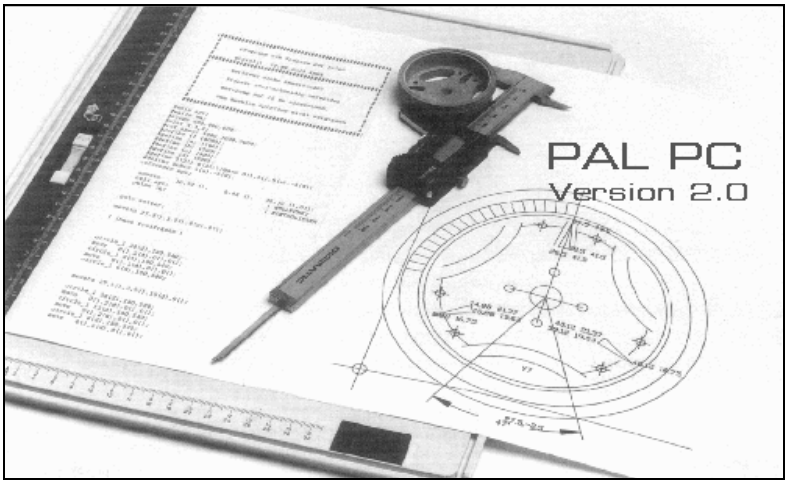
iselautomation KG



PAL-PC 2.0

Programmiervorschrift

isel®

www.isel.com

Zu dieser Anleitung:

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

Achtung:	Beispiel:	Hinweis:	Information:
			

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

Hersteller: Fa. ***isel***automation KG
Bürgermeister-Ebert-Straße 40
D-36124 Eichenzell

Tel.: (06659) 981-0
Fax: (06659) 981-776
email: automation@isel.com
http://www.isel.com

Stand: 06/2004

Inhalt

INHALT **3**

1 EINFÜHRUNG **4**

1.1 PRODUKT 4
1.2 UNTERSTÜTZTE STEUERUNGEN 5
1.3 FUNKTIONEN VON PAL-PC 5

2 SPRACHELEMENTE VON PAL-PC **6**

2.1 ÜBERBLICK 6
2.2 PROGRAMMAUFBAU EINES PAL-PC-PROGRAMMS 6
2.3 DER EDITOR 7

3 BEFEHLE PAL-PC **9**

3.1 BEFEHLE DES DEKLARATIONSBLOCKS 9
3.1.1 Übersicht Befehle des Deklarationsblocks 9
3.1.2 Der Befehl #axis 10
3.1.3 Der Befehl #steps 11
3.1.4 Der Befehl #units 12
3.1.5 Der Befehl #elev 13
3.1.6 Der Befehl #define 14
3.1.7 Der Befehl #redefine 17
3.1.8 Der Befehl #start 18
3.1.9 Der Befehl #ref_speed 19
3.1.10 Der Befehl #include 20
3.1.11 Der Befehl Kommentar 21
3.1.12 Der Befehl #GN 22
3.1.13 Der Befehl #input 23
3.2 DER ANWEISUNGSTEIL 24
3.2.1 Übersicht Befehle des Anweisungsteils 24
3.2.2 Der Befehl label 25
3.2.3 Der Befehl move 26
3.2.4 Der Befehl moveto 28
3.2.5 Der Befehl movep 30
3.2.6 Der Befehl send 31
3.2.7 Der Befehl wait 32
3.2.8 Der Befehl loop 34
3.2.9 Der Befehl port und pulse 35
3.2.10 Der Befehl time und delay 36
3.2.11 Der Befehl reference 37
3.2.12 Der Befehl tell 38
3.2.13 Der Befehl stop 39
3.2.14 Der Befehl line 40
3.1.15 Der Befehl repeat ... until 41
3.2.16 Der Befehl goto 42
3.2.17 Der Befehl null 43
3.2.18 Der Befehl on_key 44
3.2.19 Der Befehl on_port 45
3.2.20 Der Befehl set_port 46
3.2.21 Der Befehl 3D-Linearinterpolation 47
3.2.22 Der Befehl circle 48

INDEX **50**

1 Einführung

1.1 Produkt

Produktbeschreibung: **PAL-PC** ist ein Programmiersystem für die isel-Interfacekarten-Serie zur Lösung von einfachen Prozesssteuerungen. Mit PAL-PC können maximal 3 Achsen angesteuert werden.
PAL-PC kann **direkt** (DNC-Mode) oder im **Speicherbetrieb** (CNC-Mode) ausgeführt werden. Damit sind sowohl Applikationen im **Stand-Alone-Betrieb** als auch **Anwendungen mit Steuer-PC** möglich.
Unter Nutzung der CNC-Erweiterung der IMC4-Steuerung können auch Maschinen der CPM-Serie/ GFM 4433 autark betrieben werden.
PAL-PC unter Windows ist die **Nachfolge-Software** von PAL-PC unter DOS.
Sie enthält den **gesamten** Funktionsumfang der DOS-Version.

Die Benutzeroberfläche ist so gestaltet, dass die wichtigsten Programmfunktionen über die Buttons der Symbolleiste ausgelöst werden können.

PAL-PC verfügt über einen **integrierten Editor und Compiler**. Übliche Editorfunktionen wie „Suchen“ und „Ersetzen“, „Kopieren“ und „Einfügen“ sowie Formatfunktionen zur Farb- und Schriftgestaltung ermöglichen eine komfortable und schnelle Programmerstellung bis hin zum fehlerfrei übersetzten Anwenderprogramm.

Die Hardwareoption Battery-Backup gewährleistet eine dauerhafte Programmspeicherung auch nach Abschalten der Anlage.
Über eine **Memory Card** kann das übersetzte Anwenderprogramm gesichert und direkt wieder in den Speicher des Controllers zurückgeladen werden.

Pal-PC läuft unter den Betriebssystemen 98, Windows 2000 und Windows XP.

DNC- und CNC-Mode : Im **DNC-Mode** erfolgt die Übertragung des Anwenderprogramms an die Steuerung befehlsweise / segmentweise mit direkter Ausführung. Startbar ist das Programm in diesem Modus nur mit angeschlossenem Steuer-PC (Direktmode).

Im **CNC-Mode** wird nach Übertragung (download) des Anwenderprogramms an die Steuerung (Zielcontroller) das Programm in der Interfacekarte gespeichert. Startbar ist es direkt am Controller bzw. an der Maschine (Speicherbetrieb bzw. Stand-Alone-Betrieb). Der PC ist nur erforderlich für Programmerstellung, -testung und download.

1.2 Unterstützte Steuerungen

Steuerungen: Folgende Steuerungen werden unterstützt:

- Interfacekarte V5.c
- Kompaktanlage EP10905
- Einachssteuerung IT116
- 4-Achs Schrittmotor-Controller CSI 464

Maschinen: Folgende Maschinen sind lauffähig mit PAL-PC:

- für alle Maschinen der CPM-Familie (IMC4 mit CNC-Erweiterung)
- GFM 4433 (IMC4 mit CNC-Erweiterung)

1.3 Funktionen von PAL-PC

Verwaltung mit PAL-PC:

- 1 Achssystem mit max. 4 Achsen (X, Y, Z, A)
- Kommunikation mit anderen Rechnersystemen über serielle Schnittstelle

- Direktausführung (DNC-Mode) und Speicherbetrieb (CNC-Mode)
- Memory Card für Speicherung der CNC-Programme auf ein externes Speichermedium
- Battery-Backup zur Programmspeicherung nach Abschalten der Anlage
- Übertragungsraten von 2400 Baud bis 19200 Baud
- Start und Durchführung des Programms ohne Steuerrechner möglich

- Teach-, Referenzgeschwindigkeit einstellbar
- Teach-In
- Werkstücknullpunkt setzen
- Referenzpunktfahrt

Programmfunktionen:

- Integrierter Editor zur Programmerstellung
- Compiler zur Übersetzung des Anwenderprogramms
- Wegbefehle zur relativen und absoluten Positionierung
- 2D-Interpolation umschaltbar auf 3D-Interpolation
- Auswertung von Ein- und Ausgangssignalen zur Prozesssteuerung
- Schleifen zur Wiederholung von Anweisungsblöcken, unbedingte und bedingte Verzweigungen, Zeitverzögerung

2 Sprachelemente von PAL-PC

2.1 Überblick

Überblick:

PAL-PC ist eine Programmiersprache, die sich fest **definierter Tokens** in Verbindung mit befehlsspezifischen **Parametern** bedient. Tokens werden in der Regel klein geschrieben, können Buchstaben, Zahlen und den Unterstrich enthalten.
Eigendefinierte Symbole sollten groß geschrieben werden, damit keine Konflikte mit diesen Standardbezeichnern (Befehlsvorrat) von PAL-PC entstehen.
Das Programm PAL-PC unterscheidet zwischen den Steueranweisungen und den speicherbaren Instruktionen.
Die Steueranweisungen stehen im Deklarationsblock des Programms und werden mit einem vorangestellten Rautezeichen # gekennzeichnet.
Die speicherbaren Instruktionen enthalten die ausführbaren Befehle. Sie werden an die Interfacekarte bzw. angeschlossene Steuerung übertragen und gespeichert . Ist die Option Battery-Backup installiert, bleiben die gespeicherten Befehle auch nach dem Ausschalten erhalten.
Nach dem Einschalten der Maschine kann dieses gespeicherte Programm ohne Verbindung zu einem übergeordneten Rechner mit der Starttaste an der Anlage gestartet werden.

In PAL-PC sind für den Deklarationsblock und Anweisungsteil Befehle vordefiniert.

siehe: 3.1.1 Übersicht Befehle des Deklarationsblocks auf Seite 9
siehe: 3.2.1 Übersicht Befehle des Anweisungsteils auf Seite 24

Sie sollten die Verwendung der Befehlsbezeichnung für eigene Definitionen meiden, da Ihnen sonst die Originalfunktionen nicht mehr zur Verfügung stehen.

2.2 Programmaufbau eines PAL-PC-Programms

Grundsätzliches:

Ein PAL-PC-Programm besteht aus dem optionalen Deklarationsteil, der Voreinstellungen zur angeschlossenen Mechanik und weitere Parameter enthält, die für die Befehle im anschließenden Anweisungsteil des Programms wichtige Daten liefern.
Die Befehle des Deklarationsteils bestehen aus dem Schlüsselwort mit vorangestelltem Rautezeichen **#** und den Parameterdefinitionen.
Der Anweisungsteils mit den speicherbaren Befehlen wird in der Regel mit dem Befehl **#input** eingeleitet.
Diese Befehle bestehen aus dem Befehlswort, gefolgt von den befehlsspezifischen Parametern. Der Anweisungsteils enthält u. a. die Bewegungsbefehle, Schleifen, direkte Sprünge, Tastaturabfragen, die im Anwenderprogramm zur Erfüllung des gewünschten Automatisierungsablaufs erforderlich sind.
Kommentare können in beiden Programmabschnitten zur besseren Transparenz des Anwenderprogramms eingefügt werden.
Zur Kennzeichnung des Programmendes wird der Befehl **stop** verwendet, der Befehl **#start** steht dafür, dass ein an die Steuerung übergebener Datenblock nach Übertragung gestartet wird.

Satzaufbau:

Ein Satz im PAL-PC besteht aus in der Regel aus **einem** Befehlswort, gefolgt von den zugeordneten Parametern. In diesem Fall muss die Befehlszeile nicht zwingend wie bisher mit einem Semikolon abgeschlossen werden.
Zugelassen ist ebenfalls die Verwendung von mehreren Befehlsworten in einem Satz, dann ist unbedingt das Ende jeder Befehlsdeklaration mit Semikolon zu markieren.
Einem Satz kann ein Label, das als Abgrenzung zum Befehlswort den **Doppelpunkt** enthält, vorangestellt sein.
Welche Parameter dem Befehlswort folgen müssen oder ob der Befehl nur eine Umschaltfunktion in einen bestimmtem Modus erfüllt, ist im Kapitel 3 Befehle PAL-PC beschrieben.
siehe auch:
3.2.4 Befehl moveto (Befehl mit Parameterblock)
3.2.21 Befehl set3d on/set3d off (Befehl mit Schalterfunktion)

Hinweis:

In einigen Befehlen wurden hinsichtlich Syntax Vereinfachungen vorgenommen. Diese Vereinfachungen sollen einem besseren Verständnis und plausibleren Parametereingaben dienen.
Um eine komplette Kompatibilität der Programme PAL-PC unter DOS zu gewährleisten, sind alle bereits bestehenden Syntaxvorschriften weiterhin gültig.
siehe auch: 3.2.20 Der Befehl set_port auf Seite 46

2.3 Der Editor

Quelldatei:

Im Programmsystem PAL-PC kann der Anwender seine Programme (z.B. Programme zur Prozesssteuerung oder Bearbeitungsprogramme) erstellen, nachdem entsprechend der technologischen Aufgabenstellung ein Algorithmus zur Umsetzung dieser Aufgabenstellung gefunden worden ist. Der gefundene Algorithmus wird in die Programmtextstruktur (**Sequenz, Schleife, Verzweigung**) umgesetzt. Diese Art der Programmierung wird als **textuelle Programmierung** bezeichnet.

Die Quelldatei wird als Textdatei mit dem integrierten Editor des PAL-PC erstellt. Jede Quelldatei besitzt die Erweiterung *.txt.
Ebenso kann eine außerhalb von PAL-PC erstellte Textdatei (z. B. mit Wordpad) in PAL-PC geladen und weiterverarbeitet werden.

Bei der **Erstellung** des Programmtextes müssen bestimmte **Regeln** eingehalten werden. Diese **Regeln** werden bei Sprachen allgemein als **Grammatik** bezeichnet.
Die **Grammatik definiert Syntax** (*welche Zusammenfügung von Worten ergibt einen gültigen Satz ?*) **und Semantik** (*was bedeutet dieser Satz?*) der Programmiersprache. Die gültige Grammatik ist im Kapitel 3 in diesem Handbuch ausführlich beschrieben und mit Beispielen erklärt.

Anwenderdatei:

Die Anwenderdatei entsteht nach einem Übersetzungslauf und wird als Datei mit der Erweiterung *.out im Verzeichnis, das die Quelldatei enthält, gespeichert.
Die Anwenderdatei wird nur dann erstellt, wenn bei diesem Übersetzungslauf keine syntaktischen Fehler festgestellt werden.
Eine entsprechende Anzeige und ein Fehlerprotokoll weisen Sie auf syntaktische Fehler hin.
Ein Beispiel für ein syntaktisch fehlerfreies Quellprogramm zeigt das folgende Programm:

Beispiel:

```
************************************************************************
/ Datei:          test
/
/ erstellt:       06/09/2003, iselautomation KG
/
/ ************************************************************************
/ Deklarationsteil
#axis x;          { Achsenwahl:    X-Achse}
#reference x;     { Referenzfahrt: X-Achse-}
#units mm;        { Maßeinheit setzen; mm ist default}
#input;

/ Beginn Anweisungsteil, nachfolgende Befehle werden an die Steuerung
/ übertragen und gespeichert
repeat   ; Endlosschleife ...
  reference x;

  repeat
    set_port A1,1=1;    / setzen Bit1 am Ausgangsport A1
    delay 1;
    set_port A1,1=0;    / löschen Bit1 am Ausgangsport A1
  until 2;

  moverel 100(500);       { X-Achse 100 mm verfahren mit 500 Hz }
  moverel -100(9000);       { X-Achse -100mm verfahren mit 9000 Hz }
  delay 10;                 { 1 sec warten }

until 0;                    { Schleifenende }

stop.                       { Programmende }
#start;                     { Starte Ausführung }
```

3 Befehle PAL-PC

3.1 Befehle des Deklarationsblocks

Voreinstellungen zum Programm:

Der Deklarationsteil dient dazu, dem Compiler die Voreinstellungen zu geben, die für die Bearbeitung notwendig sind. Diese wichtigen Parameter sind u. a. Anzahl der angeschlossenen Achsen (max. 3 Achsen), Bearbeitungs- und Referenzgeschwindigkeit der Achsen, Spindelsteigung, Festlegung der verwendeten Maßeinheit für alle Wegbefehle.
Der Deklarationsblock kann entfallen, allerdings werden dann die Voreinstellungen wirksam.
Defaultwerte finden Sie in der Beschreibung der einzelnen Befehlen.

3.1.1 Übersicht Befehle des Deklarationsblocks

Befehl	Inhalt
#axis	Zuweisung der angeschlossenen Achsen (Kap. 3.1.2 auf Seite 10)
#define	Textersatz definieren (Kap. 3.1.6 auf Seite 14)
#elev	Spindelsteigung definieren (Kap. 3.1.5 auf Seite 13)
#GN	Gerätenummer definieren (Kap. 3.1.12 auf Seite 22)
#include	Datei einfügen (Kap. 3.1.10 auf Seite 20)
#input	Speichermodus setzen (Kap. 3.1.13 auf Seite 23)
#redefine	Textersatz neu definieren (Kap. 3.1.7 auf Seite 17)
#ref_speed	Referenzgeschwindigkeit setzen (Kap. 3.1.9 auf Seite 19)
#start	Ausführung starten (Kap. 3.1.8 auf Seite 18)
#steps	Schrittzahl/Umdrehung angeben (Kap. 3.1.3 auf Seite 11)
#units	Maßeinheit definieren (Kap. 3.1.4 auf Seite 12)
{..} oder /	Kommentare einfügen (Kap. 3.1.11 auf Seite 21)

3.1.2 Der Befehl #axis

#axis	Achsenwahl

Syntax: **#axis** [Achsen];

Erklärung: Achsen {x, y, z}

Dieser Befehl muss als erster Befehl im Programm stehen.
Durch Übergeben der Achsenanzahl wird die Prozessorkarte neu initialisiert.
Dabei wird der Datenspeicher gelöscht und zur Speicheroptimierung entsprechend der Anzahl der Achsen neu eingeteilt.

Voreingestellt ist die Achskonfiguration X, Y und Z.

Beispiel:

#axis x; {nur X-Achse angeschlossen}
#axis xz; {X- und Z-Achse angeschlossen}

Hinweis:

Vom Bediener kann die Definition der Achsen über das **Menü** Einstellungen - Allgemein **...** vorgenommen werden.
Die Anzahl der definierten Achsen sollte mit der Anzahl der angesteuerten Achsen übereinstimmen.
Beispielsweise kann die separate Wahl der x-Achse bei drei angeschlossenen Achsen der IMC4 zu undefinierten Verfahrbewegungen führen.

Achtung:

Der Befehl „Achsenanzahl setzen“ löscht alle im RAM vorhandenen Daten, auch wenn durch die integrierte Option „Memory Back-up“ die Daten nach Wegfall der Versorgungsspannung im RAM-Speicher der Prozessorkarte gespeichert waren.

3.1.3 Der Befehl #steps

#steps	Schrittzahl/Umdrehung

Syntax: **#steps** [Schrittzahl X],[Schrittzahl Y],[Schrittzahl Z]];

Erklärung: [Schrittzahl] Schrittzahl/Umdrehung der angeschlossenen Achsen

Es werden für jede Achse durch Kommata getrennt, die Schrittzahl/Umdrehung für den Motor der jeweiligen Achse angegeben.

Die Schrittzahl pro Umdrehung gibt dem Compiler den notwendigen Umrechnungsfaktor an, um die Wandlung der Arbeitseinheit in Schrittmotorschritte vornehmen zu können.

Beispiel: Beispiel 1:

Hat der Motor der X-Achse 400 Schritte/Umdrehung und die dazugehörige Spindel eine Steigung von 4 mm, so ergibt sich folgende Zuordnung:

Allgemeingültige Formel:

$$\frac{\text{Motorschritte/Umdrehung}}{\text{Spindelsteigung(mm)}} = \text{Schritte/Millimeter}$$

$$\frac{400\ \text{Schritte}}{4\ \text{mm}} = 100\ \text{Schritte}/\text{mm}$$

Ergebnis: **#steps** 100;

Bei Verwendung eines Getriebeschrittmotors ist die Untersetzung mit der Anzahl der Motorschritte/Umdrehung zu multiplizieren.

Beispiel 2:

Der Motor der X-Achse hat 400 Schritte/Umdrehung und ein 1:9-Getriebe, die dazugehörige Spindelsteigung beträgt 4 mm:

Allgemeingültige Formel:

$$\frac{\text{Motorschritte}/\text{Umdrehung} \times \text{Untersetzung}}{\text{Spindelsteigung}} = \text{Schritte}/\text{millimeter}$$

$$\frac{400\ \text{Schritte} \times 9}{4\ \text{mm}} = 900\ \text{Schritte}/\text{mm}$$

Ergebnis: **#steps** 900;

Der Befehl muss nach der Achsenwahl verwendet werden, da er, falls vorher keine Achsenwahl stattgefunden hat, die Achsenanzahl auf XYZ initialisiert.

3.1.4 Der Befehl #units

#units	Maßeinheit festlegen

Syntax: **#units** [Einheit];

Erklärung: Angabe der Maßeinheit für Verfahrbewegungen.
Zulässig sind:

#units mm;
#units cm;
#units zoll;
#units zoll/10;
#units zoll/20;
#units inch;
#units inch/10;
#units inch/20;

Ohne Angabe einer Arbeitseinheit wird ,,mm" als Maßeinheit angenommen. Zoll/10 und Zoll/20 sind die gebräuchlichen Maßeinheiten für Leiterplatten, da diese im Zoll-Raster gebohrt werden.

3.1.5 Der Befehl #elev

#elev	Spindelsteigung definieren

Syntax: **#elev** [Steigung X],[Steigung Y],[Steigung Z];

Erklärung: [Steigung] Gibt die Spindelsteigung der X-, Y- und Z-Achse an

Falls mit **#units** nicht anders vereinbart, ist die Maßeinheit [mm].
Um eine korrekte Umrechnung der Arbeitseinheit in Schrittmotorschritte zu ermöglichen, muss die Spindelsteigung der angeschlossenen Achsen übergeben werden.

Beispiel: Beispiel 1:

Eine Anlage hat 4 mm Spindeln in der x- und Y-Achse sowie eine 2,5-mm-Spindel in der Z-Achse. Der zugehörige Befehl lautet dann:

#elev 4, 4, 2.5;

Beispiel 2:

Bei einer Anlage mit einer 2-mm-Spindel in der X-Achse und 4-mm-Spindel in der Y-Achse lautet der Befehl (eine Z-Achse ist nicht vorhanden):

#elev 2, 4;

Falls kein Spindelsteigungsbefehl verwendet wird, wird für alle angeschlossenen Achsen eine Spindelsteigung von 4 mm angenommen.

3.1.6 Der Befehl #define

#define	Textersatz definieren

Syntax: **#define** [(Zeichenkette) (Anweisung)\;...\;(Anweisung)];

Erklärung:

[Zeichenkette] Gibt den Definitionsname der folgenden Texte/Anweisungen an. Zur Bildung des Namens sind alle Buchstaben, Ziffern und das Zeichen „_“ zugelassen.
Beginnen muss der Name stets mit einem Buchstabe. Wird ein Trennzeichen (Leerzeichen oder Tabulator) erkannt, wird dies als Ende des Namens interpretiert.

[Anweisung] Gibt die gewünschten Texte/Anweisungen für den definierten Namen an.
Werden mehrere Anweisungen, die mit einem Semikolon abschließen, in einem Textersatz definiert, muss vor jedem Semikolon, das den Textersatz noch nicht abschließt, ein sogenanntes Fluchtzeichen „\“ geschrieben werden.

Hinweis:

Bitte beachten Sie: Der gewählte Definitionsname darf nicht gleich sein mit einer definierten Sprungadresse und darf auch nicht als Zeichenfolge in einer Sprungadresse enthalten sein.
Wird zur Definition wegen der rationellen Schreibweise nur 1 Zeichen verwendet, benutzen Sie bitte Sonderzeichen wie z. B. &, §, $ oder auch Buchstaben eingeschlossen in Klammern z. B. (V), (S).
Es ist auf Groß- und Kleinschreibung zu achten, da dies ebenfalls ein Unterscheidungsmerkmal darstellt.

Im PAL-PC ist es erlaubt, Textersatzsymbole zu definieren. Textersatz bedeutet, dass Sie häufig gebrauchten Text oder eine oder mehrere Anweisungsfolge(n) einmal definieren und dann im Text unter eben diesem Namen benutzen können. Textersatz kann sehr einfach sein (s. Beispiel 1) oder auch einen komplexen Anweisungsblock (s. Beispiel 2) mit Wiederholungen und vielen Einzelanweisungen enthalten.
Textersatz als komplexer Anweisungsblock hat eine dem Unterprogramm analoge Funktion.
Der gesamte Befehl muss mit einem Semikolon abgeschlossen werden.

Beispiel:

Beispiel 1: - Definition einfacher Anweisungen

```
#define () (3000);
#define & 0(21),0(21);
#define bohre 20(1000),-20(9000);
```

Die erste Definition wählt für die runden Klammern **()** ohne Angabe eine Geschwindigkeitsangabe von 3000 Hertz.

Dem Zeichen **&** wird keine Bewegung für die Z-Achse zugeordnet. Je nachdem an welcher Stelle im Befehl diese Definition eingefügt wird, kann sie z. B. für die Angabe der Z-Koordinate im move-Befehl stehen. Die Mindestangabe für die Geschwindigkeit ist 21.
siehe 3.2.3 Der Befehl move auf Seite 26

Die dritte Definition **bohre** wird anstelle von Z-Koordinaten verwendet, um die Bohrtiefe konstant für das gesamte Programm einstellen zu können. Unter Verwendung obiger Definitionen kann folgender Programmabschnitt erstellt werden:

```
/verfahre X und Y jeweils  20 mm
move 20(),20(),&;
/ verfahre X= 2mm, Y= 5 mm, bohre ein Loch mit einer Tiefe von 20 mm
move 2(),5(5000),bohre;
```

Während des Übersetzens würde der Compiler die obige Anweisungsfolge dann in folgende Befehle übersetzen:

```
move 20(3000),20(3000),0(21),0(21);
move 2(3000),5(5000),20(1000),-20(9000);
```

Beispiel 2: - Definition von Anweisungsblöcken

Da der Befehl ,,Textersatz" als Endezeichen ein Semikolon benötigt, muss das Auftreten eines Semikolons im Text speziell markiert werden. Hierzu wird direkt vor dem Semikolon ein spezielles Zeichen eingefügt, das sogenannte Fluchtzeichen. Als Beispiel wird die Definition einer 14-poligen IC-Fassung gewählt:

```
#define DIL14
  repeat
     move 1(),0(),bohre\;
  until 7\;
  move 1(),3(),&\;
  repeat
     move 1(),0(),bohre\;
  until 7;
```

Bitte beachten Sie, dass hinter der letzten Anweisung vor dem Semikolon kein Fluchtzeichen steht. Dieses Semikolon schließt die Definition ab.

Das oben definierte Symbol greift auf die in Beispiel 1 definierten Symbole **&**, **()** und **bohre** zurück. Diese Symbole müssen vor der Definition von **DIL14** vereinbart werden, sonst wird nach dem Compilerlauf Fehler angezeigt.

Sie können das definierte Symbol **DIL14** nun in eigenen Programmen folgendermaßen verwenden:

```
...
move 20(),30(),&;          { Anfangspunkt anfahren}
repeat                     {zwei Fassungen ...}
   DIL14;                  { DIL-14 bohren }
   move 1(),20(),&;        { Anfangspunkt nächste Fassung }
until 2;                   { und nächste Fassung }
```

Beschränkung: Beachten Sie, dass der Compiler Groß- und Kleinschreibung unterscheidet. Die Verwendung des Symbols ,,dil14" würde daher einen Fehler auslösen.
Die Anzahl der Definitionen, die maximal gespeichert werden können, beträgt 500.
Die Länge einer Definition darf maximal 250 Zeichen betragen. Die Länge der Zeile, in der die Definition ersetzt wird, darf 255 Zeichen nicht überschreiten.

3.1.7 Der Befehl #redefine

#redefine	Neudefinition

Syntax: **#redefine** *[(Zeichenkette) (Anweisung)\;...\;(Anweisung)];

Erklärung:

[Zeichenkette]	Gibt den Definitionsname der folgenden Texte/Anweisungen an. Zur Bildung des Namens sind alle Buchstaben, Ziffern und das Zeichen „_“ zugelassen. Beginnen muss der Name stets mit einem Buchstabe. Wird ein Trennzeichen (Leerzeichen oder Tabulator) erkannt, wird dies als Ende des Namens interpretiert.
[Anweisung]	Gibt die gewünschten Texte/Anweisungen für den definierten Namen an. Werden mehrere Anweisungen, die mit einem Semikolon abschließen, in einem Textersatz definiert, muss vor jedem Semikolon, das den Textersatz noch nicht abschließt, ein sogenanntes Fluchtzeichen „\“ geschrieben werden.

Hinweis:

Bitte beachten Sie: Der gewählte Definitionsname darf nicht gleich sein mit einer definierten Sprungadresse und darf auch nicht als Zeichenfolge in einer Sprungadresse enthalten sein.
Wird zur Definition wegen der rationellen Schreibweise nur 1 Zeichen verwendet, benutzen Sie bitte Sonderzeichen wie z. B. &, §, $ oder auch Buchstaben eingeschlossen in Klammern z. B. (V), (S).
Es ist auf Groß- und Kleinschreibung zu achten, da dies ebenfalls ein Unterscheidungsmerkmal darstellt.

Um Geschwindigkeitsdefinitionen und andere Deklarationen in einem NC-Programm zu ändern, kann der Befehl „#define" nicht mehrmals verwendet werden.
Die Funktion ,,#redefine*" ersetzt eine schon vorkommende Definition.

In der Regel wird die Funktion zur Zuweisung von Bearbeitungs- und Eilgangsgeschwindigkeiten für das Teach-In genutzt, falls mit mehr als zwei Geschwindigkeiten in einem NC-Programm gearbeitet werden soll.

Beispiel:

Zur Änderung z. B. einer Geschwindigkeitsdefinition ist die folgende Programmkonstruktion nicht geeignet:

```
#define () (2000);

.....

#define () (3000);
```

Diese Schreibweise kann unerwartete Ergebnisse erzielen.
Korrekt ist:

```
#define ()(2000);

#redefine *() (3000);
```

Die Funktion ,,#redefine" ersetzt eine schon vorkommende Definition. Dem Text () wird eine neue Geschwindigkeit (3000 Hertz) zugewiesen und ab dieser Stelle verwendet.

3.1.8 Der Befehl #start

#start	Starte Ausführung

Syntax: **#start**;

Erklärung: Nach Abschluss des Datenfeldes kann mit dem Befehl ,,Starte Ausführung" das übertragene Datenfeld sofort gestartet werden. Falls der Befehl verwendet wird, muss das Datenfeld mit dem **stop**-Befehl beendet werden. Falls kein Datenfeld übertragen wurde, startet der Befehl ein bereits in der Interfacekarte vorliegendes Datenfeld.

3.1.9 Der Befehl #ref_speed

#ref_speed	Referenzgeschwindigkeit definieren

Syntax: **#ref_speed** [Speedr X],[Speedr Y],[Speedr Z];

Erklärung: Mit diesem Befehl werden die Geschwindigkeiten für die Referenzfahrt der Achsen gesetzt.
Werden keine Informationen zur Referenzgeschwindigkeit übergeben, erfolgt die Ausführung mit einem Defaultwert von 800 Hz. Ein geänderter Wert bleibt nach dem Ausschalten erhalten, falls die Option Batterie-Backup eingebaut ist.
siehe auch **Menü** Einstellungen - Steuerungsparameter.

Die Referenzgeschwindigkeit muss im Bereich der zulässigen Geschwindigkeiten liegen (21 bis 3000 Hz).

Beispiel:

Einstellen der Referenzgeschwindigkeit von 2000 Hz für die Achsen X, Y und 800 Hz für die Z-Achse:

#ref_speed 2000,2000,2000;

3.1.10 Der Befehl #include

#include	Datei einfügen

Syntax: **#include <**[Dateiname]>;

Erklärung: Im PAL-PC besteht die Möglichkeit, Programmteile in mehreren Programmen zu benutzen. Dazu wird der öfter benötigte Programmteil in einer eigenen Datei erstellt oder mittels Einfügen aus einem bestehenden Programm heraus in eine Datei geschrieben. An der Stelle, an der diese Anweisungsfolge in einem Programm benötigt wird, kann dann der Befehl **#include** verwendet werden.

Beispiel: Der Compiler erwartet hinter dem Befehl einen Dateiname, optional mit vorangestellter Laufwerks- bzw. Pfadangabe.

#include <C:\Benutzer\editor\reference.txt>;

Diese Anweisung würde die Datei ,,reference.txt" vom Laufwerk ,,C:" aus dem Unterverzeichnis ,,Benutzer\editor" einfügen. Der Dateiname in der Anweisung muss entweder in **spitze Klammern** oder **Anführungszeichen** eingeschlossen werden.

Falls die angegebene Datei nicht existiert, gibt der Compiler eine entsprechende Meldung aus.

An der Stelle, an der der Befehl #include steht, werden beim Übersetzen alle Anweisungen so in die angegebenen Datei eingefügt, als ob diese Anweisungen in der zu übersetzenden Datei stehen.

3.1.11 Der Befehl Kommentar

{ } oder /	Kommentare einfügen

Syntax: **{ [**kommentar**] }** *oder* **/ [**kommentar**]**

Erklärung: Kommentare können überall im Programm vorkommen. Sie sollten Kommentare dazu benutzen, wichtige Informationen innerhalb Ihres Programmablaufes festzuschreiben.
Es sind zwei Arten von Kommentaren zugelassen:

- Kommentare, die durch die geschweifte Klammer auf "**{**" eingeleitet und durch die geschweifte Klammer zu "**}**" beendet werden

- Kommentare, die mit dem Zeichen „/“ beginnen und mit dem Zeilenendezeichen CR = Carrige Return (Enter-Tastencode) beendet werden

Ein Kommentar enthält Text und kann neben einem Befehl oder eigenständig innerhalb einer Befehlsfolge stehen.

Beispiel: #define VEL (3000); **{**VEL repräsentiert die Geschwindigkeit 3000 Hz**}**

/ verfahren der X- und Y-Achse 2 mm mit 100 Hz
move 2(100),2(100)

3.1.12 Der Befehl #GN

#GN	Gerätenummer setzen

Syntax: **#GN** [Gerätenr];

Erklärung: Mit diesem Befehl wird definiert, welche Gerätenummer die angesprochene Interfacekarte besitzt.
Die Interfacekarte muss auf die verwendete Gerätenummer eingestellt sein.

Beispiel: **#GN** 1 {Interfacekarte mit Gerätenummer 1 wird programmiert}

#GN 2 {Interfacekarte mit Gerätenummer 2 wird programmiert}

3.1.13 Der Befehl #input

#input	Speichermodus setzen

Syntax: **#input**

Erklärung: Alle dem Befehl **#input** folgenden Anweisungen werden im internen Datenspeicher abgelegt. Die Ausführung des gespeicherten Datenfeldes wird durch Betätigen der Starttaste an der Interfacekarte/Steuerung/Maschine, durch die Wahl des Befehls Start im **Menü** Transfer oder den Befehl **#start im Anwenderprogramm** veranlasst.

3.2 Der Anweisungsteil

Speicherbare Anweisungen:

Im Anweisungsteil stehen alle Anweisungen, die an die Interfacekarte übertragen und dort gespeichert werden.
Er beinhaltet die Bewegungsbefehle, Befehle zur Prozesssteuerung (Verwaltung uns Auswertung von Synchronisationszeichen, Ein- und Ausgangsimpulsen), Befehle für Wiederholschleifen und direkte Verzweigungen, Befehle zur Referenzfahrt und Definition des Nullpunktes.

3.2.1 Übersicht Befehle des Anweisungsteils

Befehl	Inhalt
circle_ccw / circle_cw	Kreisinterpolation (Kap. 3.1.22 auf Seite 48)
goto	Verzweigung (Kap. 3.2.16 auf Seite 42)
label	Sprungziele setzen (Kap. 3.2.2 auf Seite 25)
line	Interpolationsebene setzen (Kap. 3.2.14 auf Seite 40)
loop	Schleife (Kap. 3.2.8 auf Seite 34)
move/moverel	Bewegung relativ (Kap. 3.2.3 auf Seite 26)
movep	Bewegung bis Impuls (Kap. 3.2.5 auf Seite 30)
moveto/moveabs	Bewegung absolut (Kap. 3.2.4 auf Seite 28)
null	Nullpunkt setzen (Kap. 3.2.17 auf Seite 43)
on_key	Tastaturabfrage (Kap. 3.2.18 auf Seite 44)
on_port	Eingangsport lesen (Kap. 3.2.19 auf Seite 45)
port und pulse	Impuls Eingabe (Kap. 3.2.9 auf Seite 35)
reference	Referenzfahrt (Kap. 3.2.11 auf Seite 37)
repeat ... until	Wiederholung (Kap. 3.2.15 auf Seite 41)
send	Synchronisationszeichen senden (Kap. 3.2.6 auf Seite 31)
set_port	Ausgangsport setzen (Kap. 3.2.20 auf Seite 46)
set3d on / set3d off	3D-Linearinterpolation (Kap. 3.2.21 auf Seite 47)
stop	Programmende (Kap. 3.2.13 auf Seite 39)
tell	Ausgabe von Steuerzeichen (Kap. 3.2.12 auf Seite 38)
time und delay	Zeitverzögerung (Kap. 3.2.10 auf Seite 36)
wait	Warte auf Synchronisationszeichen (Kap. 3.2.7 auf Seite 32)

3.2.2 Der Befehl label

label	Sprungziele setzen

Syntax: [**label**]:

Erklärung: Ein Label ist ein Wort, das bestehen kann aus Buchstaben, Ziffern und dem Unterstreichungszeichen. Das erste Zeichen des Labels muss ein Buchstabe sein. Nach dem Label muss ein Doppelpunkt zur Markierung des Endes des Labels stehen. Das Label stellt eine Sprungadresse dar. Es ist gemischte Groß- und Kleinschreibung zugelassen, jedoch ist stets die gleiche Schreibweise zur eindeutigen Zuordnung der Sprungadressen unerlässlich.

Einige Befehle der Interfacekarte lassen Verzweigungen zu.
Die Verzweigungen können relativ angegeben werden und bestehen aus einer ganzen positiven oder negativen Zahl. Abhängig vom Vorzeichen wird im Programm um die angegebene Zahl Befehle vorwärts oder rückwärts verzweigt. Goto -5 veranlasst einen Rücksprung im Programm von 5 Befehle mit anschließender Fortsetzung.
Solche Anweisungen sind fehlerträchtig, da man sich bei weiteren Sprüngen leicht verzählt, und Einfügungen sofort eine Korrektur dieser Sprunganweisungen bedingen.
Aus dieser Überlegung heraus erlaubt es PAL-PC, Sprungziele (sogenannte Labels) im Programmtext zu verwenden.

Beispiel: Zulässige Label sind z.B.:

ANFANG:
prog_bohren:
anfang_zweitens:

Die Label ,,ANFANG" und ,,anfang" unterscheidet der Compiler, da Groß- und Kleinschreibung in PAL-PC unterschieden wird.

Unzulässig sind dagegen z.B.:

124:	eine Zahl ist als Label nicht zugelassen
1.Unterprog:	enthält ein unzulässiges Zeichen und beginnt mit einer Zahl
PROG FRAESEN:	enthält ein Leerzeichen (unzulässiges Zeichen)

In den Anweisungen, die Sprungziele verwenden, ist die Benutzung der Label im jeweiligen Befehl erläutert.

3.2.3 Der Befehl move

move **moverel**	Bewegung relativ

Syntax:

move [X(X_v)],[Y(Y_v)],[Z_1(Z_{v1})],[Z_2(Z_{v2})];
oder
moverel [X(X_v)],[Y(Y_v)],[Z_1(Z_{v1})],[Z_2(Z_{v2})];

[X(X_v)]	Zielkoordinate X relativ zum aktuellen Startpunkt, Geschwindigkeitsangabe für X-Achse
[Y(Y_v)]	Zielkoordinate Y relativ zum aktuellen Startpunkt, Geschwindigkeitsangabe für Y-Achse
[Z_1(Z_{v1})]	1. Zielkoordinate Z relativ zum aktuellen Startpunkt Geschwindigkeitsangabe für Z-Achse
[Z_2(Z_{v2})]	2. Zielkoordinate Z relativ zum Startpunkt und Geschwindigkeitsangabe für Z-Achse

Erklärung:

Die Angabe der Verfahrwege der Achsen (X, Y, Z) erfolgt in der gewählten Maßeinheit (Defaultwert ist Millimeter [mm], die Umrechnung in Schrittmotorschritte wird anhand der im Deklarationsteil definierten Mechanikparameter vorgenommen.
Bei Verwendung von gebrochenen Zahlen für die Koordinatenangabe ist stets der Punkt anstelle des Kommas zu benutzen.
Die Geschwindigkeitsangabe erfolgt als ganze Zahl in Hertz (Hz), der Mindestwert ist 21.

Für jede Achse wird eine Bewegungsgröße und eine Geschwindigkeit angegeben. Die Geschwindigkeit wird dabei in runden Klammern hinter der Bewegungsgröße vermerkt. Die Angaben für die einzelnen Achsen werden durch Kommata getrennt.

Es wird für jede angeschlossene Achse ein Parameterpaar eingeben (Zielkoordinate, Geschwindigkeit).
Für die Z-Achse werden zwei Parameterpaare erwartet, da bei Bearbeitungsvorgängen häufig der Vorgang Werkzeug absenken und anheben vorkommt.

Beispiel:

/ verfahren X-Achse 2 mm mit einer Geschwindigkeit von 2000 Hz
move 2(2000);

/ verfahren X-Achse 2 mm mit 2000 Hz, Y-Achse 2 mm und 3000 Hz
move 2(2000),2(3000);

/ verfahren X-Achse 20 mm mit 900 Hz, Z-Achse 30 mm in positive und 30 mm in
/ negative Richtung mit 1000 Hz
moverel 20(900),30(1000),-30(1000);

/ verfahren X-Achse 2 mm und Y-Achse mit 100 Hz
/ Z-Achse 2,8 mm in positive Richtung und 2 mm in negative Richtung mit 100 Hz
move 2(100),2(100),2.8(200),-2(100);

Der Bewegungsablauf erfolgt immer in der Form, dass zunächst die angeschlossenen X/Y-Achsen interpoliert verfahren werden, dann die erste Z-Koordinate verfahren wird, anschließend wird die zweite Z-Koordinate ausgeführt.
Die Interpolation der Achsen ist frei wählbar (siehe Der Befehl **line***)*

Für Achsen, die keine Bewegung ausführen sollen, kann die Bewegungsgröße auf Null gesetzt werden. Die Geschwindigkeit muss dabei allerdings auf einen zulässigen Wert gesetzt werden (zulässige Geschwindigkeitswerte liegen zwischen 21 und 20000 Hz).

Bitte beachten Sie, dass weder PAL-PC noch die Interfacekarte prüfen können, ob die Bewegung den zulässigen Bereich der angeschlossenen Mechanik verlässt.

3.2.4 Der Befehl moveto

moveto **moveabs**	Bewegung absolut

Syntax:

moveto [X(X_v)],[Y(Y_v)],[Z_1(Z_{v1})],[Z_2(Z_{v2})];
oder
moveabs [X(X_v)],[Y(Y_v)],[Z_1(Z_{v1})],[Z_2(Z_{v2})];

[X(X_v)]	Zielkoordinate X, Geschwindigkeitsangabe für X-Achse
[Y(Y_v)]	Zielkoordinate Y, Geschwindigkeitsangabe für Y-Achse
[Z_1(Z_{v1})]	Zielkoordinate Z1, Geschwindigkeitsangabe für Z-Achse
[Z_2(Z_{v2})]	Zielkoordinate Z2, Geschwindigkeitsangabe für Z-Achse

Erklärung:

Die Angabe der Zielkoordinaten der Achsen (X, Y, Z) erfolgt in der gewählten Maßeinheit (Defaultwert ist Millimeter [mm], die Umrechnung in Schrittmotorschritte wird anhand der im Deklarationsteil definierten Mechanikparameter vorgenommen.
Bei Verwendung von gebrochenen Zahlen für die Koordinatenangabe ist stets der Punkt anstelle des Kommas zu benutzen.
Die Geschwindigkeitsangabe erfolgt als ganze Zahl in Hertz (Hz), der Mindestwert ist 21.

Der Befehl **moveto/moveabs** veranlasst eine lineare Bewegung der Achsen zu den angegebenen Zielkoordinaten mit den vereinbarten Geschwindigkeiten, dies entspricht einem Wegbefehl mit Absolutmaßangabe, d. h. die Zielkoordinaten beziehen sich auf den gesetzten Nullpunkt des Werkstückkoordinatensystems.

Aus Kompatibilitätsgründen zum relativen Positionierbefehl werden auch hier für die Z-Achse zwei Zahlenpaare erwartet. Da in diesem Befehl Zielkoordinaten definiert werden, muss der Z_2- Wert stets Null sein und wird ignoriert.

Beispiel:

/ verfahren X-Ist+2 mm mit einer Geschwindigkeit von 2000 Hz
moveto 2(2000);

/ verfahren X-Ist+2 mm mit 2000 Hz, Y-Ist+2 mm und 3000 Hz
moveabs 2(2000),2(3000);

/ verfahren X-Ist+20 mm mit 900 Hz, Z-Ist +30 mm in positive und 30 mm in
/ negative Richtung mit 1000 Hz
moveto 20(900),30(1000),-30(1000);

/ verfahren X-Ist+2 mm und Y-Ist+2mm mit 100 Hz
/ Z-Ist+2,8 mm in positive Richtung und 2 mm in negative Richtung mit 100 Hz
moveabs 2(100),2(100),2.8(200),-2(100);

Der Bewegungsablauf erfolgt immer so, dass zunächst die angeschlossenen X/Y-Achsen interpoliert verfahren werden, dann die Z-Koordinate verfahren wird. Die Interpolation der Achsen ist frei wählbar (siehe Der Befehl **line***)*

Für Achsen, die keine Bewegung ausführen sollen, wird die Bewegungsgröße auf Null gesetzt werden. Die Geschwindigkeit muss dabei allerdings einen zulässigen Wert erhalten (zulässige Geschwindigkeitswerte liegen zwischen 21 und 20000).

Hinweis:

Der Befehl kann erst verwendet werden, nachdem die Achsenanzahl gesetzt worden ist.
Bitte beachten Sie, dass weder PAL-PC noch die Interfacekarte prüfen können, ob die Bewegung den zulässigen Bereich der angeschlossenen Mechanik verlässt.

3.2.5 Der Befehl movep

movep	Bewegung bis Impuls

Syntax: **movep** [X(X_v)],[Y(Y_v)],[Z_1(Z_{v1})],[Z_2(Z_{v2})],[A(A_v)];

[X(X_v)]	Zielkoordinate X relativ zum aktuellen Startpunkt, Geschwindigkeitsangabe für X-Achse
[Y(Y_v)]	Zielkoordinate Y relativ zum aktuellen Startpunkt, Geschwindigkeitsangabe für Y-Achse
[Z_1(Z_{v1})]	1. Zielkoordinate Z relativ zum aktuellen Startpunkt Geschwindigkeitsangabe für Z-Achse
[Z_2(Z_{v2})]	2. Zielkoordinate Z relativ zum Startpunkt und Geschwindigkeitsangabe für Z-Achse

Erklärung: Die Maßeinheit der Zielposition (X, Y, Z) ist Millimeter [mm] (Defaultwert).
Bei Verwendung von gebrochenen Zahlen für die Koordinatenangabe ist stets der Punkt anstelle des Kommas zu benutzen.
Die Geschwindigkeitsangabe erfolgt ganzzahlig in Hertz (Hz), der Mindestwert ist 21.

Der Befehl ,,Bewegung bis Impuls" verhält sich genau wie der Befehl ,,Bewegung relativ", tritt jedoch ein Impuls am Impulseingang auf (z. B. ein Stop-Impuls bei Betätigen der Stop-Taste), wird die Verfahrbewegung beendet und der nächste Befehl ausgeführt.
Wird während der Verfahrbewegung kein Impuls empfangen, findet die Bewegung mit der angegebenen Distanz statt und anschließender Fortsetzung im Programm mit dem folgenden Befehl.
Der auftretende Impuls muss eine Minimumsbreite haben, diese liegt bei etwa 20 µsec. Die maximale Breite des Impulses darf 100 µsec nicht überschreiten.
Beachten Sie bei Anwendung des Befehls, dass der Impuls am Stoptastereingang der Interfacekarte anliegt. Falls ein Impuls auftritt und die Interfacekarte eine normale Positionierung ausführt, wird diese Bewegung gestoppt und der folgende Befehl wird ausgeführt. Sollten Probleme mit der anliegenden Impulsbreite auftreten, wird empfohlen, nach dem movep-Befehl einen time- oder delay-Befehl mit einer Wartezeit >= 1 (100msec) einzufügen.

Beispiel:

```
#axis x;   {angeschlossene Achse = X-Achse }
#input     ; speichere die folgenden Befehle
movep 20(1000); {die X-Achse 20 mm verfahren bis Impuls anliegt}
reference x;      {Referenzfahrt der X-Achse}
```

3.2.6 Der Befehl send

send	Synchronisationszeichen senden

Syntax: **send** [Zahl];

Erklärung: [Zahl] Zahl zwischen 33 und 126 (ASCII-Zeichen)

Um der Interfacekarte eine Prozesssynchronisation mit einer zweiten Interfacekarte oder einem übergeordneten Rechner zu ermöglichen, kann bei Erreichen eines bestimmten Punktes im Datenfeld ein Synchronisationszeichen gesendet werden.

Das Synchronisationszeichen muss im Bereich zwischen 33 und 126 liegen, das Zeichen 64 (@) sollte nicht verwendet werden.
Dieser Zeichenvorrat beinhaltet u. a. alle Ziffern sowie Groß- und Kleinbuchstaben.

Den verwendeten Synchronisationszeichen können am Anfang des Programms mit dem Befehl **#define** Namen zugewiesen werden, die im Befehl **send** anstelle der Ziffer verwendet werden.
Die Wahl sinnvoller Namen verbessert die Transparenz des Programms und der Prozesssynchronisation.

Beispiel:

```
/ dem Synchronisationszeichen 90 wird der Text Bohren_ist_fertig
/ zugewiesen
#define Bohren_ist_fertig  90;
.....
send Bohren_ist_fertig;
```

Sie können die Deklarationen für die beiden zu synchronisierenden Geräte mittels des Befehles **#include** in einer gemeinsam genutzten Datei definieren.

Beispiel der Kommunikationsfunktion:

Zur Sichtbarmachung der Befehle send/wait können Sie mit Hilfe des Kommunikationsfensters das nachstehende Programm testen:

```
#axis x;
repeat
    move 20(800);
    send 90;
    wait 65;
    move -20(800);
until 5;
stop
```

Nach Übersetzung und Übertragung des Programms an die Steuerung, starten Sie das Programm bitte aus dem Kommunikationsfenster (**Menü** Transfer - Terminal ...) mit der Tastenkombination (Shift+F1).
Die X-Achse wird 20 mm vorwärts bewegt und sendet als Aufforderung ein „Z“ (das Zeichen 90 entspricht Z). Das Programm wartet nun auf das Zeichen 65. Geben Sie das Zeichen ein, indem Sie die Alt-Taste drücken und eine 65 auf der numerischen Tastatur eingeben. Das Zeichen A erscheint im Kommunikationsfenster, nach dieser Eingabe wird das Programm fortgesetzt bis zum nächsten wait-Befehl in der Programmschleife.

3.2.7 Der Befehl wait

wait	Warte auf Synchronisationszeichen

Syntax: **wait** [Zahl], [Offset];

[Zahl]	Zahl zwischen 33 und 126 (ASCII-Zeichen)
[Offset]	Zahl, die angibt, wie viele Zeilen vor- oder zurückgesprungen werden sollen *oder* Label (Sprungadresse), zu dem verzweigt und fortgefahren werden soll.

Erklärung: Analog zu dem zuletzt besprochenen Befehl dient ,"Warte auf Synchronisationszeichen" ebenfalls der Prozesssynchronisation der Interfacekarte mit einer weiteren Interfacekarte oder einem übergeordneten Rechner.

Die Interfacekarte kann bei der Ausführung des Befehles mehrere Aktionen durchführen. Diese Aktionen umfassen:

Das Zeichen selbst wird empfangen:
Die Interfacekarte bearbeitet dann den nächsten im Datenfeld abgelegten Befehl.

Das Zeichen +1 wird empfangen:
Die Interfacekarte verzweigt (relativ) zu der Stelle, die in dem Befehl angegeben wurde.

Das Zeichen 127 wird empfangen:
Die Interfacekarte löst einen Reset aus und wartet auf Befehle des Rechners

Beispiel: Beispiel 1:

wait 90; {Warte auf das Zeichen 90 und arbeite nach Empfang den nächsten Befehl im abgelegten CNC-Programm ab}

wait 90,-5; {bei Empfang des Zeichens 90, gehe zum nächsten Befehl im Programm, bei Empfang des Zeichens 91, gehe 5 Zeilen im Programm zurück}

wait 80,anfang; {bei Empfang des Zeichens 81 gehe zum Label ,,anfang"}

Beispiel 2:

Ein übergeordneter Rechner prüft Teile und teilt der Interfacekarte nach Aufforderung (Teil ist da) mit, ob die Prüfung erfolgreich war. Falls das Teil in Ordnung ist, wird TEIL_OK+1 übergeben, die Interfacekarte legt dann das Teil in die bearbeitende Maschine, anderenfalls sendet der übergeordnete Rechner TEIL_OK+0, und die Interfacekarte legt das Teil beiseite und holt das nächste Teil.

Nach gezeigtem Muster lassen sich sehr komplexe Systeme aufbauen und steuern.

```
label1:
        move ...                    { hole Teil }
        send TEIL_DA;               { signalisiere bereit für Prüfung }
        wait TEIL_OK, weiter;       { warte auf Freigabe }
        move ...                    { lege Teil beiseite (defektes Teil) }
        goto label1;                { versuche nächstes Teil }

weiter:
        move ...                    { lege Teil in Maschine }
        move ...                    { in Ausgangsposition }
        goto label1;                { verzweige wieder zu label1 }
```

3.2.8 Der Befehl loop

loop	Schleife

Syntax: **loop** [Anzahl] **times** [Label];

Erklärung:

[Anzahl] Zahl zwischen 0 und 32767. 0 bedeutet eine Endlosschleife.

[Label] Markiert den Schleifenanfang, wird bei jedem Schleifenzyklus entsprechend der Anzahl Wiederholungen angesprungen

Schleifen dienen dazu, hintereinander vorkommende gleichartige Abläufe bzw. Bewegungsfolgen zusammenzufassen. Somit wird der zur Verfügung stehende Speicherplatz effektiver genutzt.

Beispiel:

Beispiel 1:

```
schleife_1:
        move ...                    ;Bewegungsbefehl

;wiederhole alle Befehle ab Label schleife_1" fünfmal

        loop 5 times schleife_ 1;
```

Beispiel 2:

```
schleife_endlos:
        move …                      ; Bewegungsbefehl

; Endlosschleife: wiederhole alle Befehle ab Label ,,schleife_endlos" endlos

        loop 0 times schleife_endlos;
```

Zur Schleifenbildung können ebenfalls die Befehle ,,repeat" und ,,until" genutzt werden.

3.2.9 Der Befehl port und pulse

port **pulse**	Impuls Eingabe / Impuls Ausgabe

Syntax: **port** [Zustand]
pulse [Zustand)

Erklärung: [Zustand] Gibt den gewünschten Zustand des Ausgangs an.

Syntax	**Zustand**
port on;	Portausgang ein
port off;	Portausgang aus
pulse out;	50 msec Impuls geben
pulse in;	Auf Impuls warten
pulse sync out;	Puls senden, auf Bestätigung warten
pulse sync in;	Auf Impuls warten, Bestätigung senden

Falls die Hardwareoption ,,Impulsausgabe" an der Interfacekarte angeschlossen ist, kann diese durch diese Befehle angesteuert werden. Die einzelnen Optionen sind hier kurz noch einmal aufgeführt, der Compiler versteht statt des Befehles ,,port" auch den Befehl ,,pulse":

Auch der Impulsausgang lässt sich zur Synchronisation zweier Geräte nutzen (die letzten beiden Optionen). Eine der Hauptaufgaben der Impulssteuerung ist es, während des Ablaufes eines Datenfeldes auf den Start-Taster zu warten. Dies kann notwendig sein, wenn an einer bestimmten Stelle ein manueller Eingriff in den Prozessablauf notwendig wird, und die Interfacekarte erst nach Beendigung des Eingriffes weiterarbeiten darf. Da der Start-Taster an den Impulseingang angeschlossen ist, kann hierfür der Befehl ,,pulse in" benutzt werden.

Beispiel: pulse in; { warten auf Bestätigen des Starttasters }

3.2.10 Der Befehl time und delay

time **delay**	Zeitverzögerung

Syntax: **time** [Zeit];
delay [Zeit];

Erklärung: [Zeit] Gibt die Wartezeit an. Die Angabe der Zeit erfolgt in Zehntelsekunden.

Eine Zeitverzögerung veranlasst die Interfacekarte die angegebene Wartezeit zu warten. Die Wartezeit kann durch Bestätigen des Stop-Tasters nicht abgebrochen werden.

Beispiel: Beispiel:

```
time 500;    { warte 50 Sekunden }
delay 20;    { warte 2 Sekunden }
```

Die maximale Wartezeit beträgt 3276.7 Sekunden.
Die Befehle müssen mit einem Semikolon abgeschlossen werden

3.2.11 Der Befehl reference

reference	Referenzfahrt der Achsen

Syntax: **reference** [Achsen];

Erklärung: Achsen {x, y, z}

PAL-PC erwartet nach dem Referenzbefehl die Angabe, für welche Achsen eine Referenzfahrt ausgelöst werden soll. Sie können hier jede Achse angeben, die im Achsenwahlbefehl enthalten ist.

Beispiel:

Beispiel 1:
reference xy; { führt eine Nullpositionierung der X- und Y-Achse durch}
reference x; { führt eine Nullpositionierung der X-Achse durch }
reference xyz; { führt eine Nullpositionierung aller drei Achsen durch }

Zu beachten ist, dass die Achsen in der Reihenfolge ZYX verfahren, d.h. es wird zunächst die Referenzfahrt für die Z-Achse, dann die Referenzfahrt für die Y-Achse und als letztes die Referenzfahrt für die X-Achse ausgeführt. Wenn dieses Verhalten unerwünscht ist, müssen zwei Referenzbefehle verwendet werden (s. Beispiel 2).

Beispiel 2:

reference x; { führt Nullpositionierung der X-Achse durch }

reference y; { führt Nullpositionierung der Y-Achse durch }

Hinweis:

Soll mit einer vom Defaultwert anweichenden Geschwindigkeit referenziert werden , vereinbaren Sie bitte die Referenzgeschwindigkeit im Deklarationsteil des Programms.
siehe
3.1.9 Der Befehl #ref_speed auf Seite 19

3.2.12 Der Befehl tell

tell	Ausgabe von Steuerzeichen

Syntax: **tell** [GN] [Optionen];

Erklärung:

[GN]	Gibt die Gerätenummer der anzusprechenden Interfacekarte an
[Optionen]	Gibt die Befehle für die anzusprechende Interfacekarte an (siehe Beispiele)

Die Ausgabe von Steuerzeichen gehört zur Gruppe der Befehle zur Prozesssynchronisation. Der Befehl gibt bei seiner Ausführung bis zu vier Zeichen aus. Primär ist der Befehl dazu gedacht, bei einer zweiten angeschlossenen Interfacekarte einen Start oder eine Referenz auszulösen. Er kann jedoch auch für beliebige andere Zwecke genutzt werden.

Beispiel:

tell 0 start;	{ Starte Gerät 0 }
tell 0 start,wait;	{ Starte Gerät 0, warte auf Ende }
tell 0 reference xyz;	{ löse Referenzfahrt bei Gerät 0 aus }
tell 0 reference,wait xyz;	{ löse Referenzfahrt bei Gerät 0 aus, warte auf Ende }

Die Ausführung des Befehles können Sie am besten mit der Funktion Kommunikation testen.

3.2.13 Der Befehl stop

stop.	Programmende markieren

Syntax: **stop**.

Erklärung: Dieser Befehl kennzeichnet das Ende eines Programms.

Beispiel: Beispiel:

```
#axis x;                    { Achsenwahl X-Achse  }
#units mm;                  { Maßeinheit = Millimeter }
#input
reference x;                { Nullpositionierung der X-Achse }
move 100(8000);             { fahre 100 mm in  X-Richtung}
stop.                       { Programmende markieren }
```

Der Befehl muss in jedem Programm enthalten sein.

3.2.14 Der Befehl line

line	Interpolationsebene setzen

Syntax: **line** [Achsen];

Erklärung: Achsen {x, y, z}

Gibt die Achsen an, die interpoliert werden sollen:
In der Standardeinstellung der Interfacekarte werden die angeschlossenen X/Y-Achsen gegeneinander interpoliert (geradlinig zum Zielpunkt verfahren). Durch die Wahl der Interpolationsebene besteht jedoch die Möglichkeit, jede andere Ebenenkonfiguration als Hauptebene zu definieren.

Beispiel:

```
line xy;    { interpoliere X und Y }
line xz;    { interpoliere X und Z }
line yz;    { interpoliere Y und Z }
```

Eine Einstellung der Interpolationsebene bleibt innerhalb eines NC-Programms solange aktiv, bis ein anderer ,,line"-Befehl verwendet wird. Beim Start eines Programms ist generell ,,line xy" eingestellt.
Die Zuordnung der Interpolationsebene hat keinen Einfluss auf die Referenzreihenfolge, d. h., die Referenzfahrt wird weiterhin mit der Reihenfolge ,,ZYX" vorgenommen.

3.1.15 Der Befehl repeat ... until

repeat ... until	Wiederholung

Syntax:

repeat
until [Anzahl];

Erklärung:

[Anzahl] Gibt die Anzahl der Abschnittswiederholung an. Wird für Anzahl der Wert 0 eingesetzt, entsteht eine Endlosschleife.

Die Befehle ,,repeat" und ,,until" dienen dazu, Abschnitte in einem Programm zu wiederholen. ,,repeat" markiert den Anfang der Wiederholung, ,,until" das Ende.

Beispiel:

```
repeat  { Schleifenanfang kennzeichnen }
        move 5(800)   { relative Bewegung }
        delay 20      { 2 sec warten... }

until 7;  { Schleifenende }
```

Nach ,,repeat" muss kein Semikolon stehen, da der Befehl keine Parameter enthält. ,,repeat" und ,,until" können geschachtelt sein, d.h., eine Wiederholung kann eine weitere Wiederholung enthalten. Die Anzahl der Befehle, die maximal zwischen ,,repeat" und ,,until" stehen dürfen, ist nur durch den maximal in der Interfacekarte zur Verfügung stehenden Speicherplatz begrenzt.

Der Befehl ,,until" muss mit einem Semikolon abgeschlossen werden.

3.2.16 Der Befehl goto

goto	Verzweigung

Syntax: **goto** [Ziel];

Erklärung:

[Ziel] Anzahl Zeilen, die im Programmablauf übersprungen werden
oder
Label (Sprungziel), markiert die Programmzeile, ab welcher fortgesetzt wird

Zahl kann positiv oder negativ sein, je nachdem ob vor- oder rückwärts verzweigt werden soll.
Zur Definition von Label siehe auch 3.2.2 Der Befehl label auf Seite 25
Der Befehl goto veranlasst die Programmfortsetzung ab dem angegebenen Sprungziel.

Beispiel:

goto 5; {die nächsten 5 Anweisungen überspringen}
goto -5; {5 Anweisungen zurück verzweigen}
goto anfang; {zu Label ,,anfang" zurück verzweigen}
goto ende; {zu Label ,,ende“ vorwärts verzweigen}

3.2.17 Der Befehl null

null	Nullpunkt setzen

Syntax: **null** [Achsen];

Erklärung: Achsen {x, y, z}

Definieren eines Nullpunktes auf der aktuellen Position für die angegeben Achsen.

Der Werkstücknullpunkt kann durch eine Referenzfahrt wieder auf den mechanischen Nullpunkt der Anlage gelegt werden.

Beispiel:

```
#define () 800;
reference xyz;

/ verfahren an die Position x=20mm, y=30mm, z=15mm
moveto 20( ),30( ),15( ),0( );
/ neuen Werkstücknullpunkt an dieser Position setzen
null xyz;

/ verfahren zur Position x=10mm, y=20mm, z=20mm relativ zum
/ Werkstücknullpunkt oder x=30mm, y=50mm, z=35mm relativ zum
/ Maschinennullpunkt
moveto 10( ),20( ),20( ),0( );
```

3.2.18 Der Befehl on_key

on_key	Tastaturabfrage

Syntax: **on_key** [Tastnr], [Label];

Erklärung:

[Tastnr] Gibt die jeweilige Tastaturnummer an.

[Label] Gibt den Label an, zu dem nach Drücken der Tastaturnummer verzweigt werden soll

Dieser Befehl findet seine Anwendung zum Zeitpunkt des Anschlusses einer "isel"-Programmwahleinheit. Diese steht dem Anwender als programmierbare Tastatur zu Verfügung (Es besteht nicht die Möglichkeit per Programmwahleinheit die Interfacekarte zu programmieren). Die isel-Programmwahleinheit besteht aus einer 12er-Tastatur und wird über die serielle Schnittstelle mit der Interfacekarte verbunden.

Beispiel:

```
#axis x;
#units mm;
#elev 4;
Anfang:
    on_key 1, do_reference;
    on_key 2, do_move;
    on_key 3, Ende;
goto Anfang;

do_reference:
    reference x;
goto Anfang;

do_move:
    move 5(8000);
goto Anfang;

Ende:
stop.
```

Wird die Taste F1 der Programmwahleinheit betätigt, verzweigt das Programm zum Unterprogramm do_reference, und die Interfacekarte führt den Befehl Referenzfahrt aus.
Wird die Taste F2 der Programmwahleinheit betätigt, verzweigt das Programm zum Unterprogramm do_move, und die Interfacekarte führt den Befehl Bewegung aus.
Wird die Taste F3 der Programmwahleinheit betätigt, verzweigt das Programm zum Unterprogramm Ende und die Interfacekarte führt den Befehl stop. aus.

3.2.19 Der Befehl on_port

on_port	Eingangsport lesen

Syntax: **on_port** [ADRESSE], [BITNR]=[WERT],[OFFSET];

Erklärung:

[ADRESSE] Angabe der Eingangsports E1 oder E2 (ersetzt die Zahlenwerte 65531 bzw. 65532 aus der DOS-Version)

[BITNR] bitweises Lesen BITNR=[1,...,8]
byteweises Lesen BITNR=0 bzw. 128

[WERT] bitweises Lesen →WERT=0 *oder* 1
byteweises Lesen →Abfrage des Bitmusters des Eingangsports

[OFFSET] Zahlenwert *oder* Label, der eine Verzweigung vorwärts oder rückwärts im Programmablauf bewirkt

Der Eingangsport wird auf ein gewünschtes Bit oder Bitmuster abgefragt. Ist die Bedingung erfüllt, wird eine Verzweigung durchgeführt.

Die Interfacekarte liest einen Eingangsport ein und verzweigt bei einer wahren Bedingung.

Beispiel: 1. Fall: bitweises Lesen

Befehl	Abfragekriterium	Verzweigung
on_port E1,2=0,3;	Bit 2 = aus	3 Zeilen vorwärts
on_port E1,8=1,-2;	Bit 8 = ein	2 Zeilen rückwärts

2. Fall: byteweises Lesen

Befehl	Abfragekriterium	Verzweigung
on_port E1,0=10,3;	Dual 00001010	3 Zeilen vorwärts
on_port E1,0=0,-2;	Dual 00000000	2 Zeilen rückwärts
on_port E1,0=205,-4;	Dual 11001101	4 Zeilen rückwärts

3.2.20 Der Befehl set_port

set_port	Ausgangsport setzen

Syntax: **set_port** [ADRESSE], [BITNR]=[WERT];

Erklärung:

[ADRESSE] Angabe der Ausgangsports A1 *oder* A2 (ersetzt die Zahlenwerte 65529 bzw. 65530 aus der DOS-Version)

[BITNR] Zahlenwert, der zur Fallunterscheidung zwischen bitweisem oder byteweisem Setzen des Ausgangsmusters dient.

bitweises Setzen BITNR= [1,...,8]
byteweises Setzen BITNR=0 bzw. 128

[WERT] bitweises Setzen →WERT= 0 *oder* 1

byteweises Setzen →Zahl zwischen 0 und 255, wird als Bitmuster (Dualzahl) optisch wie schaltungstechnisch am Ausgangsport angelegt

An der Interfacekarte wird ein gewünschtes Ausgangsmuster oder ein definiertes Ausgangsbit gesetzt.

Beispiel: 1. Fall: bitweises Setzen

Befehl	Ausgangsport	Bit	Zustand
set_port A1,5=0;	A1	5	aus
set_port A1,4=1;	A1	4	ein
set_port A2,4=0;	A2	4	aus
set_port A2,1=1;	A2	1	ein

2. Fall: byteweises Setzen

Befehl	Ausgangsport	Dual
set_port A1,0=10;	A1	00001010
set_port A1,0=27;	A1	00011011
set_port A2,0=205;	A2	11001101
set_port A2,0=255;	A2	11111111
set_port A2,0=0;	A2	00000000

3.2.21 Der Befehl 3D-Linearinterpolation

set3d on **set3d off**	3D-Interpolation

Syntax: **set3d on**
set3d off

Erklärung: Ein- oder ausschalten der 3-dimensionalen (räumlichen) Interpolation

Die Anweisung wirkt modal, das bedeutet, alle dem Befehl set3d on folgenden Bewegungsbefehle (z. B. move/moverel, moveto/moveabs) werden solange dreidimensional ausgeführt, bis mit dem Befehl set3d off dieser Modus wieder ausgeschaltet wird.
Die Angabe von Z_2-Parametern in diesen Verfahrbewegungen wird ignoriert.
Als Geschwindigkeitsangabe der Interpolation wird der Wert der X-Achse herangezogen.
Die maximale Geschwindigkeit für eine 3D-Interpolation beträgt 10000 Hz.

Beispiel:

```
#axis xyz;
reference xyz;
set3d on;        {umschalten auf 3D-Interpolation}
/ Verfahrbewegung von X-, Y-, Z-Achse gleichzeitig
move 10(700),15(700),3(400),0(30);
set3d off;       {ausschalten der 3D-Interpolation}
```

Hinweis:

Die Einleitung einer Referenzfahrt schaltet automatisch auf 2.5 dimensionale Interpolation zurück.
Die korrekte Bearbeitung einer 3D-Interpolation setzt als Bezugsebene eine XY-Ebene voraus.
siehe auch 3.2.14 Der Befehl line auf Seite 40

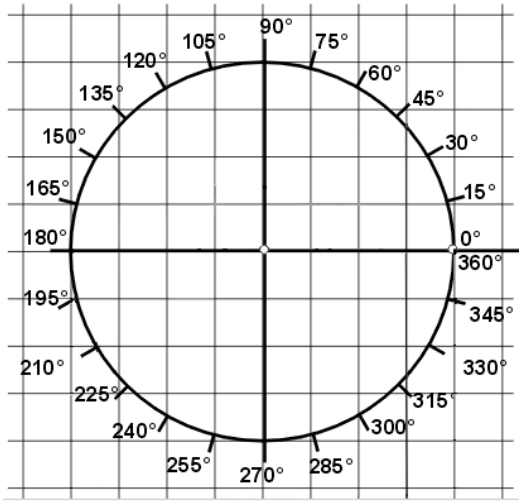
3.2.22 Der Befehl circle

circle_cw **circle_ccw**	Kreisinterpolation im Uhrzeigersinn Kreisinterpolation entgegen dem Uhrzeigersinn

Syntax: **circle_cw [r(v)], [Winkel1], (Winkel2]**
circle_ccw [r(v)], [Winkel1], (Winkel2]

cw	Drehrichtung im Uhrzeigersinn
ccw	Drehrichtung entgegen dem Uhrzeigersinn
[r(v)]	r = Radius v = Bearbeitungsgeschwindigkeit
[Winkel1]	Startwinkel der Kreisbahn in [grad]
[Winkel2]	Endwinkel der Kreisbahn in [grad]

Erklärung: Es wird eine Kreisbewegung verfahren, abhängig vom definierten Radius und vom Anfangs- und Endpunkt (definiert über Start- und Endwinkel).
Die Drehrichtung wird durch die Parameter cw (clockwise) im Uhrzeigersinn (mathematisch positiver Drehsinn) und ccw (counter clockwise) entgegen dem Uhrzeigersinn (mathematisch negativer Drehsinn) bestimmt und ist vom Startpunkt aus zu sehen.
Mit einem circle-Befehl kann maximal ein Vollkreis (Differenz zwischen Start- und Endwinkel = 360 grad) gefahren werden.
Basis bei der Definition des zu fahrenden Kreissegments ist der nachfolgend abgebildete Kreis.



Beispiel:

1. Verfahren verschiedener Kreise bzw. Kreisabschnitte in positiver Richtung (entgegen dem Uhrzeigersinn). Der Durchmesser des Radius beträgt 20 mm, die Verfahrgeschwindigkeit 5000 Hz.

```
/ Vollkreis, beginnend bei 0 grad, endend bei 360 grad
circle_ccw 20(5000),0,360

/ Kreissegment mit einem Winkel = 45 grad
/ beginnend bei 0 grad, endend bei 45 grad
circle_ccw 20(5000),0,45

/ Kreissegment mit einem Winkel = 150 grad
/ beginnend bei 60 grad, endend bei 210 grad
circle_ccw 20(5000),60,210

/ Vollkreis, beginnend bei 225 grad, endend bei 225 grad
circle_ccw 20(5000),225,585
```

2. Verfahren verschiedener Kreise bzw. Kreisabschnitte in negativer Richtung (im Uhrzeigersinn). Der Durchmesser des Radius beträgt 20 mm, die Verfahrgeschwindigkeit 5000 Hz. Hier ist zu beachten, dass der Startwinkel immer größer ist als der Endwinkel. Dies wird durch die Addition von 360 grad zum Startwinkel erreicht.

```
/ Vollkreis, beginnend bei 0 grad, endend bei 360 grad
circle_cw 20(5000),360,0

/ Kreissegment mit einem Winkel = 305 grad
/ beginnend bei  0 grad, endend bei 45 grad
circle_ccw 20(5000),360,45

/ Kreissegment mit einem Winkel = 150 grad
/ beginnend bei 60 grad, endend bei 210 grad
circle_ccw 20(5000),420,210

/ Vollkreis, beginnend bei 225 grad, endend bei 225 grad
circle_ccw 20(5000),585,225
```

Index

3

3D-Interpolation 47

A

Achsenwahl 10
Anweisungsteil 6
Ausgabe von Steuerzeichen 38
Ausgangsport setzen 46
axis 10

B

Battery-Backup 4, 5
Bewegung absolut 28
Bewegung bis Impuls 30
Bewegung relativ 26

C

ccw 48
circle_ccw 48
circle_cw 48
CNC-Mode 4
cw 48

D

Datei einfügen 20
define 14
Deklarationsteil 6
delay 36
DNC-Mode 4

E

Eingangsport lesen 45
elev 13
Endwinkel der Kreisbahn 48
EP10905 5

F

Fluchtzeichen 14, 17

G

GN 22, 38
goto 42

I

IML4 5
Impuls Ausgabe 35
Impuls Eingabe 35
include 20
input 23
Interfacekarte 5
Interpolationsebene 40

K

Kommentare 21
Kreisinterpolation 48

L

Label 25
line 40
loop 34

M

Maßeinheit 12
Memory Card 5
move 26
moveabs 28
movep 30
moverel 26
moveto 28

N

Neudefinition 17
null 43
Nullpunkt 43

O

on_key 44
on_port 45

P

port 35
Programmende 39
pulse 35

R

redefine 17
ref_speed 19
reference 37
Referenzfahrt der Achsen 37
Referenzgeschwindigkeit 19
repeat 41

S

Schleife 34
Schrittzahl 11
send 31
serielle Schnittstelle 5
set_port 46
set3d off 47, 48
set3d on 47
Speichermodus 23
Spindelsteigung 13
Sprungziele 25
Stand-Alone-Betrieb 4
start 18
Startwinkel der Kreisbahn 48
Steigung 13
steps 11
stop 39

Synchronisationszeichen 31

T

Tastaturabfrage 44
tell 38
Textersatz 14
time 36

Ü

Übertragungsrate 5

U

units 12
until 41

V

Verzweigung 42
Verzweigungen 25

W

wait 32
Warte auf Synchronisationszeichen 32
Wiederholung 41

Z

Zeitverzögerung 36
Zielkoordinaten 26