arbeit im vorangegangenen Untermenü erfolgreich abgeschlossen werden konnte. Diese Betriebsart ist dafür gedacht, wenn Sie die Achse zum erstenmal in Betrieb nehmen wollen.
Die Anlagenachse stellt ein Antriebssystem aus verschiedenen Teilen dar. Das Funktionieren eines Systemteils ist die notwendige Voraussetzung für die Inbetriebnahme eines anderen Systemteils. Sie können z. B. keinen Regler dimensionieren, falls der Encoder noch nicht läuft. Mit der Betriebsart *Nacheinander* wollen wir verhindern, dass Sie schwerwiegende Fehler während der Inbetriebnahme machen.

In der Betriebsart *Beliebig* werden alle Untermenüs auf einmal freigegeben. Sie können jederzeit ein beliebiges Untermenü benutzen. Diese Betriebsart ist für Nachbesserungen bei einer Achse gedacht. Sie ist z. B. für solche Fälle sehr nützlich, in denen Sie Änderungen an einer bereits in Betrieb genommenen Achse vornehmen wollen. Sie können dann in der Betriebsart *Beliebig* unnötige Schritte überspringen und sofort zu dem gewünschten Untermenü gelangen, um die Nachbesserungen durchzuführen.

4.3 Das Untermenü *Achsfixieren*

Während der Inbetriebnahme einer Achse stehen alle anderen Achsen normalerweise im stromlosen Zustand. In manchen Anlagenkonfigurationen kann es zu einer verlagerten Bewegung der Achsen während des starken Beschleunigens und Bremsens der gewählten Achse führen.
Mit Hilfe dieses Menüs können Sie eine oder alle nicht gewählten Achsen auf bestimmten Positionen während der Inbetriebnahme der gewählten Achse fixieren.

4.4 Das Untermenü *V_Regler*

Dieses Untermenü wird nur freigegebenen, wenn Ihre Anlage Geschwindigkeitsregler hat (siehe Handbuch PARKON.EXE). Die Benutzung eines Geschwindigkeitsreglers ist zwar optional, aber Sie sollten sich bewusst sein, dass Sie mit einem Geschwindigkeitsregler, der in hochwertigen Antriebssystemen normalerweise immer vorhanden ist, eine wesentlich höhere Dynamik, einen besseren Gleichlauf, eine größere Robustheit, usw. erreichen können.
In den meisten Fällen ist der Geschwindigkeitsregler direkt in der Leistungsendstufe integriert. Deswegen müssen Sie genau den Hinweisen des jeweiligen Herstellers folgen, um den Regler einzustellen. Hier bieten wir Ihnen aber die entsprechenden Hilfsmittel, damit Sie beurteilen können, ob der Regler optimal eingestellt ist. Die Grundlage dafür ist das Bewerten der Sprungantwort (siehe Abschnitt 4.7.2).

Nach der Auswahl dieses Menüs haben Sie die Möglichkeit, den Offset der Leistungsendstufe einzustellen. Vom Achscontroller (Chip LM628) wird hier 0 Volt ausgegeben. Sie sollten zuerst den Offset auf der PC-Einsteckkarte einstellen. Zu diesen Zweck stehen auf der PC-Einsteckkarte für jede Achse entsprechende Potentiometer zur Verfügung (siehe Handbuch *isel*-Servo-steuerkarte UPMV 4/12).
Durch das Messen der Spannung am jeweiligen Ausgang können Sie feststellen, ob der Offset gut eingestellt ist. Die Offset-Spannung auf der PC-Einsteckkarte sollte 0 Volt betragen. Im Allgemeinen können Sie davon ausgehen, dass wir den Offset der jeweiligen Achse bereits ab Werk richtig eingestellt haben.
Nachdem Sie mit dem Offset der PC-Einsteckkarte fertig sind, können Sie zum Offset der Leistungsendstufe kommen. Wie Sie hier den Offset einstellen können, müssen Sie im Handbuch des jeweiligen Herstellers nachlesen.
Aber in den meisten Fällen wird der Offset der Leistungsendstufe auch wieder durch das Drehen von entsprechenden Potentiometern eingestellt.

Um die Sprungantwort bewerten zu können, simuliert das Inbetriebnahme-programm für Sie einen einfachen Oszillographen auf dem Bildschirm (siehe Bild 4.1). Der PID-Positionsregler im Chip LM628 wird ausgeschaltet.

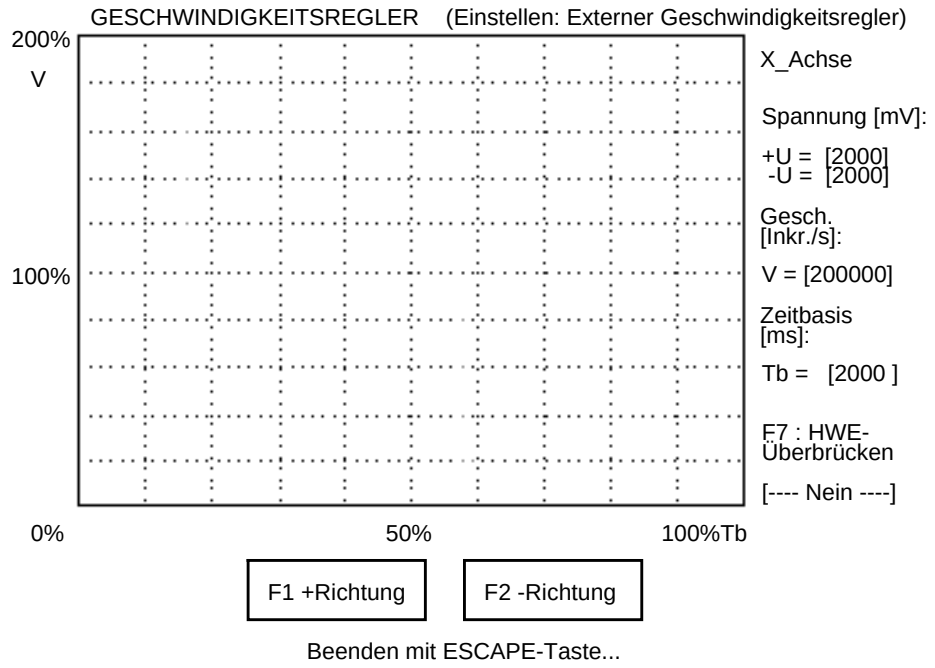


Bild 4.1: Das Arbeitsmenü beim Einstellen des Geschwindigkeitsreglers

Wenn Sie die Funktionstaste F1 betätigen und niederhalten, erzeugt das Inbetriebnahmeprogramm für Sie eine Spannungssprungfunktion mit der positiven Sprunghöhe, die Sie über die Variable +U auf der linken Seite des Bildschirms eingeben können (über das Bedienen der Eingabefelder können Sie im Abschnitt 3.7.2 nachlesen). Wenn Sie die Funktionstaste F2 betätigen und niederhalten, bekommen Sie eine Spannungssprungfunktion mit der negativen Sprunghöhe, die Sie über die Variable -U definiert haben. Beachten Sie, dass die Einheit der Spannungswerte von +U und -U [mV] ist. Die so erzeugte Spannungssprungfunktion wird an dem jeweiligen Ausgang der gewählten Achse ausgegeben. Beim Loslassen der jeweiligen Taste wird die Spannung sofort auf 0 zurückgesetzt. Diese Spannungssprungfunktion führt zu einer Sprungantwort der Achse. Die Sprungantwort der Achse ist nichts anders als der Ist-Geschwindigkeitsverlauf, der on-line auf dem Bildschirm dargestellt wird.
Über die Variablen V und Tb auf der linken Seite des Bildschirms können Sie den Maßstab der graphischen Darstellung festlegen. Mit der Taste F7 können Sie bei einem *isel*-Servocontroller jederzeit die Endlageschalter überbrücken, um den Controller bei aktiven Endlageschaltern einschalten zu können. Nachdem Sie diese Taste betätigt haben, erscheint das Wort „Ja“ statt des Wortes „Nein“. Gleichzeitig gibt es eine blinkende Warnung „HWE-Überbrückung“ auf der unteren rechten Ecke des Bildschirms.

Anhand der gezeichneten Kurve können Sie dann feststellen, ob der Geschwindigkeitsregler schon optimal eingestellt ist oder nicht. Da man mit dem Geschwindigkeitsregler eine größere Dynamik erreichen will und weil der Geschwindigkeitsregler die hochfrequenten Störungen ausregeln soll, sollten

Sie ihn so einstellen, dass die Sprungantwort eine Überschwinghöhe von etwa 60 % der Sprunghöhe hat (siehe Bild 4.2).

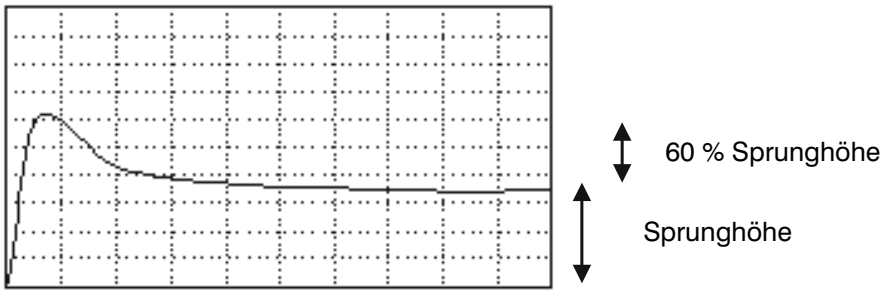


Bild 4.2: Sprungantwort der Achse mit dem optimal eingestellten Geschwindigkeitsregler

Mit der ESC-Taste können Sie jederzeit das Arbeitsmenü zum Einstellen des Geschwindigkeitsreglers beenden.

4.5 Das Untermenü *Offset*

Vom Achscontroller (Chip LM628) wird hier 0 Volt ausgegeben. Sie sollten zuerst den Offset auf der PC-Einsteckkarte einstellen (wenn Sie diesen Offset bereits im Untermenü *V_Regler* eingestellt haben, brauchen Sie es hier nicht noch einmal zu machen). Zu diesem Zweck stellt die PC-Einsteckkarte für jede Achse entsprechende Potentiometer zur Verfügung (siehe Handbuch *isel*-Servosteuerkarte UPMV 4/12).
Durch das Messen der Spannung an den jeweiligen Ausgängen können Sie feststellen, ob der Offset gut eingestellt ist. Die Offset-Spannung auf der PC-Einsteckkarte sollte 0 Volt betragen. Im Allgemeinen können Sie davon ausgehen, dass der Offset jeder Achse auf der PC-Einsteckkarte bereits ab Werk richtig eingestellt ist.
Nachdem Sie mit dem Einstellen des Offset der PC-Einsteckkarte fertig sind, können Sie zum Offset der Leistungsendstufe kommen. Wie Sie hier den Offset einstellen können, müssen Sie im Handbuch des jeweiligen Herstellers nachlesen. Aber in den meisten Fällen wird der Offset der Leistungsendstufe auch hier wieder durch das Drehen von entsprechenden Potentiometern eingestellt. Hier sollte der Offset so eingestellt sein, dass die Achse auf jeder Position beharren kann und dass Sie mit der Hand ohne Mühe die Achse in beide Richtungen bewegen können.

4.6 Das Untermenü *Encoder*

Die Funktionstüchtigkeit des Encoders ist eine unbedingt notwendige Voraussetzung für das einwandfreie Funktionieren der Achse in einem Regelkreis.
In diesem Menü bieten wir Ihnen die Möglichkeit, zu kontrollieren, ob die Verbindung zwischen der PC-Einsteckkarte und dem Encoder vorhanden ist und ob die Encoder-Signalleitungen richtig angeschlossen sind.

Beachten Sie, dass das Inbetriebnahmeprogramm nicht in der Lage ist, Störungen auf den Encoder-Signalleitungen aufzuspüren und anzuzeigen, weil solche Störungen und deren Beseitigung eine höchst komplizierte Angelegenheiten sind. Sie sollten aber besonders auf die Abschirmung der Signalleitungen, die verschiedenen Erdschleifen, das Bezugspotential sowie die Signalentkopplungen achten.
Um auf das Vorhandensein der Verbindung zwischen der PC-Einsteckkarte und dem Encoders zu prüfen, müssen Sie die Achse mit der Hand hin und her bewegen. Anhand der Ist-Positionsanzeige auf dem Bildschirm wissen Sie sofort, ob die Verbindung da ist. Wenn sich die Ist-Positionsanzeige nicht ändert, ist die Verbindung offensichtlich nicht vorhanden. In diesem Fall bleibt Ihnen nichts anderes übrig, als das Inbetriebnahmeprogramm zu beenden und den Hardwarefehler zu beheben.

Eine vorhandene Verbindung zwischen der PC-Einsteckkarte und dem Encoder bedeutet längst noch nicht, dass die Welt schon in Ordnung ist.
In diesem Fall kann es ja noch passieren, dass die Encoder-Signalleitungen verkehrt angeschlossen sind. Sie haben z. B. die Encoder-Signalleitung A bzw. A_Negation mit dem Signaleingang B bzw. B_Negation des Achscontrollers und die Encoder-Signalleitung B bzw. B_Negation mit dem Signaleingang A bzw. A_Negation des Achscontrollers verbunden. Bei solchen Hardwarefehlern kann es beim Einschalten der Leistungsendstufe passieren, dass die Achse mit voller Kraft gegen einen Anschlag fährt. Mechanische Schäden sind vorprogrammiert.
Aus diesem Grund bieten wir Ihnen hier eine Testmöglichkeit an, um diesen Fehler abzufangen. Bei diesem Test müssen Sie zuerst eine Bewegungslänge definieren. Eine Bewegungslänge von 0 Inkrement wird intern automatisch in eine Bewegungslänge von +1 Inkrement umgewandelt. Nachdem die Bewegungslänge bestätigt wird, beginnt der Test. Wenn der Test fertig ist, werden die Soll- und die Ist-Position angezeigt. Wenn die Vorzeichen der Soll- und der Ist-Position unterschiedlich sind, liegt ein Verbindungsfehler vor.
In diesem Fall müssen Sie das Inbetriebnahmeprogramm beenden und den Hardwarefehler beheben. Eine Ist-Position von 0 oder einigen Inkrementen deutet in den meisten Fällen darauf hin, dass die Achse momentan auf einem Anschlag steht oder dass Sie vergessen haben, die Leistungsendstufe einzuschalten.
Es kann natürlich auch sein, dass die Leistungsendstufe defekt oder dass die Verbindung zwischen der PC-Einsteckkarte und der Leistungsendstufe nicht in Ordnung ist. Sie sollten den Hardwarefehler beheben und den Test neu starten. Wenn die Soll- und Ist-Position das gleiche Vorzeichen haben, ist die Verbindung zwischen der PC-Einsteckkarte und dem Encoder in Ordnung. Trotzdem sollten Sie den Test ein paar Mal sowohl mit positiven als auch mit negativen Bewegungslängen durchführen.

4.7 Das Untermenü *Posi._Regler*

4.7.1 Allgemeine Bemerkungen

Servomotoren können nur in geschlossenen Regelkreisen betrieben werden. Die Qualität eines Servo-Antriebssystems hängt sehr stark von den benutzten Reglerstrukturen und von der Dimensionierung der Reglerparameter ab.
Es gibt verschiedene Reglerstrukturen, wie z. B. Mehrgrößenregler, Adaptivregler, Zustandsregler, konventionelle Regler u. a. m., es ist eine Liste fast ohne Ende. Abgesehen von einigen Strukturen, die in der Praxis nie richtig funktioniert haben, wie z. B. Dead-Beat-Regler, hat jede Reglerstruktur ihr eigenes Anwendungsfeld in der Verfahrenstechnik, in der Antriebstechnik, in der Kraftfahrzeugtechnik, usw. In der Antriebstechnik ist der konventionelle PID-Regler mit verschiedenen Modifikationen zu Hause. Weil sich diese konventionelle Reglerstruktur schon seit langem bewährt hat, bieten wir Ihnen für jede Achse einen Achscontroller (Chip LM628) an, der einen PID-Regler zur Verfügung stellt. Sie brauchen sich also keine Gedanken darüber zu machen, welche Reglerstruktur Sie nehmen sollten. Diese ist schon fest definiert. Sie haben "nur" noch das Problem, herauszufinden, wie die Parameter dieses Reglers dimensioniert werden müssen.

Die Dimensionierung der Reglerparameter ist keine einfache Geschichte.
Es gibt verschiedene Dimensionierungsmethoden, wie z. B. das Symmetrieoptimum-Verfahren, das Betragoptimum-Verfahren, das Wurzelortverfahren, das Amplituden- und Phasenrandverfahren, usw. Solche Dimensionierungsmethoden haben zwar einen hohen theoretischen Wert, sind aber in der Praxis der Antriebstechnik kaum einsetzbar, weil sie vom Anwender viel Erfahrungen und tiefgründige Kenntnisse in der Regelungstechnik voraussetzen. Außerdem muss das Antriebssystem in seinem Amplituden- und Phasengang bekannt sein, um diese Methoden zu benutzen. Theoretisch besteht zwar die Möglichkeit, ein Antriebssystem in seinem Amplituden- und Phasenrand sowie in seiner Struktur und deren Parameter programmmäßig mit entsprechenden Identifikationsverfahren zu ermitteln, aber aufgrund der mathematisch nie erfassbaren Störungen sowie aufgrund der bei einem realen Antriebssystem immer existierenden Totzeit und auch wegen der Vielfalt der in der Praxis eingesetzten Antriebsyssteme ist es nur schwer möglich, mit einem Identifikationsprogramm zu brauchbaren Ergebnissen zu gelangen.
Aus diesem Grund haben wir für Sie ein experimentelles Einstellverfahren entwickelt.

Mit diesem Verfahren sind Sie in der Lage, sehr schnell die notwendigen Parameter für den PID-Regler zu bestimmen. Auf eventuelle Fragen, wie, warum und weshalb können wir hier leider nicht eingehen, weil es den Rahmen dieser Beschreibung sprengen würde.

4.7.2 Das Arbeitsmenü bei der Dimensionierung der PID-Regler

Das Grundgedanke bei unserem Einstellverfahren zur Dimensionierung der PID-Regler ist die Bewertung der Sprungantwort der Ist-Geschwindigkeit. Die Reglerparameter sind Kp, Ki, Kd, Td und Il (siehe Handbuch *isel*-Servo-steuerkarte UPMV 4/12). Ihre Aufgabe besteht darin, diese Parameter so einzustellen, dass die Sprungantwort der Ist-Geschwindigkeit auf dieser Achse optimal ist. Um diese Arbeit zu ermöglichen, simuliert das Inbetriebnahmeprogramm für Sie einen einfachen Oszillographen mit den entsprechenden Eingabemöglichkeiten auf dem Bildschirm (siehe Bild 4.3).

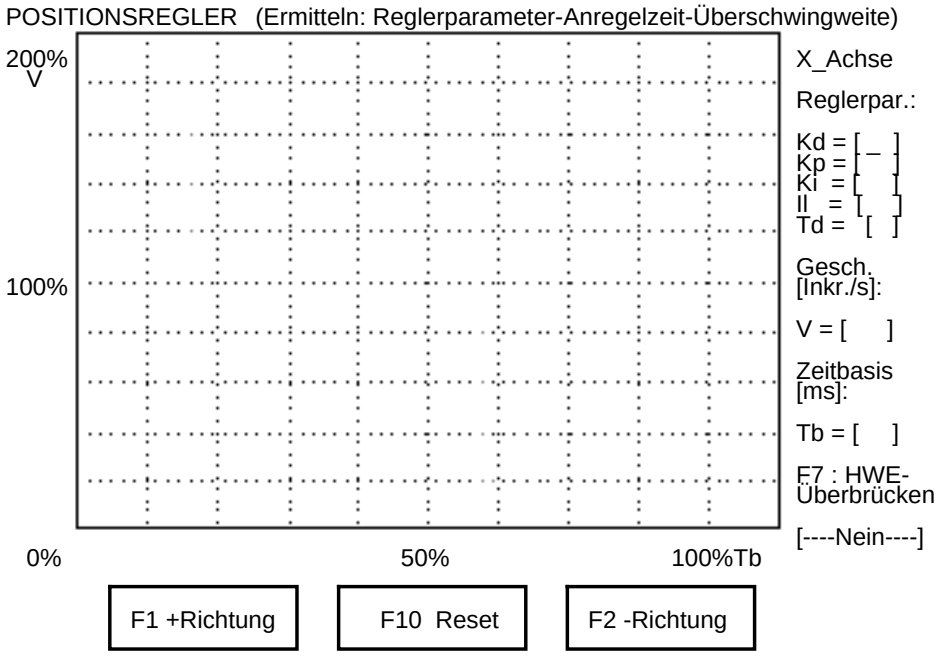


Bild 4.3: Das Arbeitsmenü bei der Dimensionierung des PID-Reglers

Auf der linken Seite haben Sie die Eingabefelder für die Reglerparameter Kd, Kp, Ki, Il, Td. Die Anfangswerte in diesen Eingabefeldern werden von der geöffneten Initialisierungsdatei übernommen.

Über die Variable V können Sie die Soll-Geschwindigkeit definieren, mit der sich die Achse bei der Teach-In-Bewegung mit den Funktionstasten F1 und F2 bewegen soll. Die Einheit von V ist Inkrement/Sekunde. Die Soll-Geschwindigkeit V ist der Maßstab der Ordinatenachse. Auf dieser Achse wird eine Geschwindigkeit im Bereich 0 % ... 200 % von V dargestellt.
Die Variable Tb ist der Zeitmaßstab auf der Abszissenachse. Die Einheit von Tb ist Millisekunden. Auf dieser Achse wird eine Zeit im Bereich 0 % ... 100 % von Tb dargestellt.

Über die beiden Cursortasten CURSOR_UP und CURSOR_DOWN können Sie zwischen den Eingabefeldern wechseln. Welches Eingabefeld momentan aktiv ist, können Sie an dem kleinen Cursor erkennen. In jedem Eingabefeld können

Sie eine beliebige Zahl ohne Vorzeichen eingeben, solange diese Zahl den jeweiligen Rahmen nicht überschreitet. Mit den Cursortasten CURSOR_LEFT oder BACKSPACE können Sie die Eingabe löschen. Mit dem Betätigen einer der Funktionstasten F1 ... F10, der Taste ENTER oder der Cursortasten CURSOR_UP und CURSOR_DOWN werden die eingegebenen Werte übernommen.
Durch das Niederhalten einer der beiden Funktionstasten F1, F2 können Sie veranlassen, dass die Achse eine Teach-In-Bewegung mit den zuletzt eingegebenen Reglerparametern und mit der zuletzt eingegebenen Soll-Geschwindigkeit V ausführt. Beim Loslassen der Taste wird die Bewegung sofort unterbrochen. Diese Sicherheitsmaßnahme ist notwendig, um mechanische Schäden durch eine falsche Bedienung zu vermeiden.

Während der Teach-In-Bewegung wird der Verlauf der Ist-Geschwindigkeit auf dem Bildschirm gezeichnet. Die gezeichnete Kurve ist die Sprungantwort der Ist-Geschwindigkeit mit den zuletzt eingegeben Reglerparametern. Sie dient als die Grundlage für die Bewertung der Reglerqualität.
Um die Einflüsse der Umkehrspanne zu vermeiden, sollten Sie die entsprechende Taste F1 bzw. F2 kurz betätigen, bevor der Ist-Geschwindigkeitsverlauf richtig aufgenommen werden sollte.

Bei einer ungünstigen Einstellung der Reglerparameter kann es während der Inbetriebnahme passieren, dass die Achse zu stark schwingt. Dies kann dann zu mechanischen Schäden führen. Aus diesem Grund haben wir die Reset-Funktion mit der Funktionstaste F10 eingeführt. Beim Betätigen dieser Taste wird der PID-Regler mit einem intern definierten Satz von Werten geladen. Der PID-Regler mit diesen Parametern garantiert eine schwingungsfreie Achse.

Mit der Taste F7 können Sie bei einem *isel*-Servocontroller jederzeit die Endlageschalter überbrücken, um den Controller bei aktiven Endlageschaltern einschalten zu können. Nachdem Sie diese Taste betätigt haben, erscheint das Wort „Ja“ statt des Wortes „Nein“. Gleichzeitig gibt es eine blinkende Warnung „HWE-Überbrückung“ auf der unteren rechten Ecke des Bildschirms.

Mit der ESC-Taste beenden Sie den Einstellvorgang. Das Inbetriebnahmeprogramm verlangt die Eingabe der Überschwingweite und der Anregelzeit, um den Geschwindigkeitsfaktor und den Dämpfungsgrad der Achse bei dem gerade ermittelten Reglerparametersatz zu berechnen. Die Reglerparameter, der Geschwindigkeitsfaktor und der Dämpfungsgrad werden angezeigt.
Wenn Sie diese Werte akzeptieren, werden die ermittelten Parameter mit Ausnahme des Dämpfungsgrades automatisch in die Initialisierungsdatei übernommen. Der Dämpfungsgrad ist nur ein Hilfswert, der vom Treiber nicht benötigt und auch nicht in der Initialisierungsdatei übernommen wird. Anhand dieses Dämpfungsgrades und dem ermittelten Reglerparametersatz können Sie das Schwingungsverhalten der Achse bewerten.

4.7.3 Die Dimensionierung der PID-Regler

Für den Erfolg des Einstellverfahrens ist es sehr wichtig, dass Sie die im Folgenden beschriebenen Schritte genau in der angegebenen Reihenfolge verfolgen.

Schritt 1: Definieren der Soll-Geschwindigkeit V

Die Soll-Geschwindigkeit V sollten Sie während des Einstellens der Reglerparameter auf mindestens 10 000 Inkremente/Sekunde setzen. Nach dem fertigen Einstellen dieser Parameter können Sie die Soll-Geschwindigkeit noch variieren, um feiner einstellen zu können.

Schritt 2: Definieren der Zeitbasis Tb

Die Zeitbasis Tb ist der Maßstab für die graphische Darstellung der Ist-Geschwindigkeit. Bei einer kleinen Zeitbasis können Sie den Übergang der Ist-Geschwindigkeit von 0 bis zur Soll-Geschwindigkeit besser beobachten. Eine große Zeitbasis eignet sich für das Beobachten des Langzeitverhaltens der Ist-Geschwindigkeit. Am Anfang können Sie den Standardwert Tb = 1 000 ms nehmen. Während des Einstellvorgangs können Sie Tb jederzeit ändern, um sie auf Ihre Bedürfnisse anzupassen.

Schritt 3: Einstellen von Kd

Sie müssen Ki und Td gleich 0 setzen. Kp sollte im Bereich 2 ... 20 sein. In diesem Schritt bleiben die Parameter Kp, Ki, Td, Il während des Einstellvorgangs für Kd unverändert.
Kd muss schrittweise vergrößert werden. Der Anfangswert für Kd kann gleich 50 sein. Ein Vergrößerungsschritt von 50 ist angemessen. Nach jeder neuen Eingabe von Kd müssen Sie die Achse mit Hilfe der Funktionstasten F1 oder F2 in Bewegung setzen. Auf dem Bildschirm erscheint der Ist-Geschwindigkeitsverlauf. Im Bild 4.4 können Sie einen typischen Ist-Geschwindigkeitsverlauf während dieser Phase sehen.

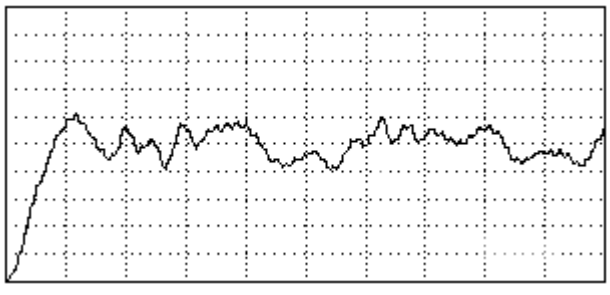


Bild 4.4: Typischer Ist-Geschwindigkeitsverlauf während des Einstellvorgangs für Kd

Während des Einstellens von Kd spielt der Ist-Geschwindigkeitsverlauf nur eine untergeordnete Rolle. Viel wichtiger für Sie ist das Geräusch, das Sie während der mit F1 oder F2 durchgeführten Achsbewegung hören können. Sie müssen Kd schrittweise erhöhen, bis Sie das Geräusch während der Bewegung deutlich hören können. Je größer Kd ist, desto besser. Dann müssen Sie Kd schrittweise verkleinern, bis das Bewegungsgeräusch verschwindet. Somit ist der Parameter Kd fertig eingestellt.
Ab jetzt können Sie die Bewegungsgeschwindigkeit V noch variieren, um Kd feiner einzustellen. Im Bild 4.5 können Sie den typischen Ist-Geschwindigkeitsverlauf nach dem Einstellen von Kd sehen.

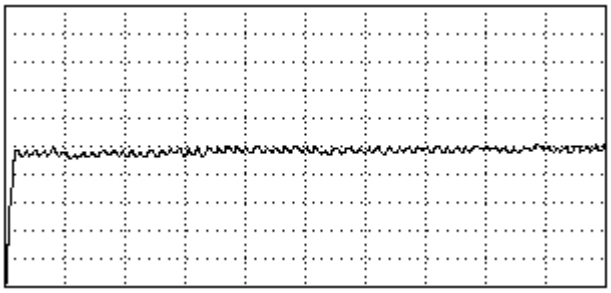


Bild 4.5: Typischer Ist-Geschwindigkeitsverlauf nach dem Einstellen von Kd

Schritt 4: Einstellen von Kp

Sie müssen Ki und Td gleich 0 setzen. Kd hat den Wert, den Sie im Schritt 3 eingestellt haben. Sie müssen Kp schrittweise erhöhen. Der Anfangswert von Kp sollte im Bereich 2 ... 10 sein. Ein Vergrößerungsschritt von 5 ist angemessen. Nach jeder neuen Eingabe von Kp sollten Sie mit F1 oder F2 die Achse bewegen, um den Ist-Geschwindigkeitsverlauf aufzunehmen. Je größer Kp ist, desto besser. Aber ein zu großer Wert von Kp führt zu folgenden Erscheinungen:

- Während der Bewegung schwingt die Ist-Geschwindigkeit um den Sollwert (siehe Bild 4.6).
- Die Schwingung der Ist-Geschwindigkeit um den Sollwert verursacht ein Bewegungsgeräusch, das Sie während der Achsbewegung deutlich hören können.
- Während des Ruhezustands neigt die Achse zu kleinen Schwingungen um ihre aktuelle Position. Das können Sie sehr einfach feststellen, in dem Sie die Achse leicht stoßen. Bei einem zu großen Wert von Kp fängt die Achse an, um ihre Position zu schwingen. Diese kleinen Dauerschwingungen verursachen wiederum ein deutliches Schwingungsgeräusch, das Sie sehr gut hören können.

Aufgrund dieser Erscheinungen können Sie sehr schnell feststellen, ob Kp zu groß ist.

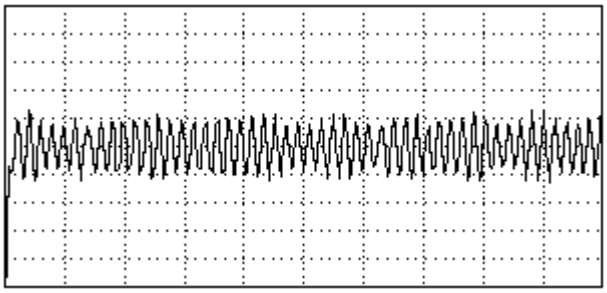


Bild 4.6: Typischer Ist-Geschwindigkeitsverlauf bei einem zu großen Wert von Kp

Sie müssen Kp schrittweise erhöhen, bis die oben genannten Erscheinungen auftreten. Danach wird Kp wieder schrittweise verkleinert, bis diese Erscheinungen nicht mehr auftreten. Damit ist der Parameter Kp fertig eingestellt. Ab jetzt können Sie die Bewegungsgeschwindigkeit V noch variieren, um Kp feiner einzustellen. Im Bild 4.7 können Sie den typischen Ist-Geschwindigkeitsverlauf nach dem Einstellen von Kd und Kp sehen.

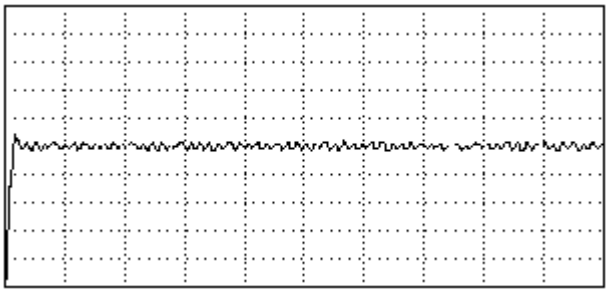


Bild 4.7: Typischer Ist-Geschwindigkeitsverlauf nach dem Einstellen von Kd und Kp

Schritt 5: Einstellen von Ki

Ein Ki ungleich 0 ist notwendig, um eine bleibende Abweichung zwischen dem Soll- und Ist-Wert auszuregeln, d. h. es führt zu einer Regelabweichung gleich 0. Aber ein Ki ungleich 0 führt zwangsläufig immer zu einem Überschwingen. Um diesen Widerspruch zu lösen, ist es oft notwendig, einen Geschwindigkeitsregler mit Integrationsanteil zu benutzen.
Der Integrationsanteil des Geschwindigkeitsregler sorgt dafür, dass die bleibende Regelabweichung verschwindet. In diesem Fall brauchen Sie Ki gar nicht einzustellen. Ki wird einfach gleich 0 gesetzt. In machen Fällen können Sie Ki einen ganz kleinen Wert zuweisen, im Allgemein ist es aber nicht notwendig. Damit hat die Achse sowohl kein Überschwingen als auch keine Regelabweichung.

Wenn die Geschwindigkeitsregler nicht vorhanden sind, ist das Einstellen von Ki ein muss. Im Folgenden wollen wir Ihnen erläutern, wie Sie es machen sollten.
Sie müssen Td gleich 0 setzen. Der Integrallimit Il darf nicht gleich 0 sein. Wir haben Il immer gleich 2048 gesetzt. Diesen Standardwert sollten Sie auch für Ihre Anlage übernehmen. Kd und Kp haben die Werte, die Sie in den Schritten 3 und 4 eingestellt haben. In diesem Schritt müssen Sie Ki schrittweise erhöhen. Der Anfangswert von Ki soll 2 oder 3 sein. Ein Vergrößerungsschritt von 10 ist angemessen. Nach jeder neuen Eingabe von Ki müssen Sie mit dem Betätigen von F1 oder F2 die Achse in Bewegung setzen, um den Ist-Geschwindigkeitsverlauf aufzunehmen. Je größer Ki ist, desto besser. Aber ein zu großer Wert von Ki führt zu folgenden Erscheinungen:

- Die Ist-Geschwindigkeit schwingt während der Bewegung um den Sollwert (siehe Bild 4.8).
- Bei einem sehr großen Wert von Ki können Sie wegen der Schwingung der Ist-Geschwindigkeit um den Sollwert wieder ein deutliches Bewegungsgeräusch hören.
- Während des Ruhezustands neigt die Achse sehr leicht zu Dauerschwingungen um ihre Position. Durch einen leichten Anstoß können Sie sofort bei der Achse kleine Dauerschwingungen hervorrufen. Diese Dauerschwingungen sind wiederum durch ein deutliches Schwingungsgeräusch zu erkennen.

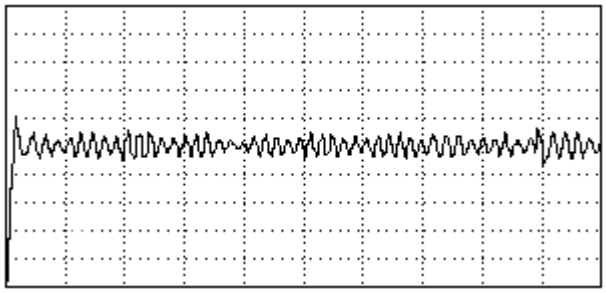


Bild 4.8: Typischer Ist-Geschwindigkeitsverlauf bei einem zu großen Wert von Ki

Sie müssen Ki schrittweise erhöhen, bis die oben genannten Erscheinungen auftreten. Danach Ki wieder schrittweise verkleinern, bis diese Erscheinungen nicht mehr auftreten.
Damit ist Ki fertig eingestellt. Eine Variation von V kann sehr nützlich sein, um Ki noch feiner einzustellen. Im Bild 4.9 können Sie den typischen Ist-Geschwindigkeitsverlauf nach dem Einstellen von Kd, Kp und Ki sehen.

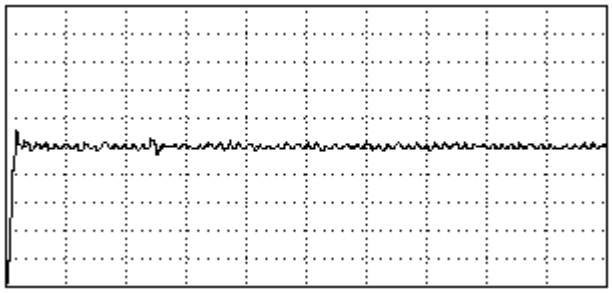


Bild 4.9: Typischer Ist-Geschwindigkeitsverlauf nach dem Einstellen von Kd, Kp und Ki

Schritt 6: Einstellen von Td

Die Parameter Kd, Kp, Ki und Il haben die Werte, die Sie in den letzten Schritten eingestellt haben. Td hatte bis jetzt den Wert 0. In diesem Schritt müssen Sie Td schrittweise um 1 erhöhen. Sie sollten darauf achten, dass Sie jedes Mal Kd halbieren müssen, wenn Sie Td um 1 erhöhen. Nach jeder neuen Eingabe sollten Sie mit Hilfe von F1 oder F2 die Achse bewegen, um den Ist-Geschwindigkeitsverlauf aufzunehmen.
Bei einer Vergrößerung von Td werden Sie bemerken, dass sich das Bewegungsgeräusch deutlich vermindert. Ein zu großer Wert von Td führt zu einer Schwingung der Ist-Geschwindigkeit um den Sollwert (siehe Bild 4.10). Deswegen sollten Sie Td nur dann weiter vergrößern, solange Sie keine Verschlechterung des Ist-Geschwindigkeitsverlaufes bemerken. Eine Verbesserung des Regelverhaltens durch die Vergrößerung von Td ist normalerweise nur bei trägen Achsen erreichbar. Eine hoch dynamische Achse neigt sehr schnell zu einer Dauerschwingung, wenn Sie Td vergrößern.
Bei unseren Anlagen liegt Td meistens im Bereich 0 ... 2.

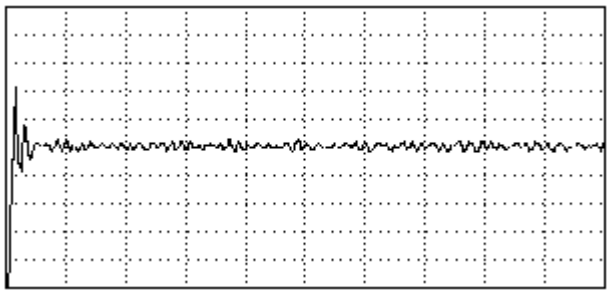


Bild 4.10: Typischer Ist-Geschwindigkeitsverlauf bei einem zu großen Wert von Td

Während des Einstellens der Reglerparameter ist es sehr empfehlenswert, dass Sie die Achse in beide Richtungen bewegen, um die Ist-Geschwindigkeits-verläufe in beide Richtungen aufnehmen und bewerten zu können. Das Bild 4.11 zeigt Ihnen den Ist-Geschwindigkeitsverlauf bei optimal eingestellten Reglerparametern.

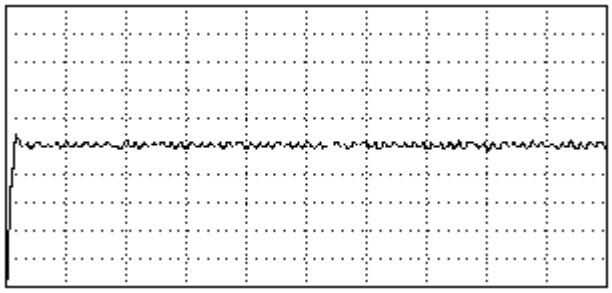


Bild 4.11: Typischer Ist-Geschwindigkeitsverlauf bei optimal eingestellten Reglerparametern

Schritt 7: Ermitteln der Anregelzeit und der Überschwingweite

Sie können die Anregelzeit und die Überschwingweite nicht einstellen sondern nur anhand der Sprungantwort ermitteln. Diese beiden Parameter sind Kennwerte der geregelten Achse und hängen sehr stark von den eingestellten Reglerparametern ab.
Bevor Sie mit ESCAPE das Regler-Einstellmenü beenden, müssen Sie diese beiden Kennwerte ermitteln, weil das nächste Arbeitsmenü diese beiden Kennwerte verlangt, um den Geschwindigkeitsfaktor und den Dämpfungsgrad zu ermitteln.

Die Überschwingweite ist die Differenz zwischen der Spitze des ersten Überschwingens der Sprungantwort und dem Sollwert. Im Fall einer Überschwingweite größer als 0 ist die Anregelzeit die Zeit zwischen dem Zeitpunkt 0 und dem Zeitpunkt, in dem die Sprungantwort zum erstenmal den Sollwert erreicht. Im Bild 4.12 wollen wir verdeutlichen, wie Sie diese beiden Kennwerte anhand der Sprungantwort ermitteln können.

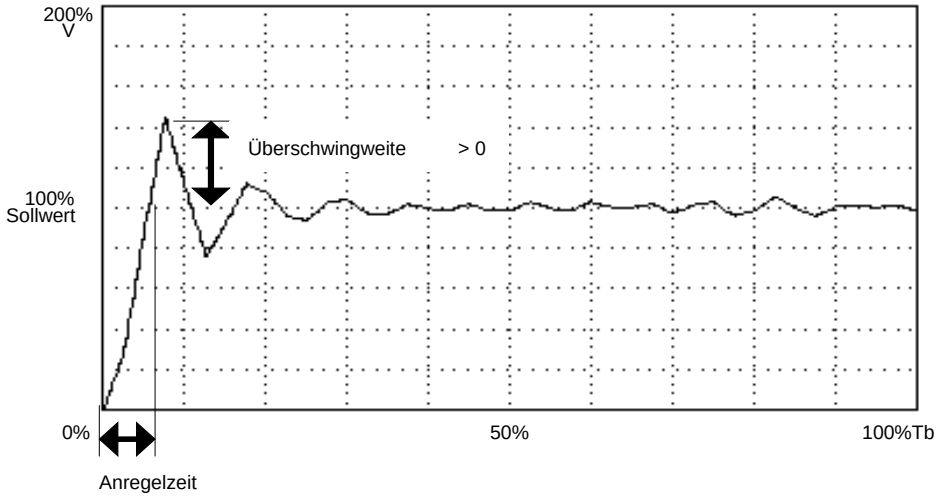


Bild 4.12: Ermitteln der Anregelzeit und der Überschwingweite im Fall einer Überschwingweite größer als 0

Die Sprungantwort im Bild 4.12 ist ein bisschen übertrieben dargestellt, um die Ermittlung der Überschwingweite zu verdeutlichen. In vielen Fällen haben wir keine Überschwingweite (siehe Bild 4.13), d. h. der Wert der Überschwingweite ist gleich 0. Hier ist die Anregelzeit die Zeit zwischen dem Zeitpunkt 0 und dem Zeitpunkt, in dem die Sprungantwort den Sollwert erreichen könnte, falls die Sprungantwort ihre Anfangssteigung beibehalten würde.

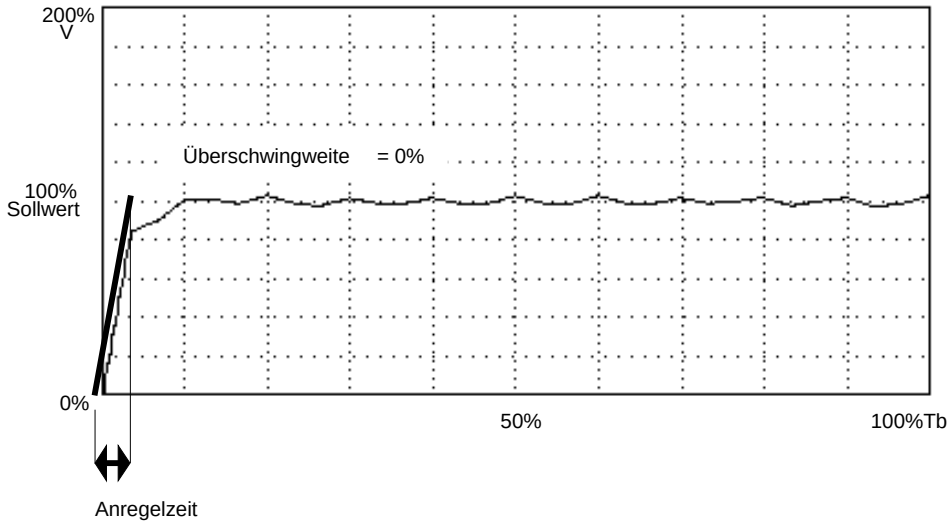


Bild 4.13: Ermitteln der Anregelzeit und der Überschwingweite im Fall einer Überschwingweite gleich 0

Um die Anregelzeit und die Überschwingweite einfacher ablesen zu können, sollten Sie die Zeitbasis Tb verkleinern. Es werden die Anregelzeit in Prozent der Zeitbasis Tb und die Überschwingweite in Prozent des Sollwertes abgelesen. In Bild 4.12 haben wir z. B. eine Anregelzeit von 7 (% Tb) und eine Überschwingweite von 42 (% Sollwert). Im Bild 4.13 haben wir eine Anregelzeit von 4 (% Tb) und eine Überschwingweite von 0 (% Sollwert). Der Wert der Anregelzeit und der Wert der Überschwingweite müssen Sie für das nächste Arbeitsmenü notieren.
Das Arbeitsmenü, das nach dem ESCAPE zum Beenden des Regler-Einstellmenüs kommt, verlangt die Eingabe der Anregelzeit und der Überschwingweite, um den Geschwindigkeits-Verstärkungsfaktor und den Dämpfungsgrad der Achse zu berechnen.
Der Geschwindigkeits-Verstärkungsfaktor ist der dynamische Kennwert der Achse, den der Treiber für die interne Berechnung braucht. Deswegen wird dieser Faktor in die Initialisierungsdatei übernommen. Je größer dieser Faktor ist, desto besser ist die Dynamik der Achse.

Aufgrund der Anlagensymmetrie ist es oft wünschenswert, dass alle Achsen einer Anlage ungefähr dieselbe Dynamik besitzen, d. h. Sie sollten die Reglerparameter so einstellen, dass die Geschwindigkeits-Verstärkungsfaktoren der Achsen etwa gleich sind.
In der Praxis wird häufig so verfahren, dass die Reglerparameter der Achsen zuerst so eingestellt, dass alle Achsen die maximal mögliche Dynamik haben.

Die Achse mit der schlechtesten Dynamik wird als Bezugsachse genommen. Die Reglerparameter der anderen Achsen werden so variiert, dass diese Achsen annähernd die gleiche Dynamik wie die der Bezugsachse haben. Beim Variieren der Reglerparameter gilt im Allgemeinen, dass große Werte von Kd und Kp zu einem großen Geschwindigkeits-Verstärkungsfaktor führen und umgekehrt.

Um Ihnen den Vergleich der Achsdynamik zu erleichtern, werden die Geschwindigkeits-Verstärkungsfaktoren aller Achsen im letzten Arbeitsmenü eingeblendet.

Der Dämpfungsgrad ist ein Kennwert für die Schwingneigung der Achse. Der Wert der Dämpfung liegt im Bereich 0 ... 1. Je größer der Dämpfungsgrad ist, desto kleiner die Schwingneigung und desto besser das Achsverhalten. Ein Dämpfungsgrad größer oder gleich 0,7 ist wünschenswert.

4.8 Das Untermenü *Übersetzung*

Bei vielen Anlagen ist es nicht einfach, den Übersetzungsfaktor der Achsen zu bestimmen. Hier denken wir z. B. an Achsen mit Zahnriemen.
In diesem Untermenü haben Sie die Möglichkeit, den Achsübersetzungsfaktor experimentell zu bestimmen.
Zuerst wird der alte Übersetzungsfaktor aus der Initialisierungsdatei entnommen und angezeigt. Sie können selbst entscheiden, ob Sie den Übersetzungsfaktor neu einstellen wollen. Falls ja wird das nächste Arbeitsmenü geöffnet. In diesem Arbeitsmenü können Sie mit den Tasten F1 oder F2 Teach-In-Bewegungen durchführen. Beim Betätigen der Taste F10 können Sie jederzeit den aktuellen Punkt als den Nullpunkt der Achse definieren.
Über das Eingabefeld V_Teach-In haben Sie die Möglichkeit, die Teach-In-Geschwindigkeit beliebig zu definieren. Während der Teach-In-Bewegung wird die Anzahl der zurückgelegten Inkremente on-line angezeigt.

Um den Übersetzungsfaktor der Achse zu bestimmen, gehen Sie wie folgt vor:

- Zuerst wird die Achse bis zu einem Ende bewegt. Hier markieren Sie den Anfangspunkt auf der Achse. Durch das Betätigen der Taste F10 definieren Sie den Anfangspunkt gleichzeitig als den Nullpunkt der Achse. Dieser Nullpunkt der Achse dient als Bezugspunkt für das Zählen der zurückgelegten Inkremente.
- Danach bewegen Sie die Achse zum anderen Ende. Hier markieren Sie den Endpunkt auf der Achse. Mit dem Betätigen der Taste ENTER kommen Sie zum nächsten Arbeitsmenü, in dem die Anzahl der zwischen dem Anfangspunkt und dem Endpunkt zurückgelegten Inkremente vom letzten Arbeitsmenü übernommen wird. Hier müssen Sie noch die Bewegungslänge ermitteln und eingeben.

Bei einer Linearachse ist die Bewegungslänge der Längenabstand zwischen dem markierten Anfangspunkt und dem markierten Endpunkt. Die Längeneinheit ist Mikrometer.
Bei einer Drehachse ist die einzugebende Bewegungslänge der Drehwinkel zwischen dem Anfangspunkt und dem Endpunkt. Die Winkeleinheit ist Bogensekunde.
Aus der eingegebenen Bewegungslänge, der Anzahl der zurückgelegten Inkremente und der Anzahl der Encoder-Linien berechnet das Einstellprogramm automatisch für Sie den Achsübersetzungsfaktor.
Sie haben noch die Möglichkeit, den errechneten Wert zu editieren. Nach dem Betätigen der Taste ENTER wird der neue Wert automatisch in die Initialisierungsdatei übernommen.
Mit einem neuen Übersetzungsfaktor ergibt sich zwangsläufig die Notwendigkeit, die Rampenparameter der Achse (Maximalachs-Geschwindigkeit und Maximalachs-Beschleunigung) dem neuen Übersetzungsfaktor anzupassen (siehe Abschnitt 4.9). Hier können Sie frei entscheiden, ob die Rampenparameter aktualisiert werden sollen oder nicht. Falls Sie wollen, erfolgt die Aktualisierung automatisch.
An folgenden Beispielen wollen wir Ihnen verdeutlichen, wie das Einstellprogramm intern den Achsübersetzungsfaktor berechnet.

Beispiel 1: Übersetzungsfaktor einer Drehachse

Die Anzahl der Encoder-Linien ist 1000. Die Anzahl der zurückgelegten Inkremente ist 25455. Der Abstand zwischen dem Anfangspunkt und dem Endpunkt beträgt 74 Grad. Aus diesen Daten erhalten wir den Übersetzungsfaktor:

74 * 3 600 Bogensekunde * (4 * 1 000 Inkr./Motorumdrehung)/ 25 455 Inkr.
= 41 862 Bogensekunden/Motorumdrehung

Der Faktor 4 in der obigen Formel ist deswegen notwendig, weil der Chip LM628 intern die Encoder-Impulse vervierfacht.

Beispiel 2: Übersetzungsfaktor einer Linearachse

Die Anzahl der Encoder-Linien ist 500. Die Anzahl der zurückgelegten Inkremente ist 34 789. Der Abstand zwischen dem Anfangspunkt und dem Endpunkt beträgt 89 mm. Aus diesen Daten erhalten wir den Übersetzungsfaktor:

89 000 μm * (4 * 500 Inkr./Motorumdrehung)/34 784 Inkr.
= 5 117 μm/Motorumdrehung

Analog zum ersten Beispiel ist der Faktor 4 auch hier notwendig.

4.9 Das Untermenü *Rampe*

Die Dynamik einer Achse wird durch die maximale Beschleunigung und durch die maximale Geschwindigkeit ausgedrückt. Diese beiden Rampenparameter kann man experimentell anhand der Ist-Geschwindigkeitsverläufe ermitteln. In diesem Untermenü bieten wir Ihnen entsprechende Hilfsmittel, um den Ist-Geschwindigkeitsverlauf einer Achse aufzunehmen.

4.9.1 Ermitteln der maximalen Beschleunigung

Das Arbeitsmenü für die Aufnahme der Ist-Geschwindigkeitsverläufe ist im Bild 4.14 dargestellt.

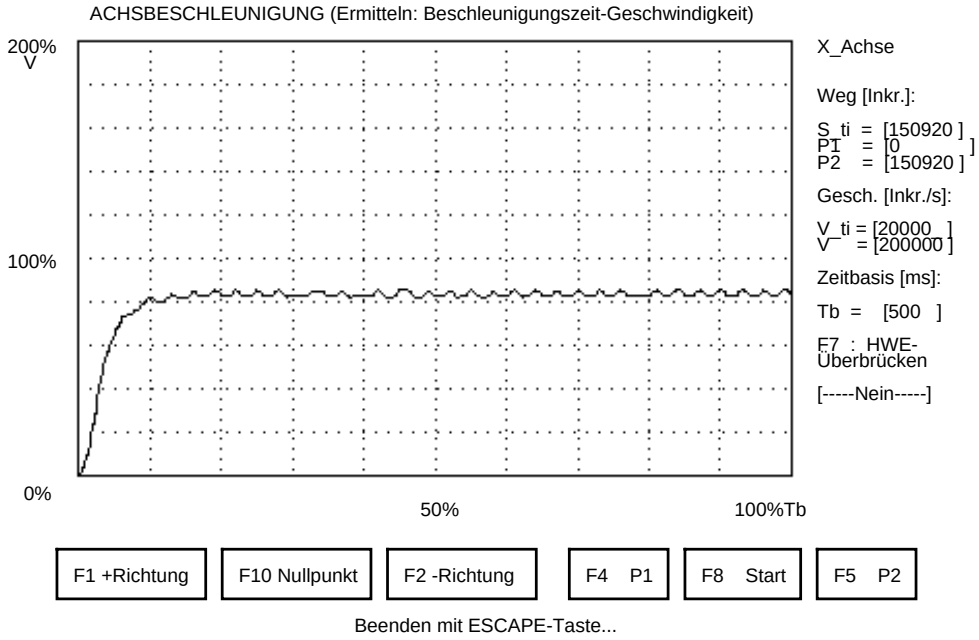


Bild 4.14: Arbeitsmenü für das Ermitteln der Maximalbeschleunigung

An der ersten Stelle auf der linken Seite haben Sie die Variable S_ti. Hier wird die Anzahl der zurückgelegten Inkremente on-line während der Teach-In-Bewegung angezeigt. Die Geschwindigkeit für die Teach-In-Bewegung wird über die Variable V_ti eingegeben.
Mit der Variable V können Sie den Maßstab für die Ordinatenachse definieren. Gleichzeitig wird hier die maximal zu bewegende Geschwindigkeit während der Aufnahme der Ist-Geschwindigkeitsverläufe auf den Wert dieser Variable begrenzt.
Die Einheit der Geschwindigkeit ist Inkrement/Sekunde. Den Maßstab für die Abszissenachse können Sie mit der Zeitbasis Tb festlegen. Die Einheit von Tb ist Millisekunde. Mit Hilfe der Taste F7 ist es jederzeit möglich, die Endlageschalter zu überbrücken. Über die Bedienung der Eingabefelder können Sie im Abschnitt 4.7.2 nachlesen.

Um die maximale Beschleunigung der Achse zu bestimmen, gehen Sie wie folgt vor:
Mit den Funktionstasten F1 oder F2 wird die Achse bis zu einem ihrer Enden bewegt. Hier definieren Sie mit der Taste F4 den Punkt 1. Danach wird die Achse zum anderen Ende bewegt. Hier definieren Sie mit der Taste F5 den Punkt 2. Die Koordinaten dieser beiden Punkte werden über die Variablen P1 und P2 angezeigt und intern vom Einstellprogramm gespeichert.

Jedes Mal wenn Sie mit der Taste F10 den Nullpunkt der Achse neu definieren, werden die beiden Koordinaten entsprechend korrigiert. Am Anfang sind diese Koordinaten gleich Null. Nachdem der Punkt 1 und der Punkt 2 definiert sind, ermittelt das Einstellprogramm jedes Mal beim Betätigen der Taste F8, welcher Punkt am weitesten von der aktuellen Position entfernt liegt.
Danach wird die Achse mit einem maximal möglichen Beschleunigungswert beschleunigt und zwar - falls möglich - bis zu der in der Variable V definierten Geschwindigkeit. Die Begrenzung der Fahrgeschwindigkeit durch die Variable V ist in solchen Fällen sinnvoll, wo eine hohe Bewegungsgeschwindigkeit z. B. wegen der Gefahr für Mensch und Maschine nicht erwünscht ist. Wenn der am weitesten entfernt liegende Punkt erreicht ist, wird die Bewegung gestoppt.

Während der Bewegung wird der Ist-Geschwindigkeitsverlaufs on-line angezeigt. Anhand des aufgenommenen Geschwindigkeitsverlaufs können Sie zwei Zahlen ermitteln:
Erstens wie viel Prozent des in der Variable V definierten Wertes erreicht die Endgeschwindigkeit.
Zweitens wie viel Prozent der in der Variable Tb definierten Zeitbasis wird für das Beschleunigen der Achse bis zur Endgeschwindigkeit benötigt.
Diese beiden Prozentsätze sollten Sie sich notieren, um sie im nächsten Arbeitsmenü wieder eingeben zu können.

Es ist im Allgemeinen aufgrund der Asymmetrie der Achse empfehlenswert, dass Sie die Rampe sowohl in positiver als auch in negativer Richtung aufnehmen. Daraus werden entsprechende Prozentsätze für die Maximalgeschwindigkeit und die Beschleunigungszeit abgelesen. Der kleinere Prozentsatz der Maximalgeschwindigkeit und der größere Prozentsatz der Beschleunigungszeit dienen dann als Grundlage für die Berechnung der Maximalbeschleunigung.

Aus Sicherheitsgründen haben wir dies bei der Aufnahme der Rampe so realisiert, dass die Bewegung nur solange ausgeführt wird, wie Sie die Taste F8 niederhalten. Beim Loslassen wird die Bewegung sofort unterbrochen.

Über die Taste ESCAPE beenden Sie das Arbeitsmenü für die Aufnahme der Ist-Geschwindigkeitsverläufe. Das nächste Arbeitsmenü verlangt die Eingabe der ermittelten Maximalachs-Geschwindigkeit und der Beschleunigungszeit. Aufgrund der Lastschwankungen während der Bearbeitung muss jede Achse immer eine gewisse Reserve bei der Beschleunigung haben. Diese Reserve

wird durch den von Ihnen einzugebenden Beschleunigungsfaktor berücksichtigt.
Mit diesem Beschleunigungsfaktor teilen Sie dem Einstellprogramm mit, wie viel Prozent des maximal zur Verfügung stehenden Beschleunigungswerts Sie als die Maximalbeschleunigung der Achse definieren wollen. In unserer Anlage geben wir für den Beschleunigungsfaktor normalerweise einen Wert von etwa 80 % ein. Sie sollten auch daran denken, dass sich die Achse um so weicher bewegt, je kleiner der Beschleunigungswert ist.

Nachdem alle Eingaben mit der Taste ENTER bestätigt sind, berechnet das Einstellprogramm automatisch die Maximalbeschleunigung für Sie. Diese Werte werden angezeigt. Sie können dann entscheiden, ob diese Werte in die Initialisierungsdatei aufgenommen werden sollen oder nicht.

Zum genaueren Verständnis wollen wir Ihnen hier am Beispiel im Bild 4.14 erläutern, wie das Einstellprogramm aus den eingegebenen Daten die Maximalbeschleunigung für einen Beschleunigungsfaktor vom 80 % berechnet. Für die Endgeschwindigkeit gilt:

$$V\_end = 91\ \% * 200\ 000\ \text{Inkr./s} = 182\ 000\ \text{Inkr./s}$$

Um diese Endgeschwindigkeit zu erreichen, braucht die Achse eine Beschleunigungszeit von 10 % * Tb. Daraus folgt:

$$\begin{aligned} A\_max &= 80\ \% * (V\_end/(10\ \% * 500\ \text{ms})) \\ &= 80\ \% * (182\ 000\ \text{Inkr./s}/0{,}05\ \text{s}) \\ &= 2\ 912\ 000\ \text{Inkr./s}^2 \end{aligned}$$

Aufgrund der Encoder-Linienzahl und des Übersetzungsfaktors der Achse wird die Einheit Inkrement in die Einheit Mikrometer (bei einer Linearachse) bzw. Bogensekunde (bei einer Drehachse) umgerechnet. Wir nehmen als Beispiel eine Linearachse. Die gleiche Berechnungsweise gilt auch für eine Drehachse. Nehmen wir an, die Achse habe einen Encoder mit 1 000 Linien und einen Übersetzungsfaktor von 4 000 μm/Motorumdrehung. Vergessen Sie nicht, dass der Controllerchip LM628 die Encoder-Linienzahl vervierfacht. Daraus folgt:

$$\begin{aligned} A\_max &= 2\ 912\ 000\ \text{Inkr./s}^2 * 4\ 000\ \mu\text{m/Motorumdrehung}/ \\ &\qquad (4 * 1\ 000\ \text{Inkr./Motorumdrehung}) \\ &= 2\ 912\ 000\ \mu\text{m/s}^2 \end{aligned}$$

4.9.2 Ermitteln der maximalen Geschwindigkeit

Ähnlich wie bei der Bestimmung der maximalen Achsbeschleunigung wird die maximale Achsgeschwindigkeit auch anhand von Ist-Geschwindigkeits-verläufen experimentell ermittelt. Das Bild 4.15 zeigt Ihnen das Arbeitsmenü für die Aufnahme der Ist-Geschwindigkeitsverläufe.

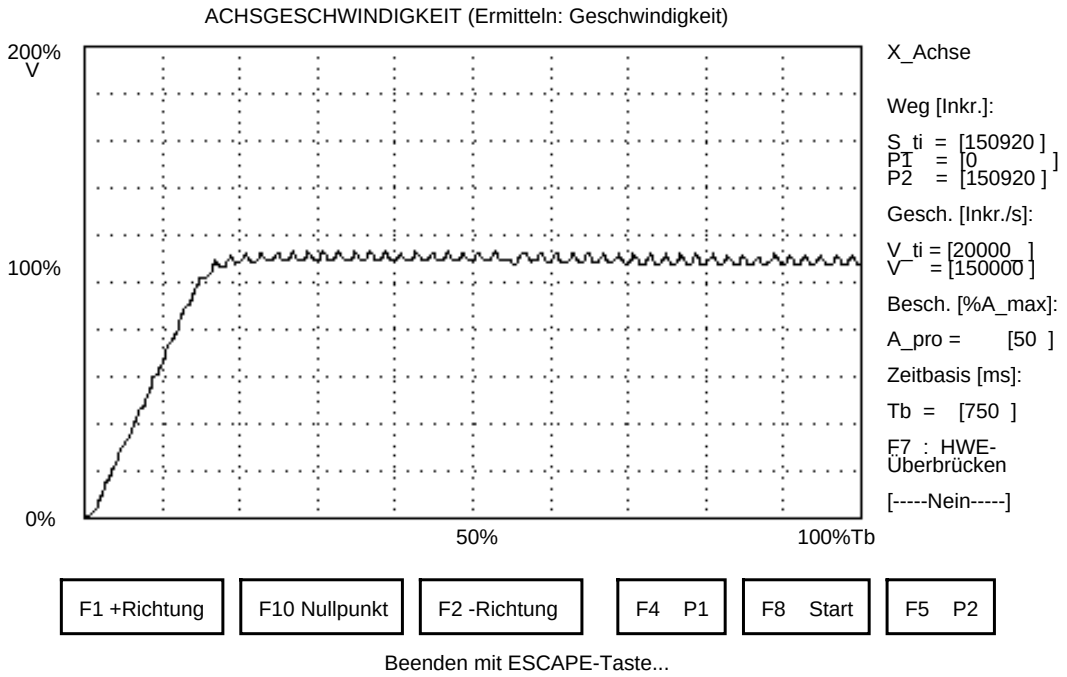


Bild 4.15: Arbeitsmenü für das Ermitteln der Maximalgeschwindigkeit

Die Art und Weise, wie Sie Ist-Geschwindigkeitsverläufe aufnehmen können, haben wir bereits im Abschnitt 4.9.1 erläutert. Es gibt hier aber zwei Besonderheiten, die Sie beachten müssen.

Die erste betrifft die auf der linken Seite stehende Variable V. Hier wird ein Geschwindigkeitswert mit der Einheit Inkrement/Sekunde eingegeben. Dieser Wert ist der Maßstab der Ordinatenachse. Bei der Ermittlung der Beschleunigung stellt dieser Wert gleichzeitig die Obergrenze für die Endgeschwindigkeit der Bewegung dar (siehe Bild 4.14). Hier hat dieser Wert aber keinen Einfluss auf die Endgeschwindigkeit der Rampe. Wir wollen schließlich die Maximalgeschwindigkeit der Achse bestimmen.

Die zweite Besonderheit betrifft die Variable A_pro. Auf dem Arbeitsmenü für die Beschleunigung haben wir diese Variable nicht. Mit Hilfe dieser Variable können Sie einen bestimmten Prozentsatz eingeben. Der Wert kann im Bereich 1 ... 100 variieren. Aus diesem Prozentsatz und der von Ihnen im letzten Arbeitsmenü ermittelten Maximalbeschleunigung berechnet das Einstellprogramm den Beschleunigungswert, mit dem die Achse zu der Maximalgeschwindigkeit beschleunigt wird, d. h. mit der Variable A_pro können Sie die Rampe flacher oder steiler gestalten. Diese Variable hat aber keinen Einfluss auf die Endgeschwindigkeit der Rampe.

Standardmäßig hat die Variable A_pro den Wert 100, d. h. die Rampe wird mit der Maximalbeschleunigung ausgeführt. Falls Sie während des Einstellvorgangs die Mechanik schonen wollen, sollten Sie der Variable A_pro einen kleineren Wert zuweisen, um einen weichen Verlauf der Rampe zu erreichen.

In diesem Arbeitsmenü wird die Achse bis zu der maximal möglichen Geschwindigkeit beschleunigt. Die Rampe wird on-line angezeigt. Anhand des aufgenommenen Ist-Geschwindigkeitsverlaufs können Sie ermitteln, wie viel Prozent des in der Variable V definierten Wertes die Rampen-Endgeschwindigkeit beträgt. Diesen Prozentsatz sollten Sie sich notieren, um ihn im nächsten Arbeitsmenü wieder eingeben zu können.

Nachdem Sie den experimentellen Vorgang mit der Taste ESCAPE beendet haben, erscheint das nächste Menü, in das der von Ihnen während des experimentellen Vorgangs ermittelten Prozentsatz für die Maximalgeschwindigkeit eingegeben werden muss.

Ähnlich wie bei der Beschleunigung können Sie hier auch mit dem sogenannten Geschwindigkeitsfaktor selbst definieren, wie viel Prozent des maximal zur Verfügung stehenden Geschwindigkeitswertes die Maximalgeschwindigkeit der Achse betragen soll. Aber hier ist es nicht so kritisch wie bei der Beschleunigung.
Für den Geschwindigkeitsfaktor können Sie ohne Bedenken einen Wert von 100 % eingeben. Bei unseren Anlagen nehmen wir normalerweise einen Wert von etwa 90 %. Nachdem alle Eingaben mit der Taste ENTER bestätigt sind, berechnet das Einstellprogramm automatisch die Maximalgeschwindigkeit für Sie. Dieser Wert wird angezeigt. Sie können dann entscheiden, ob dieser Wert in die Initialisierungsdatei aufgenommen werden soll oder nicht.

Obwohl alle Berechnungen von dem Einstellprogramm erledigt werden, wollen wir Ihnen am Beispiel im Bild 4.15 zwecks eines besseren Verständnisses erläutern, wie die Maximalgeschwindigkeit intern berechnet wird.

Die Maximalgeschwindigkeit der Achse beträgt 110 % von V. Für einen Geschwindigkeitsfaktor von 90 % haben wir:

$$\begin{aligned} V\_max &= 90\ \% * (110\ \% * 150\,000\ Inkr./s) \\ &= 148\,500\ Inkr./s \end{aligned}$$

Für einen Encoder mit 1 000 Linien und einen Übersetzungsfaktor von 4 000 μm/Motorumdrehung gilt:

$$\begin{aligned} V\_max &= 148\,500\ Inkr./s * 4\,000\ \mu m/Motorumdrehung / \\ &\qquad (4 * 1\,000\ Inkr./Motorumdrehung) \\ &= 148\,500\ \mu m/s \end{aligned}$$

4.10 Das Untermenü *Schalter*

Nach der Auswahl dieses Untermenüs erscheint das Arbeitsmenü auf dem Bildschirm (siehe Bild 4.16).

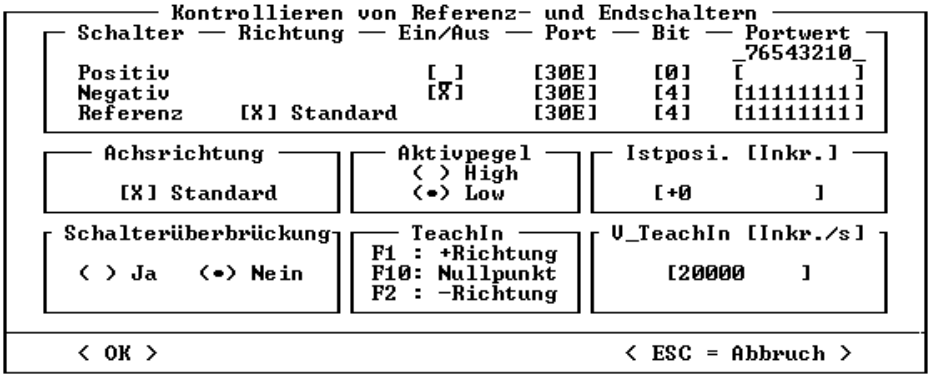


Bild 4.16: Arbeitsmenü bei der Auswahl des Untermenüs Schalter

Das Inbetriebnahmeprogramm übernimmt sämtliche Informationen über die End- und Referenzschalter sowie über die Achsrichtungen von der Initialisierungsdatei und zeigt sie im Arbeitsmenü an. Ähnlich wie bei dem Konfigurationsprogramm können Sie hier alle Angaben für den positiven bzw. den negativen Hardware-Endschalter der Achse eingeben. Sie können den jeweiligen Hardware-Endschalter freilassen oder sperren. Das Sperren und das Freilassen sieht so aus:

Ein/Aus
- [X] ---> Der jeweilige Endschalter wird benutzt.
- [ ] ---> Der jeweilige Endschalter wird nicht benutzt.

Für jeden benutzten Endschalter müssen Sie noch angeben, an welchem Port und an welchem Bit der Endschalter angeschlossen ist.
Jede Achse muss einen Referenzschalter haben. Ähnlich wie bei den Endschaltern müssen Sie auch hier die Portadresse und die Bitnummer angeben. Außerdem müssen Sie noch angeben, ob der Referenzschalter auf der negativen oder positiven Seite liegt. Das Definieren der Referenzschalterrichtung sieht so aus:

Richtung
- [X] Standard ---> Der Referenzschalter liegt auf der negativen Seite
- [ ] Standard ---> Der Referenzschalter liegt auf der positiven Seite

Für jede Achse hat die PC-Einsteckkarte bereits die positive und die negative Richtung festgelegt. Trotzdem haben Sie hier die Möglichkeit, diese vordefinierten Richtungen beizubehalten oder zu ändern. Die Beibehaltung der vordefinierten Richtungen erfolgt durch die Auswahl der Standardeinstellung.

Wenn Sie die Standardeinstellung nicht auswählen, werden die positive und die negative Richtung umgetauscht. Die Einstellung sieht so aus:

Achsrichtung
[X] Standard ---> Beibehaltung der vordefinierten Richtungen
[ ] Standard ---> Umtauschen der vordefinierten Richtung

Weiterhin haben Sie die Möglichkeit, den aktiven Pegel der Referenz- bzw. der Hardware-Endschalter zu definieren. Der aktive Pegel (High oder Low) liegt an einem Eingabeport vor, wenn der entsprechende Referenz- bzw. Hardware-Endschalter angefahren ist.
Beachten Sie, dass der definierte aktive Pegel für alle Referenzschalter und für alle Hardware-Endschalter Ihrer Anlage gilt.
Über die Funktionstasten F1 und F2 können Sie eine Teach-In-Bewegung mit der Achse durchführen. Die Teach-In-Geschwindigkeit lässt sich über das Eingabenfeld V_Teach-In frei definieren.

Um die Portadresse und die Bitnummer des jeweiligen Schalters sowie den aktiven Pegel zu kontrollieren, müssen Sie mit F1 oder F2 die Achse in die jeweilige Richtung bewegen. Der Inhalt jedes definierten Ports wird bitweise on-line angezeigt. Beim Betätigen des jeweiligen Schalters muss das entsprechende Bit gesetzt oder zurückgesetzt werden. Anhand dieser Anzeige können Sie sehr leicht feststellen, ob sie vorhanden ist oder nicht.

Beachten Sie, dass eine mit F1 durchgeführte positive Teach-In-Bewegung den positiven Endschalter sowie den Referenzschalter betätigen muss, falls der positive Schalter vorhanden ist und falls der Referenzschalter auf der positiven Seite liegt (die Position des Referenzschalters auf der positiven Seite wird als "nicht standard" bezeichnet.). Eine mit F2 durchgeführte negative Teach-In-Bewegung betätigt den negativen Endschalter sowie den Referenzschalter, falls der negative Endschalter vorhanden ist und falls der Referenzschalter auf der negativen Seite liegt (die Position des Referenzschalters auf der negativen Seite wird als "standard" bezeichnet).

Die Teach-In-Bewegungen mit F1 und F2 sind für Sie sehr nützlich bei der Auswahl der Achsrichtungen. Beachten Sie, dass das Betätigen der Tasten F1 bzw. F2 immer eine positive bzw. eine negative Bewegung hervorruft.
Eine Änderung der Richtungsauswahl führt sofort zu einer Änderung der Bewegungsrichtung. Bei einer Änderung der Achsrichtung müssen Sie natürlich die Portadresse, die Bitnummern aller Schalter sowie die Richtung des Referenzschalters entsprechend umtauschen.

Mit der Taste ENTER bestätigen Sie die Eingabe. Die neue Konfiguration wird automatisch in die Initialisierungsdatei übernommen.

4.11 Das Untermenü *Überbrückung*

Der physikalische Arbeitsbereich Ihrer Anlage wird durch die Hardware-Endlageschalter begrenzt. Es ist empfehlenswert, die Hardware-Endlageschalter in den Sicherheitskreis zu integrieren. Um die Motoren sowie die Mechanik zu schützen, werden die Leistungsendstufen in *isel*-Servocontrollern bei einem Hardware-Endlageschalterfehler sofort ausgeschaltet. Das aber bedeutet, dass man ohne besondere Maßnahmen die Achse bzw. die Achsen nicht mehr aus den Hardware-Endlageschaltern herausfahren kann.
Aus diesem Grund benutzen wir in unserem Servocontroller das Bit 5 des Ausgabeports mit der Adresse (Basisadresse+0Ch) auf der PC-Einsteckkarte, um den Sicherheitskreis zu überbrücken. Danach ist das Einschalten der Leistungsendstufen wieder möglich.

Wenn Sie den Sicherheitskreis Ihres Controllers auf eine andere Art und Weise realisieren, brauchen Sie diesen Abschnitt nicht zu lesen. Im Bild 4.17 haben wir das Arbeitsmenü für die Kontrolle der Schalterüberbrückung.

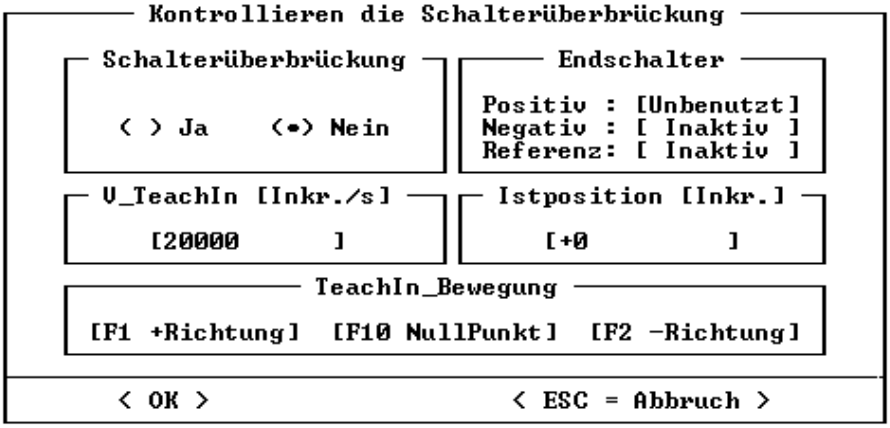


Bild 4.17: Arbeitsmenü bei der Auswahl des Untermenüs Schalterüberbrückung

Das Kontrollieren der Schalterüberbrückung ist eine sehr einfache Sache. Mit den Funktionstasten F1 und F2 fahren Sie die Achse an den jeweiligen Endlageschalter.

- Im fehlerfreien Fall müssen die Leistungsendstufen ausgeschaltet werden, falls Sie im Feld „Schalterüberbrückung“ die Option „Nein“ gewählt haben.
- Wenn die Option „Ja“ aktiviert ist, bleiben die Leistungsendstufen an.
- Wenn sich der Controller etwas anders verhält, haben Sie einen Fehlerfall vor sich.

Um Ihnen das Leben leichter zu machen, werden die Zustände aller Endlageschalter und des Referenzschalters ständig eingelesen und on-line dargestellt. Die Teach-In-Geschwindigkeit ist frei definierbar.

4.12 Das Untermenü *Enable*/*Disable*

Es ist oft wünschenswert, die Motoren stromlos zu schalten. Damit haben Sie die Möglichkeit, die Achsen per Hand zu bewegen. Das Ein- und Ausschalten von Motorströmen wird durch die Ausgabe entsprechender Werte an einem Ausgang realisiert (siehe Handbuch *isel*-Servosteuerkarte UPMV 4/12).

Bei unserem Servocontroller verbinden wir dieses Ausgangssignal mit den Enable/Disable-Eingängen der Leistungsendstufen. Falls Sie bei Ihrer Steuerelektronik etwas anders realisiert haben, brauchen Sie hier nicht weiterzulesen. Sie können mit dem nächsten Kapitel weiter arbeiten.
Das Bild 4.18 zeigt Ihnen das Arbeitsmenü für das Kontrollieren des Enable/Disable.

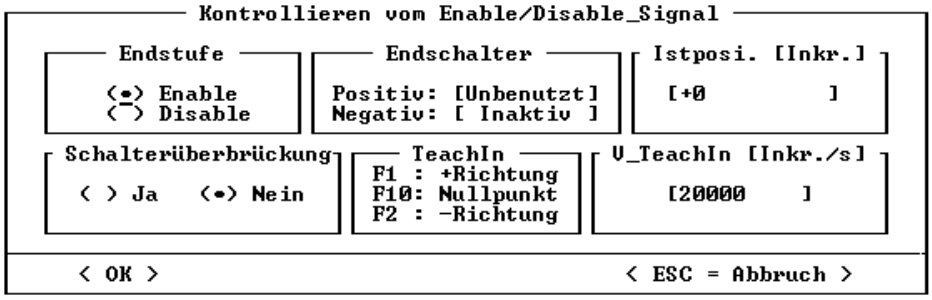


Bild 4.18: Arbeitsmenü für das Kontrollieren des Enable/Disable

Das Kontrollieren vom Enable/Disable ist sehr einfach. Mit den Funktionstasten F1 und F2 können Sie die Achse in beide Richtungen bewegen. Die Hardware-Endlageschalter sollten inaktiv sein. Zuerst wählen Sie *Enable* im Feld „Endstufe“. Im fehlerfreien Fall können Sie mit den Funktionstasten die Achse weiterhin fahren. Danach wählen Sie *Disable*. Im fehlerfreien Fall können Sie die Achse jetzt nicht mehr bewegen.

4.13 Das Untermenü *Totzeit*

Intern braucht der Treiber die Totzeit der einzelnen Achsen, um den Nachlauffehler zu überwachen. In diesem Untermenü können Sie die Totzeit der Achsen sehr schnell und einfach ermitteln.

Das Bild 4.19 zeigt Ihnen das Arbeitsmenü für das Ermitteln der Totzeit.

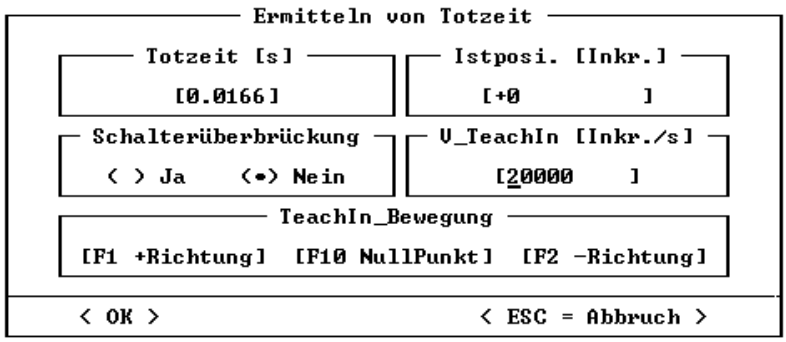


Bild 4.19: Arbeitsmenü für das Ermitteln der Totzeit

Mit den Funktionstasten F1 und F2 können Sie die Achse bewegen. Beim Starten jeder Bewegung wird die Totzeit automatisch berechnet und angezeigt. Beachten Sie, dass nur der Bewegungsstart für die Berechnung der Totzeit wichtig ist.
Aufgrund der Asymmetrie jeder Achse sollten Sie die Totzeit in beide Richtungen ermitteln. Sie haben dann noch die Möglichkeit, die Totzeit zu editieren. Dabei sollten Sie den größeren Wert der gerade ermittelten Totzeiten eingeben.

5 Das Hauptmenü *Info*

Hier können Sie sehen, welche Treiberversion Sie haben müssen, wenn Sie mit diesem Einstellprogramm arbeiten wollen. Außerdem finden Sie hier unsere Anschrift, die für Sie in manchen Situationen nützlich sein kann.

6 Mögliche Fehler beim Parameter einstellen

Bei der Inbetriebnahme können verschiedene Fehler auftreten. In diesem Abschnitt wollen wir diese Fehler und ihre Ursachen auflisten.

Dateischreibfehler
Dieser Fehler tritt nur beim Speichern der Initialisierungsdatei auf. Der Grund dafür liegt bei dem Datenträger, der entweder defekt, voll oder schreibgeschützt ist.

Dateinamenfehler
Diesen Fehler können Sie beim Öffnen oder Speichern einer Initialisierungsdatei bekommen. Die Namen der Initialisierungsdateien sind Namen nach der DOS-Konvention. Alle Buchstaben von „a“ bis „z“ werden in den Namen akzeptiert. Es gibt keine Unterscheidung zwischen Groß- und Kleinbuchstaben. Die Zahlen von „0“ bis „9“ sowie die Zeichen „-“ und „_“ werden als normale Buchstaben akzeptiert. Außerdem müssen die Dateinamen die Erweiterung „INI“ haben.

Dateiformatfehler
Sie bekommen diesen Fehler, wenn Sie versuchen, eine Datei zu öffnen, die keine Initialisierungsdatei ist oder die eine falsche Versionsnummer hat.

Druckerfehler
Diesen Fehler bekommen Sie nur, wenn Sie versuchen, die Initialisierungsdatei zu drucken. Wenn der Drucker nicht bereit ist oder wenn die Verbindung zum Drucker nicht vorhanden ist, bekommen Sie logischerweise diesen Fehler.

Der Rechner ist nicht graphikfähig
Dieser Fehler kann nur auftreten, wenn Sie ein von den Untermenüs *V_Regler*, *Posi._Regler* oder *Rampe* benutzen wollen. Hier muss das Einstellprogramm den Graphik-Modus der Videokarte in Ihrem Rechner benutzen, um Ist-Geschwindigkeitsverläufe aufzuzeichnen. Wenn die Videokarte keinen Graphik-Modus unterstützt, bekommen Sie diesen Fehler. Im Normalfall darf dieser Fehler gar nicht auftreten, weil alle gängigen Videokarten auf dem Markt grafikfähig sind. Die Ursache für diesen Fehler liegt am meistens darin, dass Sie den falschen Treiber für die Videokarte installiert haben.

Falsche Basisadresse oder defekter Timer
Dieser Fehler tritt nur bei der Auswahl des Untermenüs *Einsteckkarte* auf.
In den meisten Fällen deutet diese Fehlermeldung auf eine falsche Basisadresse hin. Sie müssen mit dem Konfigurationsprogramm PARKON die richtige Basisadresse eingeben. Sehr selten wird diese Fehlermeldung durch den defekten Timer auf der PC-Einsteckkarte ausgelöst. In diesem Fall müssen Sie den Timerbaustein umtauschen (Hersteller INTEL).

Defekter Achscontroller
Diese Fehlermeldung kommt nur bei der Auswahl des Untermenüs *Einsteckkarte*. Die Achse, deren Achscontroller defekt ist, wird angezeigt. Hier ist der Achscontroller entweder nicht vorhanden oder defekt. In den meisten Fällen wird diese Fehlermeldung durch die falsche Angabe der Achsanzahl ausgelöst. Wenn Sie z. B. die Achsanzahl gleich 4 angeben obwohl die PC-Einsteckkarte nur für 3 Achsen mit 3 Achscontrollern bestückt ist, bekommen Sie diesen Fehler.

Fehlerhafte IRQ-Nummer
Unsere Treibersoftware für die PC-Einsteckkarte benötigt intern einen Hardware-Interrupt. Sie können zwischen IRQ10 und IRQ11 auswählen. Die Auswahl muss mit der Jumper-Einstellung auf der PC-Einsteckkarte übereinstimmen und der gewählte Interrupt muss noch frei sein (siehe Handbuch *isel*-Servosteuerkarte UPMV 4/12). Sonst bekommen Sie diese Fehlermeldung.

Wichtiger Hinweis!

Lesen Sie vor dem Öffnen der Verpackung bitte die Bestimmungen des umseitigen Lizenzvertrages genau durch.
Falls Sie mit dem Lizenzvertrag oder mit Teilen des Lizenzvertrages nicht einverstanden sind, so senden Sie das Produkt gegen Erstattung des an ***isel****automation* entrichteten Kaufpreises in ungeöffnetem Zustand zurück.
Mit dem Öffnen der Verpackung erklären Sie sich mit den Bedingungen des Lizenzvertrages einverstanden.

Lizenzvertrag

1 Einräumung einer Lizenz

***isel**automation* GmbH & Co.KG gewährt Ihnen das Recht, Kopien des beiliegenden Programmes bzw. Programmpaketes (die „Software“) auf einem oder mehreren Computern zu benutzen - solange die Nutzung im Eigengebrauch stattfindet. Die Software wird benutzt, wenn sie in den temporären Speicher (RAM) oder in einem permanenten Speicher (Festplatte, Streamer, Diskette, CD-ROM, ...) installiert wird.

2 Erweiterung der Lizenz

Falls die Software ein Sprachprodukt ist (z. B. Interpreter zur freien Programmierung), so haben Sie das Recht, die für den Betrieb der NC-Steuerdateien erforderlichen Runtime-Module der Software zu vervielfältigen und zu verbreiten, unter folgenden Voraussetzungen:

a) Sie vertreiben die Runtime-Module nur als Teil Ihres Software-Produktes und als Teil desselben.
b) Sie verwenden bei der Vermarktung Ihres Produkts weder den Namen noch das Logo noch andere Kennzeichen von ***isel**automation* GmbH & Co.KG.
c) Sie nehmen die Copyright-Meldung von ***isel**automation* GmbH & Co.KG für die Software als Teil der Bereitschaftsmeldung in Ihr Produkt auf. Falls Sie Runtime-Module von ***isel**automation* GmbH & Co.KG benutzen, so dürfen deren Copyright-Meldungen weder entfernt noch modifiziert werden.
d) Sie erklären sich einverstanden, ***isel**automation* GmbH & Co.KG bezüglich aller Ansprüche oder Rechtsstreitigkeiten, die aufgrund des Gebrauchs oder der Verbreitung Ihres Produkts entstehen können, freizustellen, schadlos zu halten und gegen solche Ansprüche zu verteidigen.

Runtime-Module sind die Dateien der Software, für die in den schriftlichen Unterlagen ausdrücklich angegeben ist, daß sie erforderlich sind für:

- den Betrieb der Hardware
- den Ablauf Ihres NC-Steuerungsprogramms

Die Runtime-Module sind auf die speicherresidenten Steuerungsprogramme (Treiber) für die Maschinensteuerung, die Objekt-Dateien für die Bahndatenerstellung und Interpreter-Module für NC-Quelldateien, deren Gebrauch und Weitergabe ausdrücklich gestattet ist, beschränkt.

3 Urheberrecht

Die Software ist Eigentum von ***isel**automation* GmbH & Co.KG und durch Urheberrechtsgesetze und nationale Rechtsvorschriften gegen Kopieren geschützt. Für den Eigengebrauch und für Sicherungs- und Archivierungszwecke dürfen Sie Kopien der Software anfertigen. Weder die Software noch die Handbücher oder Teile davon dürfen kopiert oder in anderer Form vervielfältigt und an Dritte weitergegeben oder Dritten in anderer Form zugänglich gemacht werden. Zurückentwicklung (Reverse Engineering), Dekompilieren und Disassemblieren der Software durch Sie oder durch Dritte ist nicht gestattet.

4 Einschränkung der Haftung und Ansprüche des Kunden

***isel**automation* ist nicht für Schäden (uneingeschränkt eingeschlossen sind Schäden aus entgangenem Gewinn, Datenverlust, Betriebsunterbrechungen oder aus anderem finanziellem Verlust) ersatzpflichtig, die aufgrund der Benutzung der Software oder der fehlerhaften Benutzung der Software entstehen. In jedem Fall ist die Haftung von ***isel**automation* GmbH & Co.KG auf den Betrag beschränkt, den Sie an ***isel**automation* GmbH & Co.KG für die Software bezahlt haben.

Die gesamte Haftung von ***isel**automation* GmbH & Co.KG und Ihr alleiniger Anspruch besteht nach Wahl von ***isel**automation* GmbH & Co.KG entweder in der Rückerstattung des bezahlten Preises oder in der Reparatur oder dem Ersatz der Software oder der Hardware. Dies gilt nicht, wenn der Ausfall der Software auf fehlerhafte Anwendung oder Unfall zurückzuführen ist.