

PROGRAMMIERVORSCHRIFT

Info

ProNC

Die universelle CNC-Software

Version 1.45.3.0
NCComp Compiler Version: 1.45

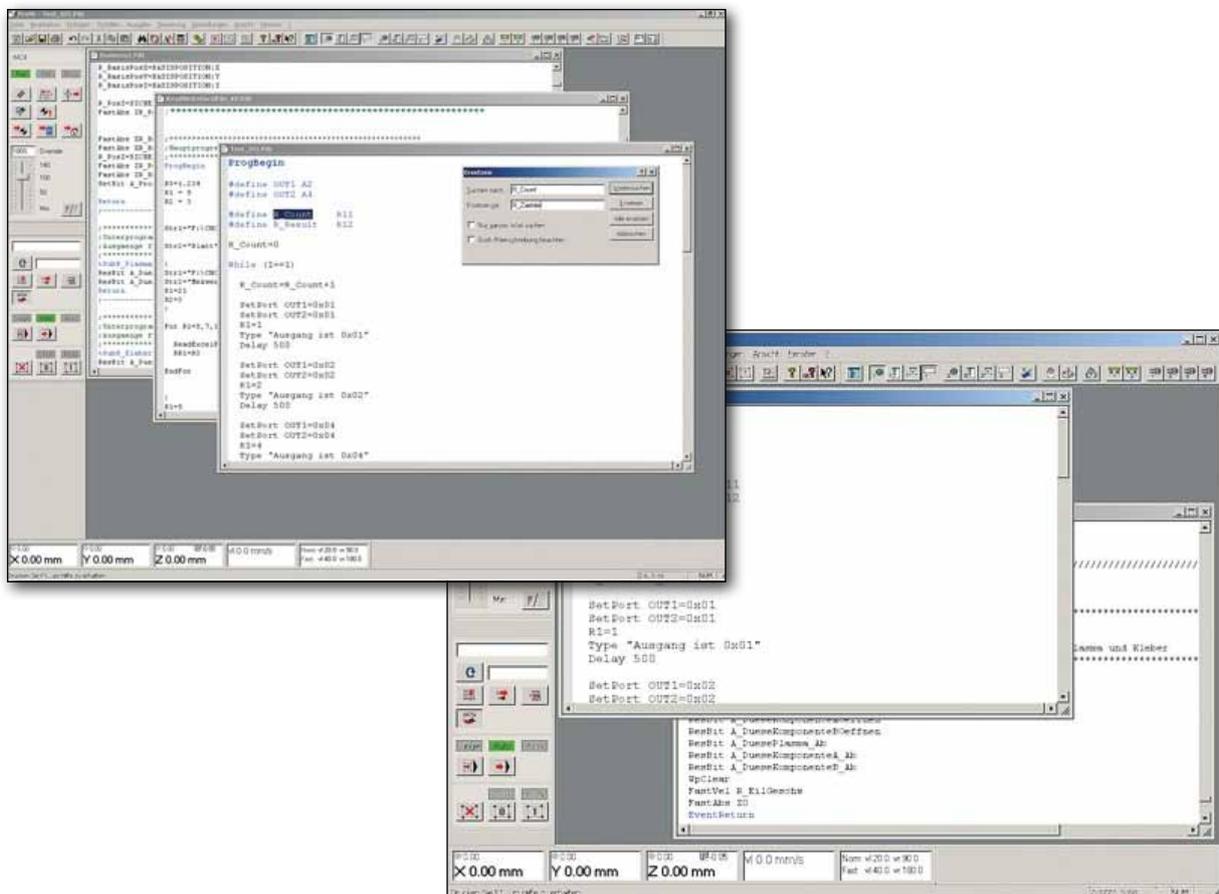
Universelle Bedien- und Programmieroberfläche mit integriertem Compiler/Interpreter für CNC-Programme nach isel PAL (Process Automation Language) sowie DIN 66025.

Diese Software ist eine Entwicklung der isel Germany AG und ist urheberrechtlich geschützt. Jede unerlaubte Vervielfältigung ist untersagt und wird strafrechtlich verfolgt.

Copyright © 1999-2009 isel Germany AG. All rights reserved.

isel Germany AG
Bürgermeister-Ebert-Straße 40
D-36124 Eichenzell
+49 6672 898-0
www.isel-germany.de





The screenshot displays the ProNC software interface. The main window shows a CNC program with the following code:

```
ProgBegin
R_Count=0
While (I=1)
  R_Count=R_Count+1
  SetPort OUT1=0x01
  SetPort OUT2=0x01
  Delay 500
  SetPort OUT1=0x02
  SetPort OUT2=0x02
  Delay 500
  SetPort OUT1=0x04
  SetPort OUT2=0x04
  Type "Ausgang ist 0x04"
EndWhile
```

The interface includes a control panel on the left with various buttons and a status bar at the bottom showing coordinates (X 0.00 mm, Y 0.00 mm, Z 0.00 mm) and spindle speed (M 0.0 mm/min).

Zu dieser Anleitung:

In dieser Anleitung finden Sie verschiedene Symbole, die Sie auf wichtige Informationen hinweisen.

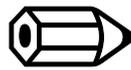
Achtung:



Hinweis:



Beispiel:



Merke:



© Fa. **isel**automation KG 2003
Alle Rechte vorbehalten

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

Kein Teil dieser Veröffentlichung darf ohne vorherige schriftliche Genehmigung der
Firma iselautomation KG in jeglicher Weise reproduziert, in einem EDV-System gespeichert oder
übertragen werden.

Alle Angaben in diesem Handbuch erfolgen ohne Gewähr. Änderungen des Inhalts sind jederzeit ohne
Vorankündigung möglich.

Hersteller: Fa. **isel** Germany AG
Bürgermeister-Ebert-Straße 40
D-36124 Eichenzell

Tel.: (06659) 981-0
Fax: (06672) 981-776
email: automation@isel.com
<http://www.isel.com>

Stand: 10/2003

Inhalt

1 EINFÜHRUNG	6
1 EINFÜHRUNG	6
1.1 COPYRIGHT	6
1.2 ZWECKBESTIMMUNG DES PROGRAMMPAKETES PRoNC	7
1.2.1 Kurzcharakteristik	7
1.2.2 Grundsätze einer Programmierung mit ProNC	9
2 PROGRAMMIERUNG MIT PRoNC	12
2 PROGRAMMIERUNG MIT PRoNC	12
2.1 AUFBAU DES STEUERPROGRAMMS	12
2.1.1 Programmstruktur (Hauptprogramm)	12
2.1.2 Anmerkungen im Programm (Kommentare)	13
2.2 AUFBAU EINES NC-SATZES	14
2.2.1 Elemente des NC-Satzes und variable Satzlänge	14
2.2.2 Anmerkungen im Satz, Kommentare	16
2.2.3 Reihenfolge und Wiederholung von Befehlen / Wörtern im NC-Satz	17
2.2.4 Weglassen von Wörtern	17
2.3 AUFBAU EINES WORTES	17
2.3.1 Adressbuchstaben	18
2.3.2 Ziffernfolgen mit Dezimalpunkt	19
2.3.3 Ziffernfolgen ohne Dezimalpunkt	19
2.3.4 Satznummer: N-Wort	20
2.3.5 Wegbefehle	20
2.3.6 Koordinaten	21
2.3.7 Zusatzbefehle: M-Wort	22
2.4 SONDERZEICHEN	22
2.5 UNTERPROGRAMME	24
3 PRoNC SPRACHBESCHREIBUNG	25
3 PRoNC SPRACHBESCHREIBUNG	25
3.1 BEFEHLE NACH DIN 66025 IM NC-SATZ	31
3.1.1 Wegbefehle	31
3.1.1.1 Übersicht Wegbefehle in ProNC	32
3.1.1.2 Eilganggeschwindigkeit	34
3.1.1.3 Geradeninterpolation	35
3.1.1.4 Kreisinterpolation im Uhrzeigersinn	37
3.1.1.5 Kreisinterpolation entgegen dem Uhrzeigersinn	39
3.1.1.6 Verweilzeit	41
3.1.1.7 Eilganggeschwindigkeit mit Frameangabe	42
3.1.1.8 Segmentgeschwindigkeit mit Frameangabe	43
3.1.1.9 Helix im Uhrzeigersinn	45
3.1.1.10 Helix entgegen dem Uhrzeigersinn	46
3.1.1.11 alle Bewegungsbefehle	47
3.1.1.12 Definition der Interpolationsebene	48
3.1.1.13 Nullpunkt einrichten	49
3.1.1.14 Bahnfahrt	50
3.1.1.15 Definition Maßeinheit	52
3.1.1.16 Referenzfahrt	53
3.1.1.17 Teach	54
3.1.1.18 Bohrzyklus definieren	55
3.1.1.19 Bohrzyklus ausführen	57
3.1.1.20 Koordinatenangaben	58
3.1.1.21 Speicher setzen	59
3.1.1.22 Manipulation von Technologievariablen	60
3.1.1.23 Textausgabe	61
3.1.2 Zusatzbefehle	62

3.1.2.1	Programmunterbrechung	63
3.1.2.2	Programmbeginn, Programmende	64
3.1.2.3	Spindelbefehle	65
3.1.2.4	Kühlmittel	67
3.1.2.5	Werkstück spannen	67
3.1.2.6	Pumpe	68
3.1.2.7	Lampe	68
3.1.2.8	Peripherieoption	69
3.1.2.9	Hand-/Test-Modus	70
3.1.2.10	Eingänge lesen / Ausgänge rücklesen	71
3.1.2.11	Ausgänge setzen	72
3.1.2.12	Setzen Analog-/PWM-Ausgang	74
3.1.2.13	Aktuelle Achsposition	75
3.1.2.14	Aktuelle Systemzeit	75
3.1.2.15	Aktuelles Datum	76
3.1.2.16	Dialogfeld für Anwendertexteingabe	76
3.1.3	<i>FastVel-Befehl</i>	77
3.1.4	<i>F-Befehl</i>	77
3.1.5	<i>S-Befehl</i>	78
3.1.6	<i>Werkzeugwechsel</i>	79
3.1.7	<i>Unterprogrammtechnik</i>	80
3.1.7.1	Deklaration Unterprogramm	81
3.1.7.2	Aufruf Unterprogramm	82
3.2	ANWEISUNGEN: SYNTAKTISCHE ERWEITERUNGEN ZUR DIN 66025	85
3.2.1	<i>Variable</i>	85
3.2.1.1	P-Variablen	87
3.2.1.2	Q-Variablen	88
3.2.1.3	R-Variablen	90
3.2.1.4	Wertübergabe R-Variable an Koordinate	92
3.2.2	<i>Parameterrechnung</i>	93
3.2.2.1	arithmetische Ausdrücke	93
3.2.2.2	Funktionen	95
3.2.2.3	Bool'sche Ausdrücke	99
3.2.2.4	Zuweisungen	100
3.2.3	<i>Anweisungen zur Steuerung des Programmablaufes</i>	102
3.2.3.1	Bedingungen	102
3.2.3.2	Verzweigung	104
3.2.3.3	Auswahanweisung	105
3.2.3.4	Zählschleife	106
3.2.3.5	Schleife mit Test am Anfang	107
3.2.3.6	Schleife mit Test am Ende	109
3.2.4	<i>Anweisungen zur Kommunikation mit externen Geräten</i>	111
3.2.4.1	Abfrage eines Bediendialogs	111
3.2.4.2	Aktivierung beliebiger Anwenderprogramme	112
4	SYNCHRONISATION AUF DAS BEWEGUNGSENDE, INTEGRATION DES TEACH-IN	115
4.1	SYNCHRONISATION AUF DAS BEWEGUNGSENDE	115
4.2	INTEGRATION DES TEACH-IN	116
4.3	BEISPIEL FÜR EIN EINFACHES ANWENDERPROGRAMM MIT INTEGRATION DES TEACH-IN	119
5	AUSGEWÄHLTE LÖSUNGEN MIT PRONC	120
5.1	ISEL-XYZ-ANLAGEN / BELIEBIGE KARTESISCHE KINEMATIKEN	120
5.1.1	<i>Lernen</i>	120
5.1.2	<i>Figuren</i>	120
5.1.3	<i>Fräsen einer einfachen Kontur</i>	121
5.1.4	<i>Bohren</i>	122
5.1.5	<i>Taschen fräsen</i>	123
5.1.6	<i>Schrift mit Laser gravieren</i>	124
5.1.7	<i>Schweißen</i>	124
6	ZUSAMMENFASSUNG	126

6 ZUSAMMENFASSUNG.....	126
GLOSSAR.....	128
INDEX	129

1 Einführung

1 Einführung

Ziel dieser Dokumentation: Diese Dokumentation zum Programmpaket ProNC soll den Einstieg in die Nutzung dieser umfangreichen Software erleichtern und dazu beitragen, dass die in ProNC integrierten Programmiermöglichkeiten (Bewegungsanweisungen, Ein- und Ausgabeoperationen, Teach-In, Parameterrechnung, Unterprogrammtechnik, arithmetische und Bool'sche Verknüpfungen, Funktionen u.a.) umfassend genutzt werden können. Das Ziel besteht in einer Senkung der Programmier- und Inbetriebnahmezeiten bei der Umsetzung kundenspezifischer technologischer Aufgabenstellungen bei der Bearbeitung (spanende Bearbeitung, Schweiß- / Schneid- / Brenn- und Klebtechnologien) sowie im Handling- und Montagebereich.

1.1 Copyright

Urheberrecht: Alle Rechte an diesem Handbuch und dem Programmpaket ProNC, insbesondere das Urheberrecht, liegen bei

**isel-automation KG
Bürgermeister-Ebert-Straße 40
D - 36124 Eichenzell**

unerlaubte Vervielfältigung oder Weitergabe: Jede unerlaubte Vervielfältigung oder Weitergabe dieses Handbuches oder des Programmpaketes ProNC oder von Teilen daraus ist verboten und wird strafrechtlich verfolgt.

Nutzung von ProNC: Der rechtmäßige Erwerb der Installations-CD oder -Disketten einschließlich der Handbücher erlaubt die Nutzung vergleichbar der Nutzung eines Buches. Da es unmöglich ist, dass ein Buch an verschiedenen Orten von mehreren Personen gleichzeitig gelesen wird, darf das Programmpaket ProNC nicht auf verschiedenen Steuerungen (auf PC-Basis) an verschiedenen Orten von mehreren Personen gleichzeitig genutzt werden.

Sicherheitskopien: Es ist erlaubt, Sicherheitskopien des Programmpaketes ProNC anzufertigen. Diese Kopien dürfen aber keinesfalls Dritten zugänglich gemacht werden.

Schadenersatz: Bei Verstößen gegen das o.g. Copyright verpflichtet sich der Käufer dem Inhaber des Urheberrechtes, der Fa. isel-automation KG, zum Schadenersatz.

Haftung für Anwenderprogramme: Das Programmpaket ProNC sowie das vorliegende Handbuch wurden mit der gebotenen Sorgfalt erstellt. Alle gedruckten oder als Testprogramme auf CD zur Verfügung gestellten Anwenderprogramme wurden auf einer entsprechenden Hardware getestet. Es wird jedoch keinerlei Haftung oder Gewähr dafür übernommen, dass dieses Handbuch, das Programmpaket ProNC



oder die bereitgestellten Anwenderprogramme fehlerfrei oder für einen speziellen Zweck geeignet sind. Für Folgeschäden ist jede juristische Verantwortung oder Haftung ausgeschlossen.

Anregungen:

Da sich Fehler oder Missverständnisse niemals a priori ganz ausschließen lassen, sind wir für Anregungen, schriftliche Hinweise oder Meinungen jederzeit dankbar.
email: tech-support@isel.com

1.2 Zweckbestimmung des Programmpaketes ProNC

1.2.1 Kurzcharakteristik

Zweckbestimmung von ProNC:

Das Softwareprodukt ProNC integriert eine Bedienoberfläche entsprechend dem SAA-Standard und eine Programmierplattform zur Erstellung und zur Inbetriebnahme / Test von Anwenderprogrammen für CNC-gesteuerte Maschinen / Anlagen. Diese NC-Anwenderprogramme genügen der ISO-Syntax (G-Code-Programmierung nach DIN 66025) oder der PAL-Syntax (Programming Assembly Language). Zur standardisierten Syntax der ISO 1834 / DIN 66025 wurden funktionelle Erweiterungen vorgenommen.

Inbetriebnahmeunterstützung:

Besonderer Wert wurde auf eine effiziente Inbetriebnahmeunterstützung innerhalb der Testphase gelegt. Um diesem Anspruch zu entsprechen, wurden in ProNC Kommandos implementiert, die denen eines Debuggers entsprechen:

- **Anzeige von Prozess- und Real-Variablen in Echtzeit**
- **Manipulation von Real-Variablen**
- **Aktivierung / Deaktivierung aller Ein- und Ausgabeoperationen sowie der Spindelansteuerung mit Hilfe von Dummy-Funktionen**
- **Programmanimation**
- **Teach-In / effiziente Frame-Verwaltung und Manipulation**
- **Einzelstapfbetrieb**
- **Programminterpretation bis Unterbrechungspunkt**
- **Aktivierung / Deaktivierung von Unterbrechungspunkten**

Im Automatikbetrieb wird der Programmtest durch die Möglichkeit der Aktivierung von Unterbrechungspunkten auf beliebigen NC-Sätzen (Programmzeilen im Anwenderprogramm) sowie der Manipulation von aktuellen Werten der R-Variablen (Datentyp: Floating-Point) effizient unterstützt.

direktes Teach-In:



Das Teach-In kann **direkt** erfolgen, wenn keine selbsthemmenden Getriebe in der kinematischen Kette vorhanden sind. Dabei werden die entsprechenden Achsen mit stromlosen Motoren per Hand in die gewünschte Lage gebracht und der eingenommene Istwertvektor (Gelenkkordinaten bei nichtkartesischen Kinematiken) wird nach einer entsprechenden Koordinatentransformation als Kinematik unabhängige Datenstruktur (**Frame**) in der Geometriedatei (**Framefile**) abgespeichert.

indirektes Teach-In:



Beim indirekten Teach-In wird innerhalb einer übersichtlichen Dialog-Box mit Hilfe der Maus, mit Funktionstasten oder der isel-Bedientafel der Tool Center Point **TCP** (das ist die Werkzeugspitze bei Werkzeugmaschinen oder der Greifpunkt bei Handlingsystemen) in die gewünschte Zielposition / Zielorientierung zum Werkstück gefahren.

Innerhalb der Hierarchie der *isel*-Steuerungssoftware unter Windows 98/Windows NT/Windows 2000/Windows XP setzt die Bedien- und Programmieroberfläche ProNC auf einer Softwareebene auf, die nahezu ausschließlich aus **Dynamic Link Libraries (DLLs)** besteht. Diese Gerätetreiber-DLLs realisieren vor allem Bewegungssteuerungsmodule (**Motion Control MCTL**), Ein- und Ausgabemodule (**Input / Output IO**), Spindelmodule (**Spindle SPN**) und Werkzeugwechselmodule (**Tool Changer TCH**).

**Steuerungsmodule:
(Geräte-DLLs)**



Alle von *isel/automation* gelieferten **Steuerungsmodule** sind eigenständige Software-Produkte mit separaten Dokumentationen. ProNC verwaltet in der aktuellen Implementierung:

- **Zwei** Bewegungssteuerungsmodule (**MCTL**) mit je maximal **6 Achsen** (Bewegungserzeugung, d.h. Interpolation und Erzeugung eines Geschwindigkeitsprofils / Slope und Bewegungsrealisierung, d.h. z.B. bei DC/AC-Servoantrieben digitale Lageregelung)
- **Vier** Ein- und Ausgabemodule (**IO**) mit je **4 Eingangs- bzw. 4 Ausgangsports (je 32 Eingänge / 32 Ausgänge)**
- **Vier** Spindelmodule (**SPN**)
- **Zwei** Werkzeugwechselmodule (**TCH**) mit **max. 128 Werkzeugen**

**Steuerungsmodul-
Konzept:**



Alle **Steuerungsmodule eines Gerätetyps** besitzen das gleiche Beauftragungsinterface und eine annähernd gleiche Funktionalität. Das bringt für den Anwender folgende Vorteile:

1. Mit der Investition in die Bedien- und Programmieroberfläche ProNC können sowohl Anlagen mit Schrittmotoren als auch Anlagen / Kinematiken mit DC/AC-Servomotoren bedient und programmiert werden, wenn für die Bewegungssteuerung die entsprechenden Steuerungsmodule (**MCTL-DLLs**) bereitgestellt sind.
2. Es können Anlagen mit maximal 4 Arbeitsspindeln programmiert werden.
3. Die erstellten Anwenderprogramme können zwischen verschiedenen Anlagen ausgetauscht bzw. übertragen werden. Alle anlagenspezifischen Details (z.B. Spindelsteigung, Getriebeuntersetzung, Geschwindigkeiten, Beschleunigung, Portadressen u.a.) werden in der jeweiligen Initialisierungsdatei des jeweiligen **Steuerungsmoduls** spezifiziert. Es ist nicht notwendig, Änderungen im Quellprogramm vorzunehmen.

1.2.2 Grundsätze einer Programmierung mit ProNC

ProNC wurde als Komponente der isel-Steuerungssoftware für Maschinen / Anlagen mit maximal **12 Achsen (2 Achssysteme zu je maximal 6 Achsen)** implementiert.

ProNC ist generell unter Windows 98/Windows NT/Windows 2000/Windows XP lauffähig. Es ist jedoch möglich, dass bestimmte **Steuerungsmodule (Geräte-DLLs)** nur unter **Windows 98** oder **Windows NT/2000/XP** genutzt werden können.

ProNC ist die Portierung der bisher ausschließlich für das PC-Betriebssystem MS-DOS angebotenen Steuerungsprogramme Remote, Pro-DIN und Pro-PAL. Daraus folgt, dass mit ProNC bisher unter MS-DOS erstellte und genutzte Anwenderprogramme im NCP-Format (von Remote), ISO-Format (von Pro-DIN) bzw. PAL-Format (von Pro-PAL) weiterhin nutzbar sein müssen.

ProNC ermöglicht sowohl die Programmierung im ISO- / DIN-Format als auch im PAL-Format. Im Kapitel 3 dieser Programmervorschrift wird die Syntax immer vergleichend dargestellt. Das heißt, dass nach „**ISO:**“ immer ein NC-Satz / Befehl entsprechend ISO-Syntax folgt bzw. dass nach „**PAL:**“ immer ein NC-Satz / Befehl entsprechend PAL-Syntax angegeben ist.

Die technologieorientierte Syntax der DIN 66025 (G- und M-Befehle) wurde um problemorientierte Konstruktionen zur Strukturierten Programmierung, zur Parameterrechnung sowie zum Zugriff auf Geometriedateien (**Geometriedatei = Framedatei**) ergänzt und als flexibles, leistungsfähiges Programmierniveau definiert. Dieses Programmierniveau ist als Grammatik im Abschnitt 3 dieser Dokumentation beschrieben.

Das Programmpaket **ProNC** löst die Programmpakete **Remote**, **Pro-DIN** und **Pro-PAL** ab. Entsprechend der Philosophie dieser „Vorgängerprogramme“ werden technologische Parameter (Spindelsteigung / Getriebeübersetzung, Referenzfahrtgeschwindigkeit, Softwareendschalter, Schaltpegel u.a.) nicht im Quellprogramm in einem sogenannten Deklarationsteil spezifiziert, sondern werden im Maschinendatensatz / Maschinen-Parameter-Datei (allgemein Initialisierungsdatei des Bewegungssteuerungsmodules) definiert.

Vorteil: Initialisierungsdatei des Bewegungs- steuerungsmodules



Zum Beispiel die konkrete Getriebeübersetzung wird lediglich bei der Konfigurierung einer Anlage einmalig mit Hilfe eines Dialogprogramms in die Initialisierungsdatei des Bewegungssteuerungsmoduls eingetragen oder geändert, wenn z.B. an der Anlage die vorhandene Kugelumlaufspindel durch eine Spindel mit anderer Steigung ersetzt wird. Der Vorteil besteht darin, dass die Quellprogramme immer portabel sind, d.h. technologische Details sind immer in der Konfigurierungsdatei (Initialisierungsdatei) des Bewegungssteuerungsmoduls „versteckt“:

In ProNC werden grundsätzlich zwei Typen von Programmzeilen unterschieden:

Eine Programmzeile kann sein:

- ein **NC-Satz** (speziell in der DIN 66025 definiert), z.B.: **G1 X100 Y150 Z-50**
- eine **Anweisung** (nicht in der DIN 66025 definiert), z.B.: **While R1 > 0.0**

[siehe auch Bedienungsanleitung:](#)

5.8.7 Menü Einstellungen - Steuerung

ProNC basiert auf der Erfahrung, dass mit den Normen **DIN 66025** in Deutschland bzw. **ISO/DIS 6983/1** international vorgegeben ist, wie numerisch gesteuerte Arbeitsmaschinen programmiert werden können. Die **ISO-Syntax** ist auf technologische Anforderungen optimiert

und durch die ausschließliche Verwendung von Buchstaben des lateinischen Alphabetes zur Kennzeichnung sowohl von Aktivitäten als auch von Parametern gekennzeichnet. Die **PAL-Syntax** basiert auf der ISO-Syntax mit dem Kennzeichen, dass die kompakten G- und M-Befehle durch **mnemonische Codes** (Mnemoniks) ersetzt werden. Im folgenden Beispiel entspricht der ISO-Syntax **G90 G1** die PAL-Syntax **MOVEABS**:

ISO: N10 **G90 G1** X100 Y200 Z-50
 PAL: N10 **MOVEABS** X100 Y200 Z-50

**ProNC-
 Programmaufbau:**



Ein **ProNC-Programm** besteht aus **NC-Sätzen** und / oder **Anweisungen**. Alle NC-Sätze (oder kurz **Sätze**) bestehen aus Worten, die häufig auch als **Befehle** bezeichnet werden. Jedes Wort / jeder Befehl beginnt mit dem sogenannten Adressbuchstaben, gefolgt von einer Ziffernfolge, mit oder ohne Vorzeichen sowie mit oder ohne Dezimalpunkt.

NC-Sätze in einem ISO-Programm können G-Befehle (z.B. G1) und / oder M-Befehle (z.B. M3) enthalten.

NC-Sätze in einem PAL-Programm können mnemonische Befehle (z.B. MOVEABS oder SCLW) enthalten .

<u>Beispiel für Befehle</u>	<u>ISO-Syntax</u>	<u>PAL-Syntax</u>
Bewegungsbefehl (Zielangabe absolut)	G90 G1	MOVEABS
Einschalten/Hochtoure der Bearbeitungsspindel	M3	SCLW

Der wesentliche Unterschied wird offensichtlich:

1.A Programmzeilenstruktur bei ISO-Syntax:

Benutzung von G- und M-Befehlen: Bei der Programmierung nach ISO-Syntax können Wegbefehle (G-Befehle), Vorschübe (F-Befehl), Zusatzbefehle (M-Befehle) und andere Befehle **kombiniert** und je Befehls-Typ **auch mehrfach** in einer Programmzeile enthalten sein.

Bei der ISO-Programmierung werden bei NC-Sätzen ausschließlich Befehle mit einem führenden Großbuchstaben (Adressbuchstaben) benutzt.

1.B Programmzeilenstruktur bei PAL-Syntax:

Benutzung von mnemonischen Befehlen: Bei der Programmierung nach PAL-Syntax werden bei NC-Sätzen als Wegbefehle und Zusatzbefehle ausschließlich mnemonische Befehle benutzt.

Vorteil der Programmierung in ProNC: Damit ergeben sich bei der Programmierung mit ProNC sehr kompakte und regelmäßige Programme, die sich in der Praxis (bei Applikation der ISO-Syntax) vor allem bei der Programmierung numerisch gesteuerter Werkzeugmaschinen international durchgesetzt und bewährt haben.

Modalität:

Modalität heißt, dass ein spezifizierter Wert (Koordinate, Geschwindigkeit oder Wegbefehl) im Programmkontext so lange erhalten bleibt, bis dieser Wert neu definiert worden ist.



Eine spezifizierte Geschwindigkeit bleibt so lange eingestellt, bis eine neue Geschwindigkeitsangabe im Programm erfolgt.

Eine spezifizierte Koordinate (Bewegungsziel) gilt so lange, bis eine neue Koordinatenangabe im Programm erfolgt. Daraus ist abzuleiten, dass innerhalb eines Bewegungssatzes nur die Koordinaten angegeben werden, die eine (absolute oder relative) Bewegung im betreffenden Satz bewirken sollen.

Modalität bei ProNC:

Bei der Programmierung sowohl nach ISO- als auch nach PAL-Syntax wirken **sowohl Wegbefehle** (ISO: G-Befehle, PAL: Wegbefehle) **als auch Koordinaten-Worte** (z.B. X, Y, Z, U, V, W, A, B oder C) modal:



Der NC-Satz

ISO: N001 **G90 G1** X100 Y200 Z300
bzw.
PAL: N001 **MOVEABS** X100 Y200 Z300

definiert mit Hilfe der G-Befehle **G90 G1** bzw. des PAL-Wegbefehles **MOVEABS** eine Geradeninterpolation. Diese Definition der Interpolationsart ist modal bzw. „selbsthaltend“. Damit ist diese Interpolationsart auch im darauffolgenden Satz

ISO: / PAL:
N002 X150 Y250 Z350

gültig und muss nicht explizit angegeben werden.

[siehe auch:](#)
Abschnitt 3 Sprachbeschreibung

Vorschau Kapitel 2:

Kapitel 2 dieses Handbuches stellt die wichtigsten Vorschriften der ISO- bzw. PAL-Syntax zusammen.

Vorschau Kapitel 3:

In **Kapitel 3** ist die komplette Sprachbeschreibung von ProNC (**ISO-Syntax** verglichen mit der **PAL-Syntax**) enthalten. Es stellt das umfangreichste Kapitel der Dokumentation dar.

Vorschau Kapitel 4:

Die Integration von Geometrieinformationen in das Anwenderprogramm und damit der Zugriff auf Geometriedaten (**Frames**) zur Laufzeit des Anwenderprogramms wird u. a. in **Kapitel 4** beschrieben.

Vorschau Kapitel 5:

Dieses Kapitel beschreibt Einstiegs-Anwenderprogramme, die auf jeder Anlage mit mindestens zwei Achsen getestet werden können.

2 Programmierung mit ProNC

2 Programmierung mit ProNC

Die Ausführungen in diesem Abschnitt beziehen sich immer sowohl auf die Anwendung der ISO-Syntax als auch der PAL-Syntax.

Worin besteht der Unterschied zwischen der ISO-Syntax und der PAL-Syntax ?

Einzig und allein in der Ersetzung von G- und M-Befehlen der ISO-Syntax durch mnemonische Befehle (mnemonische Wegbefehle und mnemonische Zusatzbefehle) bei der PAL-Syntax:

<u>Befehle nach ...</u>	<u>ISO-Syntax:</u> G- und M-Befehle	<u>PAL-Syntax:</u> mnemonische Befehle
Bewegungsbefehl (Zielangabe absolut)	<i>G90 G1</i>	<i>MOVEABS</i>
Befehl zum Einschalten der Arbeitsspindel	<i>M3</i>	<i>SCLW</i> <i>SPINDLE ON</i>

Hinweis:

Falls es zu einem Befehl keine Entsprechung im ISO-Code gibt, ist die Verwendung des Befehls in PAL-Syntax erlaubt.

Sobald das Programm einen Befehl in ISO-Syntax enthält, muss es als ISO-Programm deklariert werden.



**ProNC-
Programmierung:**

Bei der ProNC-Programmierung sind alle **Anweisungen** bei der ISO-Syntax und bei der PAL-Syntax identisch.

Das heißt, dass z.B. eine FOR-Schleife immer der gleichen Syntax genügt, aber NC-Sätze innerhalb einer FOR-Schleife entweder der ISO-Syntax oder der PAL-Syntax entsprechen müssen.

siehe auch:
Abschnitt 3.2 Anweisungen

2.1 Aufbau des Steuerprogramms

**Bestandteile eines
Anwender-
programms:**

Ein Anwenderprogramm besteht immer aus einem Hauptprogramm und keinem, einem oder mehreren Unterprogrammen. Unterprogramme werden vor dem Hauptprogramm deklariert.

2.1.1 Programmstruktur (Hauptprogramm)

Hauptprogramm:

Ein Hauptprogramm besteht aus einer Folge von NC-Sätzen und Anweisungen, wobei der erste und der letzte NC-Satz eines Hauptprogramms fest vorgeschrieben sind.

Die folgende Tabelle zeigt die einfache Struktur eines Hauptprogramms:

Eigenschaft	<u>syntaktische Kennzeichnung nach ISO-Syntax</u>	<u>syntaktische Kennzeichnung nach PAL-Syntax</u>
Programm-anfangssatz	durch das Sonderzeichen % Beispiel: %123	durch die Mnemonik PROGBEGIN Beispiel: ProgBegin
Folge von Sätzen, die den eigentlichen Programmkörper bilden	Beispiel: N0 G74 N1 G1 X100 Y200 Z300 N2 X200 Y300 Z400	Beispiel: N0 REF N1 MOVEABS X100 Y200 Z300 N2 X200 Y300 Z400
Programm-endesatz	durch den Zusatzbefehl M30	durch den Zusatzbefehl PROGEND

Tabelle 2.1.1: Struktur eines Hauptprogramms

Kennzeichnung des Programmanfanges: Vor dem Sonderzeichen % oder der Mnemonik **ProgBegin** zur Kennzeichnung des Programmanfanges können Unterprogramme oder beliebig viele Kommentare stehen.

siehe auch:

Abschnitt 2.1.2 Anmerkungen im Programm

Abschnitt 2.5 Unterprogramme

2.1.2 Anmerkungen im Programm (Kommentare)

In einem Anwenderprogramm erhöhen Kommentare den Dokumentationswert und erleichtern damit die Programminbetriebnahme bzw. den Programmtest und die Programmpflege. In ProNC werden vier Arten von Kommentaren unterschieden:

- Kommentare, die sich über mehrere Zeilen erstrecken, müssen entsprechend ISO-Syntax mit dem Sonderzeichen (beginnen und mit dem Sonderzeichen) enden, entsprechend PAL-Syntax wird { bzw. } benutzt.
- Kommentare, die als Trennzeichen behandelt werden sollen, müssen ebenfalls entsprechend der ISO-Syntax mit dem Sonderzeichen (beginnen und mit dem Sonderzeichen) enden, entsprechend PAL-Syntax wird { bzw. } benutzt.

Kommentare in runden (ISO) bzw. geschweiften {PAL} Klammern: Kommentare, die in runden bzw. geschweiften Klammern eingeschlossen sind, werden vom Compiler immer aus der Quelldatei gefiltert und demnach **nicht** in die CNC-Zieldatei übernommen. Dadurch wird die CNC-Zieldatei kompakter.



Ein Kommentar in runden bzw. geschweiften Klammern kann sich über beliebig viele Zeilen erstrecken. Im Gegensatz zur Einschränkung in der DIN 66025 können Kommentare in ProNC alle Zeichen des ASCII-Zeichenvorrates (also auch die Sonderzeichen % und :) enthalten.

ISO-Syntax: (das ist ein Kommentar)

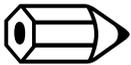
PAL-Syntax: { das ist ein Kommentar }

Ein Kommentar kann sich auch über mehrere Zeilen erstrecken. Dies hat einen großen Vorteil:

Bei der Inbetriebnahme können ganze Programmabschnitte „auskommentiert“ werden. Das heißt, beliebig lange Folgen von NC-Sätzen werden durch Klammer-Setzung zu Kommentar und damit bei der Programminterpretation als ein oder mehrere „leere Sätze“ überlesen und damit ignoriert.

- Kommentare, die sich über eine ganze Zeile bis zum Zeilenende erstrecken, müssen mit einem Semikolon ; beginnen und mit dem Zeilenendezeichen CR = Carriage Return (ENTER-Tastencode) enden.

Kommentare über eine ganze Zeile: Diese Kommentare werden vom Compiler dann **nicht** aus der Quelldatei gefiltert und damit in die CNC-Zieldatei übernommen, wenn das Kommentarfilter (Option des Compilers) ausgeschaltet ist.



Das Semikolon „;“ zur Kennzeichnung eines ganzzelligen Kommentars muss in der ersten Spalte der Kommentarzeile stehen.

Beispiel:

; das ist ein Kommentar von Spalte 1 bis Zeilenende

- Kommentare, die einen NC-Satz abschließen, beginnen mit einem Semikolon „;“ nach dem letzten Zeichen des NC-Satzes und enden mit dem Zeilenendezeichen CR.

Kommentare als Abschluss eines NC-Satzes: Kommentare als Abschluss eines NC-Satzes werden vom Compiler immer aus der Quelldatei gefiltert.

Beispiel:



N10 G1 G91 X100 ; Relativbewegung 100mm in der X-Achse

2.2 Aufbau eines NC-Satzes

2.2.1 Elemente des NC-Satzes und variable Satzlänge

NC-Satz: Ein NC-Satz besteht aus mehreren Befehlen (auch Befehls-Worte oder nur Worte genannt), wobei ein gültiger Befehl eines Satzes dadurch gekennzeichnet ist, dass dessen erstes Zeichen immer ein Großbuchstabe ist. Dieser Anfangsbuchstabe eines Befehles/Wortes wird auch *Adressbuchstabe* genannt.

Wort als Synonym für **Befehl:** Im Sprachgebrauch der NC-Programmierung wird sehr oft für **Wort** das Synonym **Befehl** benutzt. Das heißt, dass zum Beispiel Wegbefehle sowohl als G-Worte als auch als G-Befehle bezeichnet werden.

Bei der **ISO-Syntax** werden vor allem **G- und M-Befehle** benutzt, um Wegbefehle und Zusatzbefehle zu definieren.

Bei der **PAL-Syntax** dienen **mnemonische Befehle** zur Notation von Wegbefehlen und Zusatzbefehlen.

Der spezielle Großbuchstabe (Adressbuchstabe), der jedes Wort eines NC-Satzes einleitet, gibt dem Wort einen „Namen“:

ISO-Syntax:

<u>Adressbuchstabe</u>	<u>Wort / Befehl</u>	<u>Bedeutung</u>
N	N-Wort = N-Befehl	Satznummer
G	G-Wort = G-Befehl	Wegbefehl
M	M-Wort = M-Befehl	Zusatzbefehl
E	E-Wort = E-Befehl	Vorschub (Eilgang)
F	F-Wort = F-Befehl	Vorschub (Normal)
T	T-Wort = T-Befehl	Werkzeugnummer
S	S-Wort = S-Befehl	Spindeldrehzahl

Tabelle 2.2.1: Auswahl wichtiger Worte in NC-Sätzen (ISO-Syntax)

PAL-Syntax:

<u>Adressbuchstabe</u>	<u>Wort / Befehl</u>	<u>Bedeutung</u>
N	N-Wort = N-Befehl	Satznummer
	Mnemonischer Weg-Befehl, z.B.: MOVEABS	Wegbefehl
	Mnemonischer Zusatz- Befehl, z.B.: SETBIT	Zusatzbefehl
F	F-Wort = F-Befehl	Vorschub
T	T-Wort = T-Befehl	Werkzeugnummer
S	S-Wort = S-Befehl	Spindeldrehzahl

Tabelle 2.2.2: Auswahl wichtiger Worte (mnemonischer Befehle) in NC-Sätzen (PAL-Syntax)

Trennzeichen: Die Befehle / Worte eines Satzes werden durch Trennzeichen voneinander getrennt. Als Trennzeichen werden in ProNC erlaubt:

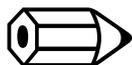
- ein oder mehrere Leerzeichen
- ein oder mehrere Tabulatoren
- Kombinationen von Leerzeichen und Tabulatoren
- ein Kommentar

Länge eines NC-Satzes:

Entsprechend der Möglichkeit, dass die Anzahl von Worten in einem Satz nicht vorgeschrieben ist, ist die Länge eines NC-Satzes variabel.

gültige NC-Sätze:

; Referenzpunktfahrt:
ISO: N1 G74
PAL: N1 REF



; relative Wegvorgabe:
ISO: N2 G1 G91 X100.0 Y200.1 Z300.234 F200.23
PAL: N2 MOVEREL X100.0 Y200.1 Z300.234 F200.23

; absolute Wegvorgabe, Spindeldrehzahl in U/min und Spindel ein:
ISO: N3 G1 G90 X100.0 S15000 M3
PAL: N3 MOVEABS X100.0 S15000 SCLW

Modalität:

Bei der Betrachtung der Länge eines NC-Satzes kommt die Modalität zum Tragen. Das heißt, alle Wegbefehle (**ISO:** G-Befehle, **PAL:** mnemonische Wegbefehle), die im Satz n angegeben wurden und im Satz n+1 ebenfalls gültig sind, müssen im Satz n+1 nicht explizit angegeben werden:

Modalität im NC-Satz:

Der Wegbefehl **G1 G90** | **MOVEABS** (Geradeninterpolation, absolute Wegangabe) wird im Satz N001 spezifiziert und bleibt im Satz N002 wirksam. Erst ab dem Satz N003 wird durch die Angabe des Wegbefehles **G91** | **MOVEREL** wieder die relative Wegvorgabe „eingeschaltet“:



ISO:
N001 G1 G90 X100 Y200
N002 X150 Y250
N003 G1 G91 X10 Y20

PAL:
N001 MOVEABS X100 Y200
N002 X150 Y250
N003 MOVEREL X10 Y20

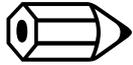
[siehe auch:](#)
Abschnitt 2.3 Aufbau eines Wortes

2.2.2 Anmerkungen im Satz, Kommentare

Ein Kommentar wird wie ein Trennzeichen behandelt, wenn er in runde Klammern (ISO-Syntax) bzw. geschweifte Klammern (PAL-Syntax) eingeschlossen ist. Demnach kann ein Kommentar auch zwischen zwei Worten stehen:

**Kommentar als
Trennzeichen:**

gültiger NC-Satz mit Kommentar als Trennzeichen:



ISO:

N10 G1 X100 Y200 Z300 (Geschwindigkeitsangabe) F1000

PAL:

N10 MOVEABS X100 Y200 Z300 {Geschwindigkeitsangabe} F1000

2.2.3 Reihenfolge und Wiederholung von Befehlen / Wörtern im NC-Satz

Die Reihenfolge der einzelnen Worte in einem Satz ist wie folgt vorgeschrieben:

Syntax	1	2	3	4	5	6	7
	N-Wort	ISO: G-Befehl PAL: Mnemo- nischer Befehl	Koordinaten- wörter: X/Y/Z- U/V/W- A/B/C- Wort	I-Wort J-Wort K-Wort	F-Wort	S-Wort	ISO: M-Befehl PAL: Mnemo- nischer Befehl
	Satz- nummer	Weg- bedingung	Ziel- koordinaten	Interpo- lationspa- rameter	Vorschub	Spindel- drehzahl	Zusatz- funktion
ISO:	N100	G91 G2	X100	I50	F75	S10000	M111
PAL:	N100	CWREL	X100	I50	F75	S10000	SETB A1.1

In ProNC ist es erlaubt, dass in einem Satz mehrere Wegbefehle (G-Befehle) und mehrere Zusatzbefehle (M-Befehle) enthalten sind.

2.2.4 Weglassen von Wörtern

In der Norm DIN 66025 wird die Problematik der Modalität wie folgt beschrieben :

Modalität:

“Ein Wort, das sich in mehreren aufeinanderfolgenden Sätzen eines Steuerprogramms in seiner Wirkung nicht ändert, braucht nur einmal angegeben zu werden und kann in allen folgenden Sätzen, für welches es unverändert gelten soll, weggelassen werden.“

2.3 Aufbau eines Wortes

Ein Wort im Sinne der DIN- / ISO-Norm besteht aus einem Adressbuchstaben, gefolgt von einer Zahl (in der DIN-Norm wird der Begriff „Ziffernfolge“ benutzt):

Worte:

G99 ist ein gültiges G-Wort (Wegbefehl).
N88 ist ein gültiges N-Wort (Satznummer).



GG_100 ist ein ungültiges Wort.
N?88 ist ein ungültiges Wort.

natürliche oder Dezimalzahlen:

Die Zahl kann **natürlich** oder eine **Dezimalzahl** sein. Es gibt eine feste Zuordnung von natürlichen bzw. Dezimalzahlen zu den Adressbuchstaben:
 Bei D-, G-, L-, M-, N-, S- oder T-Befehlen folgt dem Adressbuchstaben immer eine natürliche Zahl.
 Die Dezimalzahl kann vorzeichenbehaftet (**vzb**) sein. Bei einer positiven Zahl kann das Vorzeichen **+** weggelassen werden:
+1.0 ist identisch mit **1.0**

2.3.1 Adressbuchstaben

Adressbuchstaben: Durch die Adressbuchstaben wird das Wort in seiner Bedeutung eindeutig spezifiziert.

Es ist offensichtlich, dass entsprechend der 26 Buchstaben des lateinischen Alphabetes 26 verschiedene DIN-/ISO-Worte möglich sind (**vzb**: vorzeichenbehaftet):

<u>Adressbuchstabe</u>	<u>DIN 66025</u>	<u>ProNC</u>	<u>zugeordnete Zahl</u>
A	Drehbewegung um die X-Achse	Drehbewegung um die X-Achse	Dezimalzahl
B	Drehbewegung um die Y-Achse	Drehbewegung um die Y-Achse	Dezimalzahl
C	Drehbewegung um die Z-Achse	Drehbewegung um die Z-Achse	Dezimalzahl
D	Werkzeugkorrekturspeicher	nicht benutzt	natürlich
E	Vorschub (Eilgang)	Vorschub (Eilgang)	Dezimalzahl
F	Vorschub (normal)	Vorschub (normal)	Dezimalzahl
G	Wegbefehl	Wegbefehl	natürlich
H	nicht benutzt	nicht benutzt	
I	Interpolationsparameter zur X-Achse	Interpolationsparameter zur X-Achse	Dezimalzahl
J	Interpolationsparameter zur Y-Achse	Interpolationsparameter zur Y-Achse	Dezimalzahl
K	Interpolationsparameter zur Z-Achse	Interpolationsparameter zur Z-Achse	Dezimalzahl
L	frei verfügbar	Unterprogramm-kennzeichnung	natürlich
M	Zusatzbefehl	Zusatzbefehl	natürlich
N	Satznummer	Satznummer	natürlich
O	nicht benutzt	nicht benutzt	

P	Parameter für spezielle Berechnungen	Kennzeichnung der P-Variablen	natürlich
Q	Parameter für spezielle Berechnungen	Kennzeichnung der Q-Variablen	natürlich
R	Parameter für spezielle Berechnungen	Kennzeichnung der R-Variablen	natürlich
S	Spindeldrehzahl	Spindeldrehzahl	natürlich
T	Werkzeug	Werkzeug	natürlich
U,V,W	zweite Bewegung parallel zur X,Y,Z-Achse	Reserviert für ProNC-Version für $2 * 9 = 18$ Achsen	Dezimalzahl
X	Bewegung in Richtung der X-Achse	Bewegung in Richtung der X-Achse	Dezimalzahl
Y	Bewegung in Richtung der Y-Achse	Bewegung in Richtung der Y-Achse	Dezimalzahl
Z	Bewegung in Richtung der Z-Achse	Bewegung in Richtung der Z-Achse	Dezimalzahl

Tabelle 2.3.1: Adressbuchstaben und deren Bedeutung nach DIN 66025 und innerhalb ProNC

2.3.2 Ziffernfolgen mit Dezimalpunkt

In ProNC wird bei Dezimalzahlen (DIN 66025: Ziffernfolgen mit explizitem Dezimalpunkt) die Unterdrückung sowohl führender als auch nachfolgender Nullen erlaubt. Damit wird die Forderung der DIN 66025 Teil1 erfüllt.

zulässige Dezimalzahlen: X300. entspricht der Koordinatenangabe X300.0
Y.3 entspricht der Koordinatenangabe Y0.3

Compiler: Der Compiler (für die ISO- oder PAL-Syntax) schreibt in die CNC-Zieldatei immer Dezimalzahlen mit einer führenden Null und bei Angabe eines expliziten Dezimalpunktes immer drei Stellen nach dem Dezimalpunkt.



2.3.3 Ziffernfolgen ohne Dezimalpunkt

Ziffernfolgen ohne Dezimalpunkt (**natürliche Zahlen**) werden bei allen G-Worten, L-Worten, M-Worten, N-Worten und P/Q/R-Worten (Variable) gefordert.

Vorteil bei ProNC: Bei allen anderen in ProNC unterstützten Adressbuchstaben (Koordinaten, Interpolationsparametern, Vorschub u.a.) werden ganze Zahlen (d.h. Ziffernfolgen ohne Dezimalpunkt) grundsätzlich entsprechend DIN 66025 akzeptiert. Das heißt, dass bei einer Koordinatenangabe X100 nicht explizit X100.0 geschrieben werden muss.



2.3.4 Satznummer: N-Wort

Satznummer:



Die dem Adressbuchstaben **N** folgende natürliche Zahl gibt die Satznummer eines NC-Satzes an. Innerhalb ProNC werden bezüglich der Nummer selbst keine Bedingungen gestellt. Das heißt, dass eine bestimmte Nummer beliebig oft auftreten darf. Die Nummerierung muss nicht unbedingt in aufsteigender Reihenfolge erfolgen. Es hat sich bewährt, beim ersten Programmentwurf die Satznummern mit einer Differenz von 5 oder 10 in das Quellprogramm einzugeben, um bei späteren Korrekturen NC-Sätze an relevanten Stellen im Programm einfügen zu können. Vor Anweisungen zur Steuerung des Programmablaufes stehen keine Satznummern.

[siehe auch:](#)

Abschnitt 3.2.3 Anweisungen zur Steuerung des Programmablaufes

Satzunterdrückung:

Im Anwenderprogramm kann jeder beliebige NC-Satz während der Programmabarbeitung unterdrückt werden, wenn dem N-Wort dieses Satzes das Satzunterdrückungszeichen „/“ vorangestellt ist. Die Aktivierung der Satzunterdrückung wird an der Bedientafel der Steuerung oder im Bediendialog (Bildschirm orientierte Bedienung) vorgenommen.



Beispiel:

Der folgende NC-Satz nach ISO-Syntax wird während der Programmabarbeitung übersprungen, wenn die **Satzunterdrückung** der CNC-Steuerung aktiv ist:

/ N10 S1=1000 M3 ; NC-Satz mit wahlweiser Verarbeitung

Programmtest und Satznummer:



Es wird darauf hingewiesen, dass beim Programmtest immer zeilenorientiert gearbeitet wird. Das heißt, dass z.B. ein Unterbrechungspunkt immer auf eine bestimmte Programmzeile und nie auf eine bestimmte NC-Satznummer gesetzt wird. Denn eine Programmzeile 100 ist immer nur einmal vorhanden, eine Satznummer 100 dagegen kann n-mal im ISO- oder PAL-Quellprogramm vorkommen.

[siehe auch:](#)

Bedienungsanleitung 5.6.3 Menü Ausgabe - Satzvorlauf

2.3.5 Wegbefehle

ISO-Syntax: G-Befehle

Die Zahl, die dem Adressbuchstaben **G** folgt, ist eine natürliche Zahl und wird laut DIN 66025 Teil1 auch als Schlüsselzahl bezeichnet. Die in ProNC realisierten Wegbefehle (G-Worte bzw. G-Befehle) werden ausführlich im Abschnitt 3.1.1 Wegbefehle vorgestellt.

PAL-Syntax: mnemonische Weg-Befehle

Die laut PAL-Syntax verfügbaren mnemonischen Weg-Befehle besitzen alle einen entsprechenden G-Befehl oder eine Kombination von G-Befehlen. Die Gegenüberstellung zeigt die Tabelle im Abschnitt 3.1.1 Wegbefehle.

[siehe auch:](#)

Abschnitt 3.1.1 Wegbefehle

2.3.6 Koordinaten

Als Adressbuchstaben für Koordinaten-Worte werden in ProNC die Buchstaben **X, Y, Z, A, B, C, U, V** und **W** reserviert. Damit sind neun numerische Achsen pro Achssystem programmierbar. Einem der genannten Adressbuchstaben kann sowohl eine ganze Zahl als auch eine Dezimalzahl folgen.

Achssysteme in ProNC:

ProNC kann **zwei** Achssysteme verwalten.

Jedes Achssystem kann in der aktuellen ProNC-Version maximal die 6 numerischen Achsen X, Y, Z, A, B und C umfassen.



Wird nur ein Achssystem programmiert, besteht keine Notwendigkeit, die Koordinaten-Worte zu unterscheiden.

Werden dagegen die **zwei** Achssysteme

- Achssystem 1
- Achssystem 2

in einem Anwenderprogramm programmiert, muss eine Unterscheidung zwischen dem X-Koordinaten-Wort des ersten und dem X-Koordinaten-Wort des zweiten Achssystems möglich sein. Diese Unterscheidung erfolgt durch Indizierung der Adressbuchstaben:

- **Koordinaten-Worte im Achssystem 1:**
Xdezimalzahl oder **X1=dezimalzahl**
Ydezimalzahl oder **Y1=dezimalzahl**
Zdezimalzahl oder **Z1=dezimalzahl**
Adezimalzahl oder **A1=dezimalzahl**
Bdezimalzahl oder **B1=dezimalzahl**
Cdezimalzahl oder **C1=dezimalzahl**
- **Koordinaten-Worte im Achssystem 2:**
X2=dezimalzahl
Y2=dezimalzahl
Z2=dezimalzahl
A2=dezimalzahl
B2=dezimalzahl
C2=dezimalzahl

Dem Adressbuchstaben X, Y, Z, A, B, C folgt der Index des Achssystems (1 oder 2), darauf das Gleichheitszeichen ‚=‘ und danach die Dezimalzahl (oder ein arithmetischer Ausdruck).

Die Achsbenennung (Zuordnung der Adressbuchstaben der Koordinaten-Worte) zur numerischen Achse im mechanischen System ist an die Normen **DIN 66025** und **VDI 2861** angepasst:

Bei Werkzeugmaschinensteuerungen werden sechs translatorische (X, Y, Z, U, V, W) und drei rotatorische Achsen (A, B, C) definiert.

Werkzeugmaschinensteuerungen: (DIN 66025, DIN 66217)	
translatorische Achsen (Linearachsen): • Haupt-Achsen:	X- , Y- und Z-Achse bilden ein rechtshändiges, rechtwinkliges Koordinatensystem. Die Z-Achse ist identisch mit der Achse der Bearbeitungsspindel. Die positive Richtung der Z-Achse verläuft vom Werkstück zum Werkzeug.
• Hilfs- oder Nebenachsen:	U-Achse parallel zur X-Achse V-Achse parallel zur Y-Achse W-Achse parallel zur Z-Achse } reserviert für ProNc mit 9 Achsen je Achssystem
rotatorische Achsen (Drehachsen):	A-Achse dreht wie eine Rechtsschraube um die X-Achse. B-Achse dreht wie eine Rechtsschraube um die Y-Achse. C-Achse dreht wie eine Rechtsschraube um die Z-Achse.

Tabelle 2.3.6: Achsbezeichnung nach DIN 66025 (Werkzeugmaschinensteuerungen)

Die Dezimalzahlen, die den Koordinaten-Worten unmittelbar folgen, stellen Absolutwerte / Absolutmaß (Wegbefehl **G90** | **ABS** - selbsthaltend) oder Relativwerte / Kettenmaß (Wegbefehl **G91** | **REL** - selbsthaltend) dar.

Die Maßeinheit einer translatorischen Achse ist **mm** (Wegbefehl **G71** | **METRIC**- selbsthaltend) oder **INCH** (Wegbefehl **G70** | **INCH** - selbsthaltend).

Die Maßeinheit einer rotatorischen Achse ist immer **grad**.

2.3.7 Zusatzbefehle: M-Wort

Die Zahl, die dem Adressbuchstaben **M** folgt, ist eine natürliche Zahl und wird laut DIN 66025 Teil 1 auch als Schlüsselzahl bezeichnet. Die in ProNC realisierten Zusatz-Befehle (ISO-Syntax: M-Befehle, PAL-Syntax: mnemonische Zusatzbefehle) werden ausführlich im Abschnitt 3.1.2 Zusatzbefehle behandelt.

[siehe auch:](#)

Abschnitt 3.1.2 Zusatzbefehle

2.4 Sonderzeichen

In Übereinstimmung mit DIN 66025 bzw. in Ergänzung zu ebenda werden in der folgenden Tabelle die in ProNC erlaubten Sonderzeichen zusammengefasst:

<u>Sonderzeichen</u>	<u>Bedeutung</u>
%	% natürliche Zahl : Beginn des Hauptprogramms
ISO	%L natürliche Zahl: Beginn eines Unterprogramms
PAL	%SUBR natürliche Zahl: Beginn eines Unterprogramms
ISO: (PAL: {	Kommentaranfang , wenn Kommentar sich über mehrere Zeilen erstrecken soll oder als Trennzeichen im NC-Satz benutzt wird
ISO:) PAL: }	Kommentarende , wenn Kommentar sich über mehrere Zeilen erstrecken soll oder als Trennzeichen im NC-Satz benutzt wird
;	Kommentaranfang (einzeiliger Kommentar)
CR (Zeilenende)	Kommentarende (einzeiliger Kommentar)
ISO: [PAL: (Argumentanfang bei Funktionen oder Klammerung von Ausdrücken
ISO:] PAL:)	Argumentende bei Funktionen oder Klammerung von Ausdrücken
+	Vorzeichen bei Dezimalzahlen oder arithmetischer Operator: Addition
-	Vorzeichen bei Dezimalzahlen oder arithmetischer Operator: Subtraktion
*	arithmetischer Operator: Multiplikation
/	arithmetischer Operator: Division oder Satzunterdrückungszeichen, wenn ein N-Wort folgt
&	Bool'sche Verknüpfung: UND
	Bool'sche Verknüpfung: ODER
^	Bool'sche Verknüpfung: ANTIVALENZ bzw. EXCLUSIV ODER: $a \wedge b = (\text{not } a \ \& \ b) \ \ (a \ \& \ \text{not } b)$
<	Vergleichs-Operatoren: kleiner als
>	Vergleichs-Operatoren: größer als
!=	Vergleichs-Operatoren: ungleich
==	Vergleichs-Operatoren: gleich

:	Zeichen zur Auswahl einer Koordinaten-Komponente einer Q-Variablen oder eines symbolischen Frames
/	Satzunterdrückungszeichen
=	Zuweisung von Werten an Koordinaten-Adressbuchstaben bei der indizierten Achs-Adressierung

Tabelle 2.4: Sonderzeichen und deren Bedeutung in ProNC

2.5 Unterprogramme

Die Unterprogrammtechnik in ProNC wird nach den Vorgaben in der DIN 66025 realisiert.

[siehe auch:](#)

Abschnitt 3.1.7 Unterprogrammtechnik

<u>Unterprogramm-...</u>	<u>ISO-Syntax</u>	<u>PAL-Syntax</u>
-Anfang (Deklaration)	%L100	%SUBR100
-Ende (Deklaration)	M17	RETURN
-Aufruf (Aktivierung)	L100	SUBR100

Tabelle 2.5: Unterprogrammdeklaration und -aktivierung

3 ProNC Sprachbeschreibung

3 ProNC Sprachbeschreibung

In den Anwenderprogrammen (ISO-Quellprogramme oder PAL-Quellprogramme, im folgenden kurz *Quellprogramme*), die in ProNC abgearbeitet werden können, ist kein expliziter Deklarationsteil für Konstante oder Variable notwendig. Es besteht lediglich die Forderung, dass in jedem *Quellprogramm* die darin benötigten Unterprogramme vor dem Hauptprogramm(teil) deklariert sein müssen.

Programmtext: Der Programmtext besteht aus **Programmzeilen**.

Um einen eindeutigen Bezug zur Terminologie der Informatik (**Elektronische Datenverarbeitung EDV**) herzustellen, wird in dieser Dokumentation unterschieden zwischen **Programmzeilen**,

- die typisch sind für die Programmierung numerisch gesteuerter Anlagen (Werkzeugmaschinen, Handlingsysteme):

Diese Programmzeilen sind **NC-Sätze**, deren Struktur z.B. in der DIN 66025 / ISO 6983 standardisiert ist.

- die typisch sind für Programmiersprachen der EDV:
Diese Programmzeilen werden als **Anweisungen** bezeichnet.

Programmzeilen: Programmzeilen können **NC-Sätze** oder **Anweisungen** sein.

Demnach besteht ein jedes Quellprogramm aus einer Folge von NC-Sätzen und / oder Anweisungen.

NC-Sätze: NC-Sätze entsprechen in ihrer Syntax den Vorschriften der DIN66025.

Anweisungen: Anweisungen können sein:

- eine leere Programmzeile, das ist eine **leere Anweisung**
- eine Kommentarzeile, das ist ebenfalls eine **leere Anweisung**
- jede Programmzeile, die keinen NC-Satz darstellt, ist eine **Anweisung**

Die Satzstruktur wurde im Abschnitt 2.2 Aufbau eines NC-Satzes dieser Dokumentation definiert. Im Abschnitt 3.1 Befehle nach DIN 66025 sind alle verfügbaren **NC-Sätze** mit den relevanten Befehlen zusammengefasst.

Alle in ProNC nutzbaren **Anweisungen** werden im Abschnitt 3.2 Anweisungen beschrieben.

Zum besseren Verständnis der folgenden Abschnitte 3.1 und Abschnitt 3.2 werden die nachstehenden Erklärungen gegeben:

Programmtext: Für alle Quellprogramme in ProNC gilt, dass Programmtext in **beliebiger Schreibweise (Groß- oder Kleinbuchstaben)** eingegeben (editiert) werden kann.

Demzufolge wird nicht zwischen den Schlüsselworten

- *EndFor*,



- *ENDFOR* oder
 - *endfor*
- unterschieden.

Der Compiler kann einen optionalen Pre-Prozessorlauf ausführen, bei dem alle Kleinbuchstaben (außerhalb jeglicher Kommentare) in Großbuchstaben konvertiert werden. Dies hat zur Folge, dass auch Frame-Namen wie *ParkPosition*, *PARKPOSITION* und *parkposition* nicht unterschieden werden.

Innerhalb von Kommentaren dürfen beliebige Zeichen verwendet werden. Kommentare beginnen entweder mit der runden oder geschweiften öffnenden Klammer (**bzw. {** und enden mit der schließenden runden oder geschweiften Klammer) **bzw. }** oder beginnen mit einem Semikolon ; und enden mit einem Zeilenendezeichen **CR** (**C**arriage **R**eturn). Die Sonderzeichen und deren Bedeutung werden im Abschnitt 2.4 Sonderzeichen definiert.

NC-Sätze:

Alle NC-Sätze können mit einer Satznummer (N-Wort) beginnen. Dies gilt ebenfalls für Variablenzuweisungen / Parameterrechnung. Zur Unterscheidung von Anweisungen zur Steuerung des Programmablaufes (z.B. For-Schleife) von NC-Sätzen werden bei allen Schleifen und Verzweigungen **keine** Satznummern (N-Worte) vorangestellt.

Variable:



In ProNC kann der Anwender durch die Möglichkeit der Nutzung von Variablen eine sehr leistungsfähige und flexible Parameterrechnung (Abschnitt 3.2.3) durchführen. Für einfache, implizite Variable (**P**-, **Q**- oder **R**-Variable) werden keine komplizierten Namen / Bezeichner bzw. Deklarationen benötigt, wie dies bei höheren Programmiersprachen der EDV üblich ist. In ProNC beginnt eine Variable mit dem Großbuchstaben (Adressbuchstaben) **P**, **R** oder **Q**, gefolgt von einer natürlichen Zahl n:

- P-Variable: $0 \leq n < 100$ -> **Prozessvariable**
- Q-Variable: $0 \leq n < 500$ -> **Framevariable**
- R-Variable: $0 \leq n < 1000$ -> **Realvariable**

gültige Variable:

P0, P11, P99 sind gültige P-Variable
 R1, R222, R999 sind gültige R-Variable
 Q2, Q166, Q499 sind gültige Q-Variable



Bezeichner zur Angabe von Frame-Namen:

In einem ProNC-Anwenderprogramm werden Bezeichner benötigt, um die Elemente der Geometriedatei (Framedatei) zu benennen. Die Elemente der Geometriedatei heißen **FRAME**. Demzufolge heißt ein Bezeichner zur Benennung eines Frames auch **Frame-Name**.

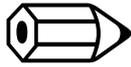


Frame-Name:

Ein Frame-Name besteht aus **minimal 4** Zeichen und **maximal 20** Zeichen. Es wird gefordert, dass die ersten vier Zeichen eines Frame-Namens Großbuchstaben sind. Ab dem vierten Zeichen dürfen Ziffern, Großbuchstaben sowie der Unterstrich „_“ in beliebiger Reihenfolge benutzt werden.

gültige / ungültige Frame-Namen:**gültige Frame-Namen:**

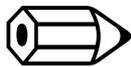
MAXI, ABCD, ABSO, MAXIMUM, MINIMUM, ELVIRA123

**ungültige Frame-Namen:**

111, AB1, 12_NORM, _1, N_1, A, AN3_ANTON

natürliche Zahlen:

Natürliche Zahlen werden zur Festlegung der Schlüsselzahl, z.B. bei allen G-Befehlen und M-Befehlen, angegeben. Sie dürfen **kein** +/- Vorzeichen besitzen.

Beispiel:**gültige natürliche Zahlen:**

100, 200, 300, 1

ungültige natürliche Zahlen:

+100, +200, +300, +1_

Hexadezimalzahlen:

Natürliche Zahlen können ebenfalls hexadezimal angegeben werden. In diesem Fall ist der Zeichenfolge die Kennung **0x** voranzustellen. Zulässig als Kennung ist ebenfalls ein vorangestelltes Dollarzeichen **\$** oder ein nachgestelltes **H**.

Dieser Kennung müssen mindestens ein und maximal acht Zeichen aus folgendem Zeichenvorrat folgen:

- die Ziffern 1,2,3,4,5,6,7,8,9,0
- die Kleinbuchstaben a,b,c,d,e,f
- die Großbuchstaben A,B,C,D,E,F

Kompakt kann diese Regel durch den *regulären Ausdruck*

0x{[0-9a-fA-F]}{1,8}

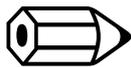
oder

\${[0-9a-fA-F]}{1,8}

oder

{[0-9a-fA-F]}{1,8} **H**

beschrieben werden.

Beispiel:

- **gültige hexadezimale Zahlen:**

0x1234, 0xaa, 0xAA, \$Ff, 12345678H, abcdH

- **ungültige hexadezimale Zahlen:**

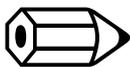
0a1234, xaa, 0yAA, 0x_Ff, 0x123456789, 0xabxycd

[siehe auch:](#)

Abschnitt 3.2.2.2 Funktionen

Binärzahlen: Eine natürliche Zahl kann als Binärzahl dargestellt werden. Die Kennzeichnung der Ziffernfolge **bbbbbbbb** als Binärzahl erfolgt mit einem **B**.
bbbbbbbbB b =[0,1]

Beispiel:



- 10101010B binäre Schreibweise für die natürliche Zahl 170
- 00000011B binäre Schreibweise für die natürliche Zahl 3

Dezimalzahlen: Dezimalzahlen zur Angabe von Koordinaten, Geschwindigkeiten, Konstanten (Direktwerte zur Zuweisung an R-Variable) oder Argumenten von Funktionen können auf drei verschiedene Arten angegeben werden:

- als Dezimalzahl mit ganzem und gebrochenem Teil, z.B. **3.142** oder **0.142**
- als Dezimalzahl ohne ganzem und mit gebrochenem Teil, z.B. **.142**
- als Dezimalzahl mit ganzem und ohne gebrochenem Teil, z.B. **3.**

Schlüsselworte (Tokens): Wie in jeder Programmiersprache sind auch in ProNC Schlüsselworte vorhanden, die bestimmte syntaktische Konstruktionen in ihrer Struktur definieren. Diese Schlüsselworte (im Sprachgebrauch der Informatik häufig auch als *Tokens* bezeichnet) sind nachfolgend zusammengefasst:



- für Anweisungen zur Steuerung des Programmablaufes:
 - **FOR, ENDFOR**
 - **WHILE, ENDWHILE**
 - **DO, ENDDO**
 - **REPEAT, UNTIL**
 - **IF, ELSE, ENDIF**
 - **SWITCH, CASE, ENDCASE, DEFAULT, ENDSWITCH**
- für trigonometrische Funktionen:
 - **SIN, COS, TAN**
 - **ASIN, ACOS, ATAN**
- für reelle Funktionen
 - **FABS**
 - **SQR, SQRT**
 - **FLOOR**
 - **EXP**
 - **LOG, LN**
 - **POW**
- für Wartezeiten
 - **TIME**
- für die Kreiszahl Pi
 - **Pi**

arithmetische und Bool'sche Ausdrücke: Bei der Parameterrechnung werden sowohl arithmetische als auch Bool'sche Ausdrücke verwendet. Allgemein ist ein Ausdruck eine Zahl, eine Variable oder eine Verknüpfung von Zahlen und / oder Variablen. Je nachdem, ob die Verknüpfung eine arithmetische oder eine Bool'sche ist, wird ein derartiger Ausdruck zum arithmetischen oder zum Bool'schen Ausdruck.

Anweisungen zur Steuerung des Programmablaufes: Bei Anweisungen zur Steuerung des Programmablaufes (FOR-Schleife, WHILE-Schleife, DO/REPEAT-Schleife) bzw. bei einer Programmverzweigung (IF-Konstruktion, SWITCH-Konstruktion) werden **Bedingungen** getestet. Bedingungen sind Vergleiche zwischen arithmetischen Ausdrücken oder Bool'sche Ausdrücke. Eine Bedingung hat immer einen sogenannten Wahrheitswert:

Ist die Bedingung erfüllt, ist der Wahrheitswert gleich 1 (**TRUE**).

Ist die Bedingung nicht erfüllt, ist der Wahrheitswert gleich 0 (**FALSE**).

Bedingung: In der Syntax-Notation wird die Bedingung klein geschrieben. Damit wird zum Ausdruck gebracht, dass an dieser Stelle einer grammatischen Konstruktion eine syntaktisch fehlerfreie Notation stehen muss, die eine Bedingung darstellt.

Syntax-Notation für eine Bedingung: In der Syntax-Notation

IF bedingung

...
ELSE
...
ENDIF

steht **bedingung** für eine zulässige Notation einer Bedingung. Diese kann z.B. heißen:



R1 > R2

Dann wäre eine syntaktisch richtiger Programmtext z.B.:

```
IF R1 > R2
  N10 G1 G91 X100
ELSE
  N20 G1 G91 X-100
ENDIF
```

Anweisung: Bei allen Anweisungen zur Steuerung des Programmablaufes taucht in der Syntax-Notation das Wort **anweisungen** (klein geschrieben) auf. Dieses Wort **anweisungen** steht als Abkürzung für:



- **leere Anweisung**
(leere Programmzeile oder Kommentarzeile)
- oder
- **NC-Satz**
- oder
- **Folge von NC-Sätzen**
- oder
- **Anweisung**

oder

- **Folge von Anweisungen**

[siehe auch:](#)

Abschnitt 3.2.3 Anweisungen zur Steuerung des Programmablaufes

Schachtelungstiefe:



Es wird offensichtlich, dass eine Anweisung zur Steuerung des Programmablaufes (z.B. FOR-Schleife) wieder Anweisungen enthalten kann. Da diese Anweisung wieder eine FOR-Schleife sein kann, ist in ProNC eine Schachtelung von Anweisungen möglich. Um den Verwaltungsaufwand dieser Schachtelungen zu begrenzen, wird die sogenannte **Schachtelungstiefe** auf **fünf** begrenzt.

In den folgenden Abschnitten wird eine einheitliche Gliederung für die Beschreibung aller NC-Sätze (Worte / Befehle innerhalb von NC-Sätzen) bzw. Beschreibung aller Anweisungen benutzt:

NC-Satz:

Befehl nach ISO-Syntax	Kurzerklärung für den NC-Satz	Befehl nach PAL-Syntax
-------------------------------	--------------------------------------	-------------------------------

Anweisung:

Name der Anweisung	Kurzerklärung für die Anweisung
---------------------------	--

- Syntax:** Die Syntax definiert, wie die Konstruktion (WORT / BEFEHL oder ANWEISUNG) im Anwender-Programmtext zu erscheinen hat. Es wird angegeben, welche Parameter, z.B. Koordinaten, Variable oder Bezeichner innerhalb der Konstruktion erlaubt sind.

Hinweis zur Notation im Syntax-Feld:

<u>Notation</u>	<u>Bedeutung</u>
[konstruktion]?	die in eckigen Klammern angegebene Konstruktion ist optional, d.h. sie kann maximal einmal programmiert oder weggelassen werden
[konstruktion]*	keine, eine oder mehrere Wiederholungen der angegebenen Konstruktion
[konstruktion]+	eine oder mehrere Wiederholungen der angegebenen Konstruktion
[konstruktion]{m,n}	minimal m und maximal n Wiederholungen der Konstruktion

- Erklärung: Der Zweck, die Aufgabe, die Besonderheiten und / oder die Anwendung der Konstruktion werden textuell erläutert.
- Beispiel: Der Zweck, die Aufgabe, die Besonderheiten und / oder die Anwendung der Konstruktion werden mit Beispielen erläutert.
- Verweis: Ein Hinweis auf verwandte Konstruktionen wird gegeben.

3.1 Befehle nach DIN 66025 im NC-Satz

3.1.1 Wegbefehle

Eilganggeschwindigkeit Die Eilganggeschwindigkeit wird in der Initialisierungsdatei des Bewegungsmoduls (*isel-Motion Control MCTL*) definiert oder mit dem Befehl FASTVEL im Anwenderprogramm eingestellt (modale Wirkung).

Mit Eilganggeschwindigkeit werden vor allem Zustellbewegungen programmiert, bei denen z.B. vor einer Bearbeitung der Werkstücknullpunkt oder nach der Bearbeitung eine Parkposition angefahren werden.

Segmentgeschwindigkeit Die Segmentgeschwindigkeit wird in der Initialisierungsdatei des Bewegungsmoduls (*isel-Motion Control MCTL*) definiert oder im Quellprogramm mit Hilfe des F-Befehles eingestellt.



Mit Segmentgeschwindigkeit werden vor allem technologische Bewegungen programmiert, so z.B. das Fräsen einer Kante, das Schweißen einer Naht oder das Bohren eines Loches. Allen diesen Bewegungen ist gemeinsam, dass ein Bewegungssegment (Gerade oder Kreis) oder eine Bahn abgefahren wird.

Interpolationsebene: Die Angabe der Interpolationsebene ist nur bei kartesischen Systemen sinnvoll, weil nur dort ein Kreisbefehl vom Bewegungsmodul ausgeführt wird.

Die Interpolationsebene legt fest, in welcher Ebenen der nächste Kreis gefahren wird: X-Y-Ebene oder X-Z-Ebene oder Y-Z-Ebene

Die Festlegung der Interpolationsebene hat für die Linearinterpolation (Geradenbefehle **G0 | FASTABS** oder **G1 | MOVEABS** oder **G10 | FASTFRAME** oder **G11 | MOVEFRAME**) bei kartesischen Kinematiken keinen Einfluss, weil diese Interpolation immer eine 3D-Interpolation ist.

Nullpunktverschiebung



Eine Nullpunktverschiebung dient bei der technologischen Bearbeitung (Fräsen, Bohren, Kleben, Schweißen u.a.) vor allem dazu, den Nullpunkt des Werkstückkoordinatensystems gegenüber dem Maschinenkoordinatensystem festzulegen.

Bei Handlingsystemen wird die Nullpunktverschiebung benutzt, um im globalen Koordinatensystem des Handlingsystems ein sogenanntes lokales Koordinatensystem, z.B. das Bezugssystem eines Bilderkennungssystems, zu eröffnen.

3.1.1.1 Übersicht Wegbefehle in ProNC

<u>ISO-Befehl</u>	<u>Bedeutung</u>	<u>PAL-Befehl</u>
G0	Bewegung mit Eilgang geschwindigkeit	FASTABS FASTREL
G1	Geradeninterpolation bei kartesischen Kinematiken S-PTP-Bewegung bei nichtkartesischen Kinematiken	MOVEABS MOVEREL
G2	Kreisinterpolation clw bei kartesischen Kinematiken	CWABS CWREL
G3	Kreisinterpolation cclw bei kartesischen Kinematiken	CCWABS CCWREL
G4	Verweilzeit	TIME DELAY
G10 G11	Bewegung mit Eilgang geschwindigkeit in Verbindung mit einer Frame-Variablen Q0 ... Q499 Bewegung mit Segment geschwindigkeit in Verbindung mit einer Frame-Variablen Q0 ... Q499	FASTFRAME MOVEFRAME
G12	Helix im Uhrzeigersinn	CWHLXABS CWHLXREL
G13	Helix entgegen dem Uhrzeigersinn	CCWHLXABS CCWHLXREL
G17 G18 G19	Definition der Interpolationsebene (X-Y-Ebene) Definition der Interpolationsebene (X-Z-Ebene) Definition der Interpolationsebene (Y-Z-Ebene)	PLANE XY PLANE XZ PLANE YZ
G53 G54 G55 G56	Nullpunktverschiebung deaktivieren Nullpunktverschiebung 1 aktivieren Nullpunktverschiebung 2 aktivieren Werkstücknullpunkt auf der aktuellen Position setzen	WPCLEAR WPREG1 WPREG2 WPZERO
G60 G64	expliziten Bahnbetrieb ausschalten (Bahn-Ende) expliziten Bahnbetrieb einschalten (Bahn-Anfang)	PATHEND PATH
G70 G71	Definition der Maßeinheit für translatorische Achsen: inch Definition der Maßeinheit für translatorische Achsen: mm	INCH METRIC
G74	Referenzpunktfahrt	REF
G75	Teach-In : die Dialogbox „aktuelle Geometriedatei: ...“ kann während des Automatikbetriebes aktiviert werden	TEACH
G80	Bohrzyklusparameter definieren	DrillDef
G81 G82 G83 G84	Einfaches Bohren Bohren mit Verweilzeit Bohren in Betriebsart Ausräumen Bohren in Betriebsart Spanbrechen	DrillN DrillT DrillD DrillB

G90	Koordinatenangaben sind Absolutangaben (Absolutmaß)	ABS
G91	Koordinatenangaben sind Relativangaben (Kettenmaß)	REL
G92	Speicher setzen (Werkstücknullpunktregister 1)	WPREG1WRITE
G93	Speicher setzen (Werkstücknullpunktregister 2)	WPREG2WRITE
G98	Parametereingabe für Technologie-Variable (R-Variable)	PARAMETER
G99	Textausgabe in die Statuszeile	TYPE

Tabelle 3.1.1: Wegbefehle in ProNC (Übersicht)

3.1.1.2 Eilganggeschwindigkeit

G0-Befehl	Bewegung mit Eilganggeschwindigkeit	FASTABS-Befehl FASTREL-Befehl
------------------	---	--

□ Syntax:	[Satznummer]? [weiterer Befehl: G17, G18, G19, G70, G71]* G0 [Ziel-Koordinaten]{1,6} [F-Befehl]? [S-Befehl]? [M-Befehl]*	[Satznummer]? [weiterer Befehl: PLANE XY, PLANE XZ, PLANE YZ, INCH, METRIC]* FASTABS oder FASTREL [Ziel-Koordinaten]{1,6} [F-Befehl]? [S-Befehl]? [Zusatz-Befehl]*
-----------	---	--

- Erklärung:
- kartesische Kinematik:**
Positionierbewegung mit Eilganggeschwindigkeit:
- die Eilganggeschwindigkeit wird in der Initialisierungsdatei des Bewegungsmoduls definiert oder durch den Befehl **FASTVEL**
 - es muss mindestens eine Koordinatenangabe im NC-Satz vorhanden sein
 - es dürfen maximal sechs Koordinatenangaben im NC-Satz vorhanden sein
 - ist Absolutmaß (**G90 | ABS**) eingestellt, beziehen sich die Zielkoordinaten auf den aktuellen Nullpunkt des Werkstückkoordinatensystems
 - ist Kettenmaß (**G91 | REL**) eingestellt, beziehen sich die Zielkoordinaten auf den aktuellen Startpunkt
 - die Maßeinheit der Zielposition (X, Y, Z) ist Millimeter [mm], für Drehachsen (A, B, C) grad [°]

- Beispiel:
- kartesische Kinematiken:**
 ; Absolut-Bewegung zum Ziel-Punkt mit den Koordinaten
 ; (100mm, 200mm, 300mm) mit Eilganggeschwindigkeit:
ISO: N200 **G00** G90 X100.0 Y200.0 Z300.0
PAL: N200 **FASTABS** X100.0 Y200.0 Z300.0



- kartesische Kinematiken:**
 ; Relativ-Bewegung der X-Achse **um** 10mm, der Y-Achse **um**
 ; 20 mm und der Z-Achse um 30mm, vom aktuellen Startpunkt
 ; aus betrachtet, mit Eilganggeschwindigkeit:
ISO: N200 **G00** G91 X10.0 Y20.0 Z30.0
PAL: N200 **FASTREL** X10.0 Y20.0 Z30.0

- nichtkartesische Kinematik:**
 ; Absolut-Bewegung zum Ziel-Punkt mit den Werten:
 ; C-Achse: 100 grad Z-Achse: 180 mm
 ; B-Achse: 45.0 grad A-Achse: -45.0 grad
 ; mit Eilganggeschwindigkeit:
ISO: N100 **G00** G90 C100.0 Z180.0 B45.0 A-45.0
PAL: N100 **FASTABS** C100.0 Z180.0 B45.0 A-45.0

- Verweis:
- | | |
|--|---|
| G1, G10,
G11, G70, G71,
G90, G91 | MOVEABS, FASTFRAME,
MOVEFRAME, INCH, METRIC,
ABS, REL |
|--|---|

3.1.1.3 Geradeninterpolation

G1-Befehl	Geradeninterpolation bei kartesischen Kinematiken S-PTP-Bewegung bei nichtkartesischen Kinematiken	MOVEABS-Befehl MOVEREL-Befehl
------------------	---	--

□ Syntax:	[Satznummer]? [weiterer Befehl: G17, G18, G19, G70, G71]* G1 [Ziel-Koordinaten]{1,6} [F-Befehl]? [S-Befehl]? [M-Befehl]*	[Satznummer]? [weiterer Befehl: PLANE XY, PLANE XZ, PLANE YZ, INCH, METRIC]* MOVEABS oder MOVEREL [Ziel-Koordinaten]{1,6} [F-Befehl]? [S-Befehl]? [Zusatz-Befehl]*
-----------	---	---

□ Erklärung: **kartesische Kinematik: Linearinterpolation mit Segmentgeschwindigkeit**
nichtkartesische Kinematik: Positionierbewegung mit Segmentgeschwindigkeit

- die Segmentgeschwindigkeit kann mit Hilfe eines F-Befehles im aktuellen NC-Satz definiert werden oder es gilt die in einem vorangegangenen NC-Satz definierte Segmentgeschwindigkeit
- es muss mindestens eine Koordinatenangabe im NC-Satz vorhanden sein
- es dürfen maximal sechs Koordinatenangaben im NC-Satz vorhanden sein
- ist Absolutmaß (**G90 | ABS**) eingestellt, beziehen sich die Zielkoordinaten auf den aktuellen Nullpunkt des Werkstückkoordinatensystems
- ist Kettenmaß (**G91 | REL**) eingestellt, beziehen sich die Zielkoordinaten auf den aktuellen Startpunkt
- die Maßeinheit der Zielposition (X, Y, Z) ist Millimeter [mm], für Drehachsen (A, B, C) grad [°]

□ Beispiel:



kartesische Kinematiken (XYZ):
; Gerade im Raum zum Absolut-Ziel-Punkt mit den
; Koordinaten (100mm, 200mm, 300mm) mit
; Segmentgeschwindigkeit:

ISO: N100 **G1** G90 X100.0 Y200.0 Z300.0
PAL: N100 **MOVEABS** X100.0 Y200.0 Z300.0

kartesische Kinematiken (XYZ):
; Gerade im Raum zum Ziel-Punkt mit den Koordinaten
; X-IST + 10mm, Y-IST + 20 mm, Z-IST - 30mm
; mit Segmentgeschwindigkeit:

ISO: N200 **G1** G91 X10.0 Y20.0 Z-30.0
PAL: N200 **MOVEREL** X10.0 Y20.0 Z-30.0

nichtkartesische Kinematik:
; Absolut-Bewegung zum Ziel-Punkt mit den Werten:
; C-Achse: 100 grd Z-Achse: 180 mm
; B-Achse: 45.0 grd A-Achse: -45.0 grd
; mit Segmentgeschwindigkeit:

ISO: N100 **G01** G90 C100.0 Z180.0 B45.0 A-45.0
PAL: N100 **MOVEABS** C100.0 Z180.0 B45.0 A-45.0

□ Verweis:

G0, G10,
G11, G70, G71,
G90, G91

FASTABS, FASTFRAME,
MOVEFRAME, INCH, METRIC,
ABS, REL

3.1.1.4 Kreisinterpolation im Uhrzeigersinn

G2-Befehl	Kreisinterpolation cw (clockwise) bei kartesischen Kinematiken	CWABS-Befehl CWREL-Befehl
------------------	--	--

□ Syntax:	[Satznummer]? [weiterer Befehl]: G17, G18, G19, G70, G71]* G2 [Ziel-Koordinaten]{1,3} [Mittelpunkt-Koordinaten]{1,3} [F-Befehl]? [S-Befehl]? [M-Befehl]*	[Satznummer]? [weiterer Befehl]: PLANE XY, PLANE XZ, PLANE YZ, INCH, METRIC]* CWABS oder CWREL [Ziel-Koordinaten]{1,3} [Mittelpunkt-Koordinaten]{1,3} [F-Befehl]? [S-Befehl]? [Zusatz-Befehl]*
-----------	--	---

□ Erklärung: **kartesische Kinematik:
Kreis / Kreisbogen in der aktiven Interpolationsebene
im Uhrzeigersinn mit Angabe der Mittelpunktskoordinaten**

- dieser Befehl ist nur für kartesische Anlagen zu verwenden
- es ist mindestens eine Ziel-Koordinate anzugeben und die korrespondierende Mittelpunkts-Koordinate:
X -> I, Y -> J, Z -> K
- Ziel-Koordinatenangaben können absolut (G90 | **ABS**) oder relativ (G91 | **REL**) erfolgen
- Mittelpunkts-Koordinatenangaben erfolgen immer relativ bezüglich des Start-Punktes
- die Maßeinheit der Zielposition ist Millimeter [mm]
- der Drehsinn ist so definiert, dass auf die Interpolationsebene derart hinabgesehen wird, dass die Blickrichtung immer von positiv nach negativ verläuft:

Hinweis:

Ist mit dem Befehl G17 | **PLANE XY** die X-Y-Ebene als Interpolationsebene ausgewählt, ist aus positiver Z-Richtung auf eine „Phantom-Uhr“ in dieser Ebene zu blicken, deren Drehrichtung stimmt mit dem Drehsinn des Kreises überein.

□ Beispiel: ; **Halb-Kreis** im Uhrzeigersinn in der X-Y-Ebene:
; **Anfangspunkt:**
 $(X_A, Y_A) = (0, 0)$
; **Endpunkt:**
 $(X_E, Y_E) = (100, 0)$

; Segmentgeschwindigkeit: **50** mm/sec:

ISO: N10 G17 ; Interpolationsebene einstellen
N20 G0 G90 X0 Y0 ; Anfangspunkt anfahren
N30 **G2** X100 I50 F50 ; Kreis fahren

PAL: N10 **PLANE XY** ; Interpolationsebene einstellen
N20 **FASTABS** X0 Y0 ; Anfangspunkt anfahren
N30 **CWABS** X100 I50 F50 ; Kreis fahren

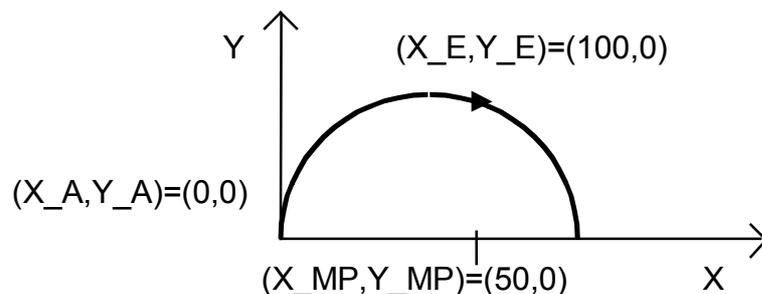


Hinweis:

die **Mittelpunkts**koordinaten (X_MP, Y_MP) ergeben sich immer durch Addition der I- bzw. J-Werte zu den Anfangswerten des Kreises (X_A, Y_A):

$$\begin{aligned} X_MP &:= X_A + I \\ Y_MP &:= Y_A + J \end{aligned}$$

Daraus folgt, dass die I-, J- und K-Koordinaten immer Relativangaben sind.



□ Beispiel:

; **Kreisbogen** im Uhrzeigersinn in der X-Y-Ebene:

; Anfangspunkt:

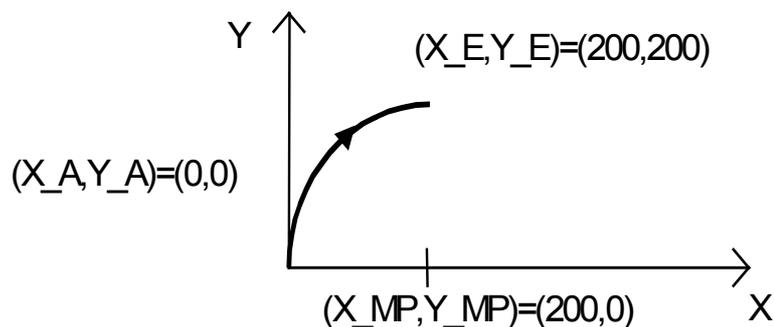
$$(X_A, Y_A) = (0, 0)$$

; Endpunkt:

$$(X_E, Y_E) = (200, 200)$$

; Segmentgeschwindigkeit von **75 mm/sec**:

```
ISO: N10 G17 G90
      N20 G0 X0 Y0
      N30 G2 X200 Y200 I200 J0 F75
PAL: N10 PLANE XY ABS
      N20 FASTABS X0 Y0
      N30 CWABS X200 Y200 I200 J0 F75
```



□ Verweis:

G3, G17, G18,
G19, G90, G91

CCWABS, PLANE XY, PLANE XZ,
PLANE YZ, ABS, REL

3.1.1.5 Kreisinterpolation entgegen dem Uhrzeigersinn

G3-Befehl	Kreisinterpolation ccw (counter clockwise) bei kartesischen Kinematiken	CCWABS-Befehl CCWREL-Befehl
□ Syntax:	[Satznummer]? [weiterer Befehl: G17, G18, G19, G70, G71]* G3 [Ziel-Koordinaten]{1,3} [Mittelpunkt-Koordinaten]{1,3} [F-Befehl]? [S-Befehl]? [M-Befehl]*	[Satznummer]? [weiterer Befehl: PLANE XY, PLANE XZ, PLANE YZ, INCH, METRIC]* CCWABS oder CCWREL [Ziel-Koordinaten]{1,3} [Mittelpunkt-Koordinaten]{1,3} [F-Befehl]? [S-Befehl]? [Zusatz-Befehl]*

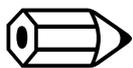
□ Erklärung: **kartesische Koordinaten:**
Kreis / Kreisbogen in der aktiven Interpolationsebene entgegen dem Uhrzeigersinn mit Angabe der Mittelpunktskoordinaten

- dieser Befehl ist nur für kartesische Anlagen zu verwenden
- es ist mindestens eine Ziel-Koordinate anzugeben und die korrespondierende Mittelpunkts-Koordinate:
X -> I, Y -> J, Z -> K
- Ziel-Koordinatenangaben können absolut (G90 | **ABS**) oder relativ (G91 | **REL**) erfolgen
- Mittelpunkts-Koordinatenangaben erfolgen immer relativ bezüglich des Start-Punktes
- die Maßeinheit der Zielposition ist Millimeter [mm]
- der Drehsinn ist so definiert, dass auf die Interpolationsebene derart hinabgesehen wird, dass die Blickrichtung immer von positiv nach negativ verläuft:

Hinweis:

Ist mit dem Befehl G17 | **PLANE XY** die X-Y-Ebene als Interpolationsebene ausgewählt, ist aus positiver Z-Richtung auf eine „Phantom-Uhr“ in dieser Ebene zu blicken, deren Drehrichtung stimmt mit dem Drehsinn des Kreises überein.

□ Beispiel: ; **Viertel-Kreis** entgegen dem Uhrzeigersinn in der XY-Ebene:
; Anfangspunkt:



(X_A, Y_A)=(600,0)

; Endpunkt:

(X_E, Y_E)=(300,300)

; Segmentgeschwindigkeit von **66** mm/sec:

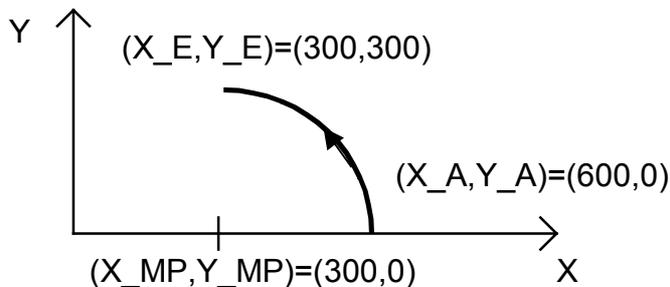
ISO: N10 G17 G90
N20 G0 X600 Y0 ; Anfangspunkt anfahren
N30 **G3** X300 Y300 I-300 F66 ; Kreis fahren

PAL: N10 **PLANE XY ABS**
N20 **FASTABS** X600 Y0 ; Anfangspunkt anfahren
N30 **CCWABS** X300 Y300 I-300 F66 ; Kreis fahren

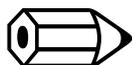
Hinweis:

Die Absolutkoordinaten des Kreismittelpunktes in der folgenden Zeichnung ergeben sich, indem der im Satz N30 spezifizierte I-Koordinatenwert von -300 zum Anfangswert der X-Koordinate addiert wird: $600 - 300 = 300$.

Da die Mittelpunktskoordinate $Y_{MP} = 0$ sich gegenüber dem Anfangswert $Y_A = 0$ nicht ändert, kann die Angabe des J-Koordinatenwertes im NC-Satz entfallen.



□ Beispiel:



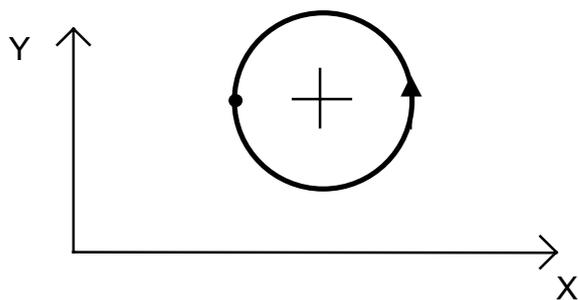
; **Vollkreis** entgegen dem Uhrzeigersinn in der X-Y-Ebene:

; Anfangspunkt: $(X_A, Y_A) = (120, 180)$

; Endpunkt: $(X_E, Y_E) = (120, 180)$

; Segmentgeschwindigkeit von **11** mm/sec:

```
ISO: N10 G17 G90
      N20 G0 X120 Y180
      N30 G3 X120 Y180 I50 J0 F11
PAL: N10 PLANE XY ABS
      N20 FASTABS X120 Y180
      N30 CWABS X120 Y180 I50 J0 F11
```



● $(X_A, Y_A) = (X_E, Y_E) = (120, 180)$

⊕ $(X_{MP}, Y_{MP}) = (170, 180)$

□ Verweis:

G2, G17, G18,
G19, G90, G91

CWABS, PLANE XY, PLANE XZ,
PLANE YZ, ABS, REL

3.1.1.7 Eilanggeschwindigkeit mit Frameangabe

G10-Befehl	Bewegung mit Eilang- geschwindigkeit in Verbindung mit einer Frame-Variablen Q0 ... Q499 oder einer indexierten Q-Variablen oder einem Frame-Name	FASTFRAME-Befehl
-------------------	---	-------------------------

<input type="checkbox"/> Syntax:	[Satznummer]? [weiterer Befehl: G17, G18, G19, G70, G71]* G10 q_variable <i>oder</i> G10 Q r_variable <i>oder</i> G10 frame_name [S-Befehl]? [M-Befehl]*	[Satznummer]? [weiterer Befehl: PLANE XY, PLANE XZ, PLANE YZ, INCH, METRIC]* FASTFRAME q_variable <i>oder</i> FASTFRAME Q r_variable <i>oder</i> FASTFRAME frame_name [S-Befehl]? [Zusatzbedingung]*
----------------------------------	---	---

Erklärung: **Positionierbewegung mit Eilanggeschwindigkeit, bei der keine expliziten Ziel-Koordinaten angegeben werden, sondern eine Frame-Variable (Q-Variable) oder eine indexierte Q-Variable oder ein Frame-Name. Die Zielangabe ist immer absolut.**

- die Eilanggeschwindigkeit ist in der Initialisierungsdatei des Bewegungsmoduls oder durch den Befehl **FASTVEL** definiert
- die Frame-Variablen müssen im Initialisierungsteil des Anwenderprogramms initialisiert werden

ACHTUNG

Nach einem **G10-Befehl** | **FASTFRAME-Befehl** ist als Maßangabe immer **Absolutmaß** aktiv, auch wenn vor dem **G10-Befehl** | **FASTFRAME-Befehl** **Relativmaß (Kettenmaß)** durch einen **G91-Befehl** | **REL-Befehl** eingestellt war.

Beispiel: ; die Q-Variable Q1 wird initialisiert:
N10 Q1 = START



; Positionierbewegung im Eilang zu der Position, die aktuell in der Q-Variablen Q1 abgelegt ist:

ISO: N20 G10 Q1
PAL: N20 **FASTFRAME** Q1

; Indexierung von Q-Variablen:

ISO: N100 G10 QR5 ; Synchron-PTP-Bewegung zum
; Q-Zielpunkt, dessen Index gerade in
; R5 steht

PAL: N100 **FASTFRAME** QR5 ;Synchron-PTP-Bewegung zum
; Q-Zielpunkt, dessen Index gerade in
; R5 steht

; direkte Angabe des Frame-Namens im Befehl:

ISO: N20 G10 PARK_POSITION
PAL: N20 **FASTFRAME** PARK_POSITION

Verweis: **G11** | **MOVEFRAME**
Abschnitt 3.2.1.2: Q-Variable
Abschnitt 3.2.2.4: Zuweisungen

3.1.1.8 Segmentgeschwindigkeit mit Frameangabe

G11-Befehl	Bewegung mit Segment- geschwindigkeit in Verbindung mit einer Frame-Variablen Q0 ... Q499 oder indexierten Q-Variablen oder einem Frame-Name	MOVEFRAME
-------------------	--	------------------

□ Syntax:	[Satznummer]? [weiterer Befehl]: G17, G18, G19, G70, G71]* G11 q_variable oder G11 Q r_variable oder G11 frame_name [F-Befehl]? [S-Befehl]? [M-Befehl]*	[Satznummer]? [weiterer Befehl]: PLANE XY, PLANE XZ, PLANE YZ, INCH, METRIC]* MOVEFRAME q_variable oder MOVEFRAME Q r_variable oder MOVEFRAME frame_name [F-Befehl]? [S-Befehl]? [Zusatzbedingung]*
-----------	---	---

□ Erklärung: **Linearinterpolation mit Segmentgeschwindigkeit, bei der keine expliziten Ziel-Koordinaten angegeben werden, sondern eine Frame-Variable oder indexierte Q-Variable oder ein Frame-Name.**

- die Zielangabe erfolgt immer absolut
- die Segmentgeschwindigkeit kann mit Hilfe eines F-Befehles oder VEL-Befehles im aktuellen NC-Satz definiert werden oder es gilt die in einem vorangegangenen NC-Satz definierte Segmentgeschwindigkeit
- die Frame-Variablen müssen im Initialisierungsteil des Anwenderprogramms initialisiert werden

ACHTUNG

Nach einem G11-Befehl | MOVEFRAME ist als Maßangabe immer Absolutmaß aktiv, auch wenn vor dem G11-Befehl | MOVEFRAME Relativmaß (Kettenmaß) durch einen G91-Befehl | REL-Befehl eingestellt war.

□ Beispiel: ;die Q-Variable Q2 wird initialisiert:
; N10 Q2 = ENDE

; Positionierbewegung mit definierter Segmentgeschwindigkeit zu der
; Position, die aktuell in der Q-Variable Q2 abgelegt ist:

ISO: N20 G11 Q2 F100.1
PAL: N20 **MOVEFRAME** Q2 F100.1

; Indexierung von Q-Variablen:

ISO: N100 G11 QR6 ; Synchron-PTP-Bewegung zum
; Q-Zielpunkt, dessen Index gerade
; in R6 steht

PAL: N100 **MOVEFRAME** QR6 ; Synchron-PTP-Bewegung zum
; Q-Zielpunkt, dessen Index gerade
; in R6 steht

; direkte Angabe des Frame-Namens im Befehl:

ISO: N20 G11 PARK_POSITION
PAL: N20 **MOVEFRAME** PARK_POSITION



□ Verweis:

G10

FASTFRAME

Abschnitt 3.2.1.2: Q-Variable

Abschnitt 3.2.2.4: Zuweisungen

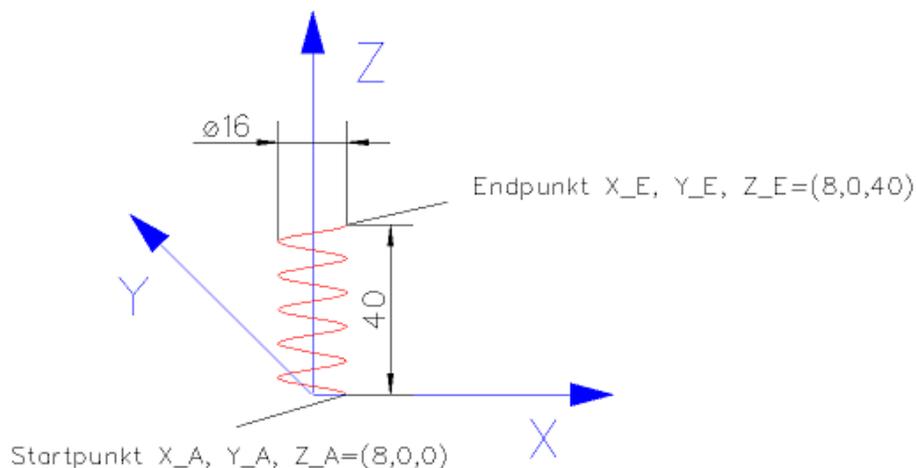
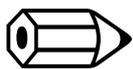
3.1.1.9 Helix im Uhrzeigersinn

G12-Befehl	Helixinterpolation CW (clockwise) bei kartesischen Kinematiken	CWHLXABS-Befehl CWHLXREL-Befehl
------------	--	------------------------------------

□ Syntax:	[Satznummer]? [weiterer Befehl: G70, G71]* G12 Drehwinkel W [Ziel-Koordinaten]{1,3} [Mittelpunkt-Koordinaten]{1,3} [F-Befehl]? [S-Befehl]? [M-Befehl]*	[Satznummer]? [weiterer Befehl: INCH, METRIC]* CWHLXABS oder CWHLXREL Drehwinkel W [Ziel-Koordinaten]{1,3} [Mittelpunkt-Koordinaten]{1,3} [F-Befehl]? [S-Befehl]? [Zusatz-Befehl]*
-----------	--	---

- Erklärung: Spiralbewegung zu einem Endpunkt (Zielkoordinaten), um einen Radiusmittelpunkt (Mittelpunktskoordinaten) mit dem Drehwinkel W im Uhrzeigersinn.
- Ziel-Koordinatenangaben können absolut (G90 | ABS) oder relativ (G91 | REL) erfolgen
 - Mittelpunkts-Koordinatenangaben erfolgen immer relativ bezüglich des Start-Punktes
 - die Angabe des Drehwinkels definiert die Anzahl der Umdrehungen; 360° definiert 1 Umdrehung

- Beispiel: ; Helix fahren mit einer Gesamtwinkelzahl von 1800 grad
; (entspricht 5 Vollkreisen) mit einem Radius = 8 mm



ISO:	N10 G17 ; Interpolationsebene einstellen
N20 G0 G90 X8 Y0 Z0 ; Anfangspunkt anfahren	N30 G12 W1800 X8 Z40 I-8 J0 ; Helix abfahren
PAL:	N10 PLANE XY ; Interpolationsebene einstellen
N20 FASTABS X8 Y0 Z0 ; Anfangspunkt anfahren	N30 CWHLXABS W1800 X8 Z40 I-8 J0 ; Helix abfahren

- Verweis: **G13, G2, G3** | **CWHLXABS, CWABS, CCWABS**

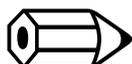
3.1.1.10 Helix entgegen dem Uhrzeigersinn

G13-Befehl	Helixinterpolation CCW (counter clockwise) bei kartesischen Kinematiken	CCWHLXABS-Befehl CCWHLXREL-Befehl
-------------------	--	--

<input type="checkbox"/> Syntax:	[Satznummer]? [weiterer Befehl: G17, G18, G19, G70, G71]* G13 Drehwinkel W [Ziel-Koordinaten]{1,3} [Mittelpunkt-Koordinaten]{1,3} [F-Befehl]? [S-Befehl]? [M-Befehl]*	[Satznummer]? [weiterer Befehl: INCH, METRIC]* CCWHLXABS oder CCWHLXREL Drehwinkel W [Ziel-Koordinaten]{1,3} [Mittelpunkt-Koordinaten]{1,3} [F-Befehl]? [S-Befehl]? [Zusatz-Befehl]*
----------------------------------	--	---

- Erklärung: Spiralbewegung zu einem Endpunkt (Zielkoordinaten), um einen Radiusmittelpunkt (Mittelpunktskoordinaten) mit dem Drehwinkel W entgegen dem Uhrzeigersinn.
- Ziel-Koordinatenangaben können absolut (G90 | **ABS**) oder relativ (G91 | **REL**) erfolgen
 - Mittelpunkts-Koordinatenangaben erfolgen immer relativ bezüglich des Start-Punktes
 - die Angabe des Drehwinkels definiert die Anzahl der Umdrehungen; 360° definiert 1 Vollkreis

- Beispiel: ; Gewindefräsen im vorgebohrten Loch
; Radius = 5 mm, Spirale mit 10 Vollkreisen



```
ISO: N10 G17 ; Interpolationsebene einstellen
      N20 G0 G90 X5 Y0 Z-10 ; Anfangspunkt anfahren
      N30 G13 W3600 X5 Y0 Z0 I-5 J0 ; Helix abfahren

PAL: N10 PLANE XY ; Interpolationsebene einstellen
      N20 FASTABS X5 Y0 Z-10 ; Anfangspunkt anfahren
      N30 CCWHLXABS W3600 X5 Y0 Z0 I-5 J0 ; Helix abfahren
```

- Verweis: **G12, G2, G3** | **CWHLXABS, CWABS, CCWABS**

3.1.1.11 alle Bewegungsbefehle

alle Bewegungsbefehle	programmierbarer Abbruch von Bewegungen im Automatik-Betrieb
------------------------------	---

□ Syntax: In jeder Programmzeile, die eine Bewegung der Achsen des mechanischen Systems bewirkt (alle **G0-**, **G1-**, **G2/G3-**, **G10-** und **G11-Befehle** | **FASTABS-**, **MOVEABS-**, **CWABS/CCWABS-**, **FASTFRAME-** und **MOVEFRAME-Befehle**) kann ein Eingang unter Angabe der Port- und der Bit-Nummer programmiert werden. Erfolgt während der Bewegung zum programmierten Zielpunkt eine low-high-Flanke bzw. high-low Flanke des entsprechenden Einganges, wird die Bewegung abgebrochen.

□ Erklärung: ProNC verfügt über die Fähigkeit, im Automatikbetrieb Bewegungen bei der Aktivierung eines programmierten binären Einganges abzubrechen und sofort mit dem im Anwenderprogramm folgenden Befehl fortzusetzen.



Diese Funktionalität kann genutzt werden, wenn ProNC eine Bewegungssteuerung (MCTL) beauftragt für Servoanlagen (numerische Achsen mit DC-/AC-Servomotoren, isel-Servo-Controller CV mit Steckkarte UPMV4/12 oder isel-CAN-Controller) resident im Speicher erkennt. Das heißt, der programmierbare Abbruch von Bewegungen im Automatik-Betrieb ist bei folgenden Anlagen mit Schrittmotoren nicht nutzbar:

- Anlagen mit Controller C116-4 / C142-4
- alle Maschinen der CPM-Reihe (CPM 2018, CPM 3020, CPM 4030)
- Maschine der GFM-Reihe (GFM 4433)

□ Beispiel: ; die Bewegung zum Zielpunkt X=100mm, Y=200mm
; wird abgebrochen, wenn der binäre Eingang
; E1.1 während der Bewegung aktiviert wird (low-high-Flanke):
ISO: N10 G1 X100 Y200 E1.1
PAL: N10 **MOVEABS** X100 Y200 E1.1



; die Bewegung zur Zielposition in Q5
; wird abgebrochen, wenn der binäre Eingang
; E4.7 während der Bewegung aktiviert wird (high-low-Flanke):
ISO: N10 G10 Q5 NOT E4.7
PAL: N10 **FASTFRAME** Q5 NOT E4.7

3.1.1.12 Definition der Interpolationsebene

G17-Befehl	Definition der Interpolationsebene (X-Y-Ebene)	PLANE XY-Befehl
G18-Befehl	Definition der Interpolationsebene (X-Z-Ebene)	PLANE XZ-Befehl
G19-Befehl	Definition der Interpolationsebene (Y-Z-Ebene)	PLANE YZ-Befehl

<input type="checkbox"/> Syntax:	[Satznummer]? G17 oder G18 oder G19 [weiterer Befehl: G53, G54, G55, G56, G70, G71, G90, G91]* [F-Befehl]? [S-Befehl]? [M-Befehl]*	[Satznummer]? PLANE XY oder PLANE XZ oder PLANE YZ [weiterer Befehl: WPCLEAR, WPREG1, WPREG2, WPZERO, INCH, METRIC, ABS, REL]* [F-Befehl]? [S-Befehl]? [Zusatz-Befehl]*
----------------------------------	--	---

- Erklärung:
- Interpolationsebene auswählen:**
- G17 | PLANE XY: die X-Y-Ebene wird ausgewählt**
- alle nachfolgenden Kreisbefehle (G2 oder G3 | **CWABS** oder **CCWABS**) beziehen sich auf die X-Y-Ebene
 - der G17-Befehl | **PLANE XY-Befehl** ist in jedem Bearbeitungsprogramm voreingestellt; das heißt, dieser Befehl muss nicht explizit programmiert werden
- G18 | PLANE XZ: die X-Z-Ebene wird ausgewählt**
- alle nachfolgenden Kreisbefehle (G2 oder G3 | **CWABS** oder **CCWABS**) beziehen sich auf die X-Z-Ebene
- G19 | PLANE YZ: die Y-Z-Ebene wird ausgewählt**
- alle nachfolgenden Kreisbefehle (G2 oder G3 | **CWABS** oder **CCWABS**) beziehen sich auf die Y-Z-Ebene

Beispiel: ; die X-Y-Ebene wird ausgewählt:

ISO: N10 G17
PAL: N10 **PLANE XY**



; die X-Z-Ebene wird ausgewählt:

ISO: N10 G18
PAL: N10 **PLANE XZ**

; die Y-Z-Ebene wird ausgewählt:

ISO: N10 G19
PAL: N10 **PLANE YZ**

Verweis: **G2, G3** | **CWABS, CCWABS**

3.1.1.13 Nullpunkt einrichten

G53-Befehl	Nullpunktverschiebung deaktivieren	WPCLEAR-Befehl
G54/G55-Befehl	Nullpunktverschiebung 1 / 2 aktivieren	WPREG1/WPREG2-Befehl
G56-Befehl	Werkstücknullpunkt auf der aktuellen Position einrichten	WPZERO-Befehl

□ Syntax:	[Satznummer]? G53 oder G54 oder G55 oder G56 [weiterer Befehl: G17, G18, G19, G70, G71, G90, G91]* [F-Befehl]? [S-Befehl]? [M-Befehl]*	[Satznummer]? WPCLEAR oder WPREG1 oder WPREG2 oder WPZERO [weiterer Befehl: PLANE XY, PLANE XZ, PLANE YZ, INCH, METRIC, ABS, REL]* [F-Befehl]? [S-Befehl]? [Zusatz-Befehl]*
-----------	--	--

□ Erklärung:	G53 WPCLEAR : Nullpunktverschiebung deaktivieren
	G54 WPREG1 - Nullpunktverschiebung 1 aktivieren : der aktuelle Nullpunkt des Werkstück-Koordinatensystems wird gegenüber dem Nullpunkt des Maschinen-Koordinatensystems absolut um die Werte im Nullpunkt-Register 1 verschoben
	G55 WPREG2 - Nullpunktverschiebung 2 aktivieren : der aktuelle Nullpunkt des Werkstück-Koordinatensystems wird gegenüber dem Nullpunkt des Maschinen-Koordinatensystems absolut um die Werte im Nullpunkt-Register 2 verschoben
	G56 WPZERO : auf der aktuellen Position wird ein neuer Nullpunkt eingerrichtet

□ Beispiel:	; die mit G54 WPREG1 oder G55 WPREG2 definierte Verschiebung des Werkstücknullpunktes wird aufgehoben oder der mit G56 WPZERO eingerrichtete Nullpunkt wird gelöscht:
-------------	--



ISO: N10 G53
PAL: N10 **WPCLEAR**

; Einrichten eines neuen Werkstücknullpunktes mit Hilfe des Nullpunkt-
Registers **1**:

ISO: N20 G54
PAL: N20 **WPREG1**

; Einrichten eines neuen Werkstücknullpunktes mit Hilfe des Nullpunkt-
Registers **2**:

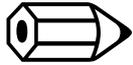
ISO: N30 G55
PAL: N30 **WPREG2**

; Einrichten eines neuen Werkstücknullpunktes auf der aktuellen Position:

ISO: N40 G56
PAL: N40 **WPZERO**

□ Verweis:	G92, G93 WPREG1WRITE, WPREG2WRITE
------------	---

□ Beispiel:



; die in den Q-Variablen Q1 bis Q4 gespeicherten Ziel-Punkte werden zu
; einer Bahn zusammengefasst:
%L200
; Unterprogramm zur Demonstration des Bahn-Betriebes (CP):
; -> Fräsbearbeitung mit Bahngeschwindigkeit von 10 mm/sec:
; Bahnbetrieb einschalten:

ISO: N5 G99 Bahnfahrt einschalten ...
N1 G64
N10 G11 Q1 F10.0
N20 G11 Q2
N30 G11 Q3
N40 G11 Q4
N50 G60
N60 G99 Bahnfahrt beendet ...
N70 M17

PAL: N5 **TYPE** Bahnfahrt einschalten ...
N1 **PATH**
N10 **MOVEFRAME** Q1 F10.0
N20 **MOVEFRAME** Q2
N30 **MOVEFRAME** Q3
N40 **MOVEFRAME** Q4
N50 **PATHEND**
N60 **TYPE** Bahnfahrt beendet ...
N70 **RETURN**

□ Hinweis:

Bahnbetrieb ist bei den Bewegungssteuerungen für IMS6-Controller, die Servo-Karte UPMV 4/12 bzw. CAN-Controller möglich.

3.1.1.15 Definition Maßeinheit

G70-Befehl	Definition der Maßeinheit für translatorische Achsen: inch	INCH-Befehl
G71-Befehl	Definition der Maßeinheit für translatorische Achsen: mm	METRIC-Befehl

<input type="checkbox"/> Syntax:	[Satznummer]? G70 oder G71 [weiterer Befehl: G17, G18, G19, G53, G54, G55, G90, G91]* [F-Befehl]? [S-Befehl]? [M-Befehl]*	[Satznummer]? INCH oder METRIC [weiterer Befehl: PLANE XY, PLANE XZ, PLANE YZ, WPCLEAR, WPREG1, WPREG2, ABS, REL]* [F-Befehl]? [S-Befehl]? [Zusatz-Befehl]*
----------------------------------	---	---

- Erklärung:
- G70 | INCH:**
Allen Koordinatenangaben für Linearachsen wird die **Maßeinheit inch** zugeordnet.
- G71 | METRIC:**
Allen Koordinatenangaben für Linearachsen wird die **Maßeinheit mm** zugeordnet (Standardeinstellung).

- Beispiel:
- kartesische Kinematiken (XYZ):**
; Gerade im Raum zum Absolut-Ziel-Punkt mit den
; Koordinaten (100 inch, 200 inch, 300 inch) mit
; Segmentgeschwindigkeit:



ISO: N100 **G70** G1 X100.0 Y200.0 Z300.0
PAL: N100 **INCH MOVEABS** X100.0 Y200.0 Z300.0

kartesische Kinematiken (XYZ):
; Gerade im Raum zum Ziel-Punkt mit den Koordinaten
; X-IST + 10mm, Y-IST + 20 mm, Z-IST - 30mm
; mit Segmentgeschwindigkeit:
ISO: N200 G91 **G71** G1 X10.0 Y20.0 Z-30.0
PAL: N200 **MOVEREL METRIC** X10.0 Y20.0 Z-30.0

nicht-kartesische Kinematik mit 3 Drehachsen und Z-Achse (z. B. Scara-Roboter):
; Absolut-Bewegung zum Ziel-Punkt mit den Werten:
; Fußdrehung: 100 grd / Vertikalbewegung: 180 **inch**
; Ellenbogengelenk: 45.0 grd / Hand drehen: -45.0 grd
; mit Eilganggeschwindigkeit:
ISO: N100 **G70** G0 C100.0 Z180.0 B45.0 A-45.0
PAL: N100 **INCH FASTABS** C100.0 Z180.0 B45.0 A-45.0

<input type="checkbox"/> Verweis:	G0, G1, G2, G3	FASTABS, MOVEABS, CWABS, CCWABS
-----------------------------------	----------------	--

3.1.1.16 Referenzfahrt

G74-Befehl	Referenzpunktfahrt	REF-Befehl
<input type="checkbox"/> Syntax:	[Satznummer]? [weiterer Befehl: G17, G18, G19, G70, G71, G90, G91]* G74 [adressbuchstabe]? [M-Befehl]*	[Satznummer]? [weiterer Befehl: PLANE XY, PLANE XZ, PLANE YZ, INCH, METRIC, ABS, REL]* REF [adressbuchstabe]? [Zusatz-Befehl]*
<input type="checkbox"/> Erklärung:	<p>Ausführung einer Referenzpunktfahrt:</p> <ul style="list-style-type: none"> • adressbuchstabe = {X, Y, Z, A, B, C} für Achssystem 1 • adressbuchstabe = {X2, Y2, Z2, A2, B2, C2} für Achssystem 2 (d.h. der Adressbuchstabe ist ein Element der angegebenen Menge) • wird der G74-Befehl REF-Befehl ohne Argument benutzt, werden alle Achsen in der Reihenfolge Z-Achse → Y-Achse → X-Achse → A-Achse → B-Achse → C-Achse synchronisiert • nach einer Referenzpunktfahrt ist das Bewegungsmodul rückgesetzt, d.h. dass ein eventuell definierter Nullpunkt gelöscht wurde und alle Initialeinstellungen (z.B. Segment- / Vorschubgeschwindigkeit) gelten 	
<input type="checkbox"/> Beispiel:	; Referenzpunktfahrt für alle Achsen mit der in der ; Initialisierungsdatei des Bewegungsmoduls definierten ; Geschwindigkeit:	
	ISO: N10 G74 PAL: N10 REF	
	; Referenzpunktfahrt mit nur einer Achse: ISO: N10 G74 X ; Referenzpunktfahrt der X-Achse N20 G74 C ; Referenzpunktfahrt der C-Achse PAL: N10 REF X ; Referenzpunktfahrt der X-Achse N20 REF C ; Referenzpunktfahrt der C-Achse	
<input type="checkbox"/> Verweis:	G70, G71, G90, G91	INCH, METRIC, ABS, REL

3.1.1.17 Teach

G75-Befehl	programmierbare Korrektur von Achspositionen	TEACH-Befehl
-------------------	---	---------------------

<input type="checkbox"/> Syntax:	[Satznummer]? G75	[Satznummer]? TEACH
----------------------------------	-----------------------------	-------------------------------

Erklärung: Mit dem Befehl **G75** | **TEACH** kann im Anwenderprogramm das Fenster aktuelle Geometriedatei: ... zur Laufzeit des Anwenderprogramms aktiviert werden.

Damit können Korrekturen von Achspositionen / Teach-In (Neueingabe oder Aktualisierung von Geometrieinformationen der aktuellen Geometriedatei) online vorgenommen werden, ohne den Automatikbetrieb verlassen zu müssen.

Das Fenster aktuelle Geometriedatei: ... wird mit OK oder Abbrechen verlassen, danach wird unmittelbar im Anwenderprogramm mit dem Befehl / Anweisung nach **G75** | **TEACH** fortgesetzt.

Die **aktuelle Geometriedatei** ist jene, welche im Verzeichnis **CNCWorkbench\NCProg\Frame** steht und den **gleichen Dateiname** besitzt wie das aktuelle ISO- bzw. PAL-Anwenderprogramm.

Zu beachten ist die Besonderheit, dass lediglich die **Dateiextension** (ISO: name.iso | PAL: name.pal) durch die für Geometriedateien typische Extension **fra** ersetzt ist.

Das heißt, wenn Sie das PAL-Anwenderprogramm ABC.PAL abarbeiten, können Sie mit der Geometriedatei ABC.FRA arbeiten.

Möchten Sie dagegen mehrere Anwenderprogramme mit einer Geometriedatei abarbeiten, dann benutzen Sie bitte die Standardgeometriedatei stdframe.fra im Verzeichnis \CNCWorkbench\Bin.

Beispiel: ;Teach-In



ISO:	N10 G75
PAL:	N10 TEACH

Verweis: Bedienungsanleitung:
 5.7.3.9 Menü Steuerung - Manuell fahren
 5.7.3.10 Menü Steuerung - Maschinenpositionen bearbeiten
 2.2.2. Die Geometriedatei

3.1.1.18 Bohrzyklus definieren

G80	Definition eines Bohrzyklus	DRILLDEF
------------	------------------------------------	-----------------

□ Syntax: [Satznummer]? **G80** | [Satznummer]? **DRILLDEF**

CY: Bohrzyklus

- 1 = Einfaches Bohren
- 2 = Ausräumen
- 3 = Spanbrechen

PL: Arbeitsebene

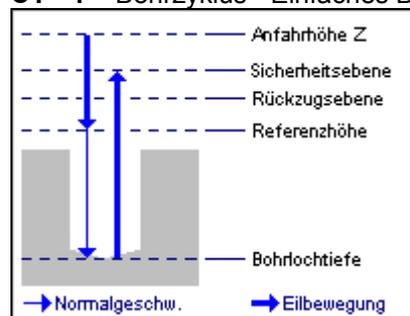
- 0 = XY
- 1 = XZ
- 2 = YZ

DI: Arbeitsrichtung

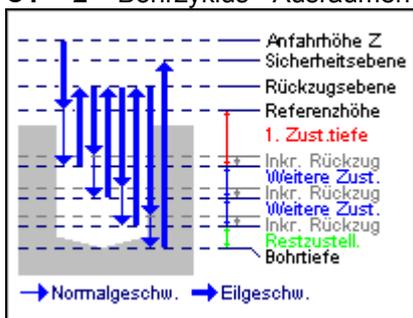
- 0 = Standard
- 1 = Invers

RF:	Referenzhöhe	(mm)
DE:	Bohrlochtiefe	(mm)
TI:	Verzögerung nach Erreichen der Tiefe	(s)
VE:	Vorschubgeschwindigkeit	(mm/s)
VF:	Eilganggeschwindigkeit	(mm/s)
FI:	erste inkrementelle Zustelltiefe	(mm)
OT:	weitere inkrementelle Zustelltiefe	(mm)
IC:	Abnahme der inkrementellen Zustelltiefe	(mm)
RE:	inkrementeller Rückzug	(mm)
LE:	Rückzugsebene nach Bohren	(mm)
SE:	Sicherheitshöhe	(mm)

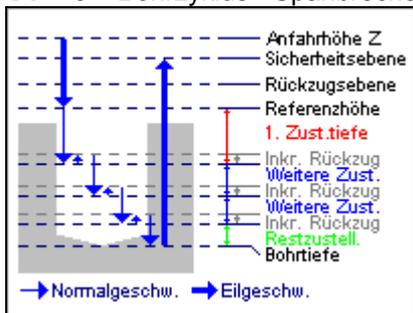
- Erklärung:
- es werden alle Bohrparameter für den Bohrbefehl Drill definiert
 - Bohrparameter sind modal, d. h. die Parameter bleiben solange gültig, bis sie neu gesetzt werden
 - am Anfang des Programms können Standardparameter definiert werden, unmittelbar vor dem Drill-Befehl können einzelne Parameter modifiziert werden

CY =1 Bohrzyklus - Einfaches Bohren

□ Erklärung: **CY = 2 Bohrzyklus - Ausräumen**



□ Erklärung: **CY = 3 Bohrzyklus - Spanbrechen**



□ Beispiel: ; CY: Bohrzyklus = 1 (Einfaches Bohren)



- ; PL: Arbeitsebene = 0 (XY)
- ; DI: Arbeitsrichtung = 0 (Standard)
- ; RF: Referenzhöhe = 1 mm
- ; DE: Bohrlochtiefe = 4 mm
- ; TI: Verzögerung nach Erreichen der Tiefe = 20 s
- ; VE: Vorschubgeschwindigkeit = 3 mm/s
- ; VF: Eilgeschwindigkeit = 20 mm/s
- ; FI: Erste inkrementelle Zustelltiefe = 4 mm
- ; OT: Weitere inkrementelle Zustelltiefen = 10 mm
- ; IC: Abnahme der inkrementellen Zustelltiefe = 7 mm
- ; RE: Inkrementeller Rückzug = 4 mm
- ; LE: Rückzugsebene nach Bohren = 3 mm
- ; SE: Sicherheitshöhe = 5 mm

ISO: N10 **G80** CY1 PL0 DI0 RF1 DE4 TI20 VE3 VF20 FI4 OT10 IC7 RE4 LE3 SE5

PAL: N10 **DrillDef** CY1 PL0 DI0 RF1 DE4 TI20 VE3 VF20 FI4 OT10 IC7 RE4 LE3 SE5

□ Verweis: G81, G82; G83, G84 | [DrillN](#), [DrillT](#), [DrillD](#), [DrillB](#)

3.1.1.19 Bohrzyklus ausführen

G81 G82 G83 G84	Einfaches Bohren Bohren mit Verweilzeit Bohren (Betriebsart Ausräumen) Bohren (Betriebsart Spanbrechen)	DrillN DrillT DrillD DrillB
--	--	--

<input type="checkbox"/> Syntax:	[Satznummer]? G81 oder G82 oder G83 oder G84 Koordinaten {x,y}	[Satznummer]? DrillN oder DrillT oder DrillD oder DrillB Koordinaten {x,y}
----------------------------------	--	--

<input type="checkbox"/> Erklärung:	ISO: G81: Normaler Bohrzyklus G82: Time G83: Tiefloch bohren G84: Zyklus 3 (Spanbrechen)	PAL: DrillN: Normaler Bohrzyklus DrillT: Time DrillD: Tiefloch bohren DrillB: Zyklus 3 (Spanbrechen)
-------------------------------------	--	--

<input type="checkbox"/> Erklärung:	<ul style="list-style-type: none"> • Starten des Bohrzyklus • die erforderlichen Parameter für den Bohrvorgang müssen im Befehl G80 DrillDef definiert werden • entsprechend der Kennung im Bohrbefehl (G81, G82, G83, G84 DrillN, DRILLT, DRILLD, DRILLB) werden die Werte aus der Definition mit G80 DrillDef entnommen
-------------------------------------	--

 Beispiel:

; einfaches Bohren an der Position X=20, Y=100


ISO: N10 G81 X20 Y100
PAL: N10 **DRILLN** X20 Y100

<input type="checkbox"/> Verweis:	G80	DrillDef
-----------------------------------	-----	----------

3.1.1.20 Koordinatenangaben

G90-Befehl	Koordinatenangaben sind Absolutangaben (Absolutmaß)	ABS-Befehl
G91-Befehl	Koordinatenangaben sind Relativangaben (Kettenmaß)	REL-Befehl

<input type="checkbox"/> Syntax:	[Satznummer]? [weiterer Befehl]: G17, G18, G19, G53, G54, G55, G70, G71]* G90 oder G91 [F-Befehl]? [S-Befehl]? [M-Befehl]*	[Satznummer]? [weiterer Befehl]: PLANE XY, PLANE XZ, PLANE YZ, WPCLEAR, WPREG1, WPREG2, INCH, METRIC]* ABS oder REL [F-Befehl]? [S-Befehl]? [Zusatz-Befehl]*
----------------------------------	---	---

- Erklärung: **G90 | ABS:** alle Zielkoordinatenangaben sind Absolutangaben (Absolutmaß)
- G91 | REL:** alle Zielkoordinatenangaben sind Relativangaben (Kettenmaß)

Beispiel:



kartesische Kinematiken (XYZ):
 ; Gerade im Raum zum Absolut-Ziel-Punkt mit den
 ; Koordinaten (100mm, 200mm, 300mm) mit
 ; aktuellem Vorschub:

```
ISO: N100 G01 G90 X100.0 Y200.0 Z300.0
PAL: N100 MOVEABS X100.0 Y200.0 Z300.0
```

kartesische Kinematiken (XYZ):
 ; Gerade im Raum zum Ziel-Punkt mit den Koordinaten
 ; X-START + 10mm, Y-START + 20 mm, Z-START - 30mm
 ; mit aktuellem Vorschub:

```
ISO: N200 G01 G91 X10.0 Y20.0 Z-30.0
PAL: N200 MOWEREL X10.0 Y20.0 Z-30.0
```

<input type="checkbox"/> Verweis:	G0, G1, G2, G3	FASTABS, MOVEABS, CWABS, CCWABS
-----------------------------------	-------------------	------------------------------------

3.1.1.21 Speicher setzen

G92-Befehl	Speicher setzen (Werkstücknullpunktregister 1)	WPREG1WRITE-Befehl
G93-Befehl	Speicher setzen (Werkstücknullpunktregister 2)	WPREG2WRITE-Befehl

□ Syntax:	[Satznummer]? [weiterer Befehl: G17, G18, G19, G70, G71, G90, G91]* G92 oder G93 [Ziel-Koordinaten]{1,6} oder Frame-Name [F-Befehl]? [S-Befehl]? [M-Befehl]*	[Satznummer]? [weiterer Befehl: PLANE XY, PLANE XZ, PLANE YZ, INCH, METRIC, ABS, REL]* WPREG1WRITE oder WPREG2WRITE Ziel-Koordinaten]{1,6} oder Frame-Name [F-Befehl]? [S-Befehl]? [Zusatz-Befehl]*
-----------	--	---

□ Erklärung: **G92 | WPREG1WRITE: Nullpunktregister 1 setzen**

mit einem nachfolgenden G54-Befehl | **WPREG1-Befehl** kann eine neue Nullpunktverschiebung aktiviert werden; die Absolutwerte für jede Koordinate werden in das sogenannte Nullpunkt-Register 1 geschrieben und erst mit dem folgenden G54-Befehl | **WPREG1-Befehl** an die Bewegungssteuerung übergeben

G93: | WPREG2WRITE: Nullpunktregister 2 setzen

mit einem nachfolgenden G55-Befehl | **WPREG2-Befehl** kann eine neue Nullpunktverschiebung aktiviert werden; die Absolutwerte für jede Koordinate werden in das sogenannte Nullpunkt-Register 2 geschrieben und erst mit dem folgenden G55-Befehl | **WPREG2-Befehl** an die Bewegungssteuerung übergeben

□ Beispiel: ; Nullpunktverschieberegister 1 setzen (die Koordinaten des Nullpunktes ; werden im Befehl direkt angegeben):



ISO: N10 **G92** X100 Y200 Z300
PAL: N10 **WPREG1WRITE** X100 Y200 Z300

; Nullpunktverschiebung aktivieren:

ISO: N20 G54
PAL: N20 **WPREG1**

; Nullpunktverschieberegister 2 setzen (die Koordinaten des Nullpunktes ; werden dem durch den Framenamen definierten Frame in der aktuellen ; Geometriedatei entnommen):

ISO: N10 **G93 NULLPUNKT1** ; Register 2 laden
N20 G55 ; Nullpunkt aktivieren

PAL: N10 **WPREG2WRITE NULLPUNKT1** ; Register 2 laden
N20 **WPREG2** ; Nullpunkt aktivieren

□ Verweis: G53, G54, G55, G56 | **WPCLEAR, WPREG1, WPREG2, WPZERO**

3.1.2 Zusatzbefehle

In der folgenden Tabelle sind die in ProNC benutzten Zusatzbefehle zusammengefasst:

<u>Zusatzbefehl</u>	<u>Bedeutung</u>	<u>Zusatzbefehl</u>
M0	programmierte Programmunterbrechung (Abbruch)	ABORT
M1	programmierte Programmunterbrechung (Halt)	QUIT
M3	Spindel einschalten (Drehrichtung im Uhrzeigersinn)	SCLW Spindle Cw
M4	Spindel einschalten (entgegen dem Uhrzeigersinn)	SCCLW Spindle Ccw
M5	Spindel ausschalten	SOFF Spindle off
M8/M9	Kühlmittel ein/ aus	Coolant on/off
M10/M11	Werkstück spannen ein/aus	WpClamp on/off
	Pumpe ein/aus	Pump on/off
	Beleuchtung ein/aus	Lamp on/off
	Peripherieoption 1 ein/aus Peripherieoption 2 ein/aus	Poption 1 on/off Poption 2 on/off
	Hand-Modus aus / ein	HOFF/HON
	Test-Modus aus / ein	TOFF/TON
M17	Rückkehr aus Unterprogramm	RETURN
M30	Programmende	PROGEND
	Eingänge lesen	GetPort GetP GetBit
Mpby	Ausgänge setzen	SetPort SetP SetBit SetAnalog SetPWM
	Istwerte abfragen	PosA.n GetDate GetTime GetValue

Tabelle 3.1.2: Zusatzbedingungen in ProNC (Übersicht)

3.1.2.2 Programmbeginn, Programmende

% M30-Befehl	Programmbeginn Programmende	ProgBegin ProgEnd
-------------------------	--	------------------------------

Syntax: % oder M30 | ProgBegin oder ProgEnd

Hinweis: Programmbeginn kennzeichnet immer den Anfang des Hauptprogramms. Vor einem Hauptprogramm können Unterprogramme deklariert werden, welche dann im Hauptprogramm aufgerufen werden können.

Erklärung: % | ProgBegin: Programmbeginn
M30 | ProgEnd: Programmende

Verweis: 3.1.7 Unterprogrammtechnik

3.1.2.3 Spindelbefehle

M3-Befehl	Spindel einschalten (Drehrichtung clw)	SCLW-Befehl
M4-Befehl	Spindel einschalten (Drehrichtung cclw)	SCCLW-Befehl
M5-Befehl	Spindel ausschalten	SOFF-Befehl

□ Syntax:	[Satznummer]? [weiterer Befehl]: [Koordinaten]{0,6} [F-Befehl]? [S-Befehl]? M3 oder M4 oder M5	[Satznummer]? [weiterer Befehl]: [Koordinaten]{0,6} [F-Befehl]? [S-Befehl]? SCLW oder SCCLW oder SOFF
-----------	--	---

- Erklärung:
- M3 | SCLW: Spindel einschalten**
- Drehrichtung: clw - im Uhrzeigersinn / Rechtslauf
- M4 | SCCLW: Spindel einschalten**
- Drehrichtung: cclw - entgegen dem Uhrzeigersinn / Linkslauf
 - die Drehzahl der Spindel wird mit dem S-Befehl eingestellt
 - obwohl der M-Befehl im NC-Satz am Ende des Satzes steht, wird der Spindel-Hochlauf gestartet, ehe eine Verfahrbewegung beginnt
 - bei den Befehlen M3 | SCLW und M4 | SCCLW wird eine definierte Hochlaufzeit der Arbeitsspindel gewartet, wenn eine Anlaufverzögerung/ Hochlaufzeit in der Initialisierungsdatei der ausgewählten spindel.dll (SETUP-Dialog) angegeben wurde.
- M5 | SOFF: Spindel ausschalten**
- bei den Befehlen M5 | SOFF wird eine definierte Auslaufverzögerung der Arbeitsspindel gewartet, wenn eine Auslaufverzögerung in der Initialisierungsdatei der ausgewählten spindel.dll (SETUP-Dialog) angegeben wurde.
- [siehe auch:](#)
Abschnitt 3.1.5 S-Befehl

- **Hinweis:** *Gleichberechtigt zur vormals beschriebenen Syntax ist ebenfalls folgende Notation zulässig:*



	Spindelbefehl	Spindle
--	----------------------	----------------

Syntax: [Satznummer]?

Spindle CW, RPM, [CCW], [RPS], [ON], [OFF], [TIME Millisekunde]

Erklärung: **Spindle CW: Spindel einschalten** im Uhrzeigersinn / Rechtslauf

Spindle CCW: Spindel einschalten entgegen dem Uhrzeigersinn / Linkslauf

Spindle ON: Spindel einschalten im zuletzt eingestelltem Modus (Rechts- oder Linkslauf)

Spindle OFF: Spindel ausschalten

Parameter RPM: Festlegung der Spindeldrehzahl in U/min

Parameter RPS: Festlegung der Spindeldrehzahl in U/sec

Eine Programmverzögerung für den Anlauf der Spindel auf Nenndrehzahl kann durch den Parameter TIME angegeben werden.

Beispiel: ; Einschalten der Spindel im Rechtslauf mit einer Drehzahl

; von 5.000 U/min, 5 Sekunden warten

N10 **Spindle** CW RPM5000 TIME 5000

3.1.2.4 Kühlmittel

M8-Befehl/ M9-Befehl	Kühlmittel ein/aus	COOLANT ON COOLANT OFF
---------------------------------	---------------------------	-----------------------------------

- Syntax: [Satznummer]?
[weiterer Befehl:
[Koordinaten]{0,6}
[F-Befehl]?
[S-Befehl]?
M8 oder M9
- [Satznummer]?
[weiterer Befehl:
Koordinaten]{0,6}
[F-Befehl]?
[S-Befehl]?
COOLANT ON oder COOLANT OFF
- Erklärung: **M8 | COOLANT ON:** Kühlflüssigkeit ein
M9 | COOLANT OFF: Kühlflüssigkeit aus
- obwohl der Zusatzbefehl im NC-Satz am Ende des Satzes steht, wird die Aktivität gestartet, ehe eine Verfahrensbewegung beginnt
- Hinweis: Die Zuordnung, welcher binäre Ausgang den Kühlmittelfluss ein- bzw. ausschaltet, erfolgt im Dialog Einstellungen - Steuerung - Ein-/Ausgabemodule - Erweiterte Einstellungen - Peripherie
- [siehe auch:](#)
Bedienungsanleitung: 5.8.7.3 Menü Steuerung - Ein-/ Ausgabemodule

3.1.2.5 Werkstück spannen

M10 M11	Werkstück spannen ein Werkstück spannen aus	WPCLAMP ON WPCLAMP OFF
--------------------	--	-----------------------------------

- Syntax: [Satznummer]?
- M10**
M11
- WPCLAMP ON**
WPCLAMP OFF
- Erklärung: aktivieren /deaktivieren einer pneumatisch o. ä. betriebenen Werkstück-Spannvorrichtung.
- M10 | WPCLAMP ON:** Werkstück **einspannen**
M11 | WPCLAMP OFF: Werkstück **ausspannen**
- Hinweis: Die Zuordnung, welcher binäre Ausgang den Kühlmittelfluss ein- bzw. ausschaltet, erfolgt im Dialog Einstellungen - Steuerung - Ein-/Ausgabemodule - Erweiterte Einstellungen - Peripherie
- [siehe auch:](#)
Bedienungsanleitung: 5.8.7.3 Menü Steuerung - Ein-/ Ausgabemodule

3.1.2.6 Pumpe

	Pumpe ein Pumpe aus	PUMP ON PUMP OFF
--	--------------------------------------	-----------------------------------

Syntax: [Satznummer]?

		PUMP ON PUMP OFF
--	--	-----------------------------------

Erklärung: einschalten/ausschalten einer Absaugvorrichtung oder Flüssigkeitspumpe

PUMP ON: Pumpe ein
PUMP OFF: Pumpe aus

Hinweis: Die Zuordnung, welcher binäre Ausgang den Kühlmittelfluss ein- bzw. ausschaltet, erfolgt im Dialog Einstellungen - Steuerung - Ein-/Ausgabemodule - Erweiterte Einstellungen - Peripherie

[siehe auch:](#)
Bedienungsanleitung: 5.8.7.3 Menü Steuerung - Ein-/ Ausgabemodule

3.1.2.7 Lampe

	Lampe ein Lampe aus	LAMP ON LAMP OFF
--	--------------------------------------	-----------------------------------

Syntax: [Satznummer]?

		LAMP ON LAMP OFF
--	--	-----------------------------------

Erklärung: einschalten/ausschalten einer Beleuchtung

LAMP ON: Lampe ein
LAMP OFF: Lampe aus

Hinweis: Die Zuordnung, welcher binäre Ausgang den Kühlmittelfluss ein- bzw. ausschaltet, erfolgt im Dialog Einstellungen - Steuerung - Ein-/Ausgabemodule - Erweiterte Einstellungen - Peripherie

[siehe auch:](#)
Bedienungsanleitung: 5.8.7.3 Menü Steuerung - Ein-/ Ausgabemodule

3.1.2.8 Peripherieoption

	Peripherieoption1 ein/aus Peripherieoption2 ein/aus	POPTION1 ON/OFF POPTION2 ON/OFF
--	--	--

Syntax: [Satznummer]?

	POPTION1 ON/POPTION1 OFF POPTION2 ON/POPTION2 OFF
--	--

Erklärung: einschalten/ausschalten einer optionalen Einrichtung (Möglichkeit des Anschlusses einer kundenspezifischen Hardware an das entsprechende Ausgabeport)

POPTION1 ON: Peripherieeinrichtung 1 **ein**
POPTION1 OFF: Peripherieeinrichtung 1 **aus**

POPTION2 ON: Peripherieeinrichtung 2 **ein**
POPTION2 OFF: Peripherieeinrichtung 2 **aus**

Hinweis: Die Zuordnung, welcher binäre Ausgang den Kühlmittelfluss ein- bzw. ausschaltet, erfolgt im Dialog Einstellungen - Steuerung - Ein-/Ausgabe-Module - Erweiterte Einstellungen - Peripherie

[siehe auch:](#)

Bedienungsanleitung: 5.8.7.3 Menü Steuerung - Ein-/ Ausgabemodule

3.1.2.9 Hand-/Test-Modus

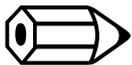
	Hand-Modus ausschalten /einschalten Test-Modus ausschalten /einschalten	HOFF/HON-Befehl TOFF/TON-Befehl
--	--	--

Erklärung: Der Hand-Modus bzw. der Test-Modus können im Anwenderprogramm ein- bzw. ausgeschaltet werden.
Der eingeschaltete Hand-Modus bewirkt, dass die Motorendstufen stromlos geschaltet werden.

Der Test-Modus ist per Programm einzuschalten, wenn z.B. eine 4. Achse (A-Achse) eine Referenzfahrt ohne Vorhandensein eines Referenzschalters ausführen soll.

Hinweis: Nicht alle Motorendstufen / Bewegungssteuerungen bieten die Möglichkeit, einen Hand- bzw. Testmodus ein- bzw. auszuschalten.

Beispiel:



PAL: **HOFF:** Hand-Modus ausschalten
HON: Hand-Modus einschalten
TOFF: Test-Modus ausschalten
TON: Test-Modus einschalten

3.1.2.10 *Eingänge lesen / Ausgänge rücklesen*

	Lesen Eingangs- / Ausgangsport	GetPort / GetP
--	---------------------------------------	-----------------------

□ Syntax:

[Satznummer]?

```

r_variable=GetPort Ep/
r_variable=GetP Ep
r_variable=GetPort Ap /
r_variable=GetP Ap

```

□ Erklärung:

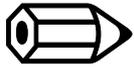
Lesen des Wertes entweder eines logischen Eingangsports p (Parameter Ep) oder des aktuellen Wertes eines Ausgangsports p (Parameter Ap).

□ Beispiel:

```

N10 R11=GetPort E1 ;Lesen von Eingangsport 1
N15 R12=GetPort A1 ;Akt. Wert von Ausgangsport 1 lesen
oder
N10 R11=GetP E1 ;Lesen von Eingangsport 1
N15 R12=GetP A1 ;Akt. Wert von Ausgangsport 1 lesen

```



3.1.2.11 Ausgänge setzen

Mpby-Befehl	Setzen Ausgangsport (y=1 SETB) setzen bzw. (y=0 RESB) rücksetzen	SetPort / SetP SetBit ResBit
--------------------	---	---

□ Syntax:	[Satznummer]?	[Satznummer]?
[Mpby]+		SetPort Ap=konstante SetP Ap=konstante SetBit Ap.b=y SetBit Ap.b ; immer setzen ResBit Ap.b ; immer rücksetzen

- Erklärung: **Mpby**: allgemeiner M-Befehl zum Setzen / Rücksetzen von binären Ausgängen:
- der Buchstabe p steht für eine Ziffer 1, 2 ... 8; diese Ziffer definiert das entsprechende Ausgangsport;
 - ein Ausgangsport umfasst immer die acht Ausgangsbits 1 bis 8
 - der Buchstabe b steht für eine Ziffer 1, 2 ... 8, diese Ziffer definiert das entsprechende Bit im Ausgangsport;
 - die Ausgangsports entsprechen den P-Variablen mit geradem Index:

ISO:

- M1by** bezieht sich auf das Ausgangsport 1 = P0
- M2by** bezieht sich auf das Ausgangsport 2 = P2
- M3by** bezieht sich auf das Ausgangsport 3 = P4
- M4by** bezieht sich auf das Ausgangsport 4 = P6
- M5by** bezieht sich auf das Ausgangsport 5 = P8
- M6by** bezieht sich auf das Ausgangsport 6 = P10
- M7by** bezieht sich auf das Ausgangsport 7 = P12
- M8by** bezieht sich auf das Ausgangsport 8 = P14

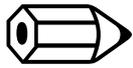
- der Buchstabe y steht für eine Ziffer 0 oder 1, diese Ziffer definiert, ob das mit b ausgewählte Bit gesetzt (y = 1) oder rückgesetzt (y=0) werden soll

PAL:

- SetBit / ResBit A1.b** bezieht sich auf das Ausgangsport 1 = P0
- SetBit / ResBit A2.b** bezieht sich auf das Ausgangsport 2 = P2
- SetBit / ResBit A3.b** bezieht sich auf das Ausgangsport 3 = P4
- SetBit / ResBit A4.b** bezieht sich auf das Ausgangsport 4 = P6
- SetBit / ResBit A5.b** bezieht sich auf das Ausgangsport 5 = P8
- SetBit / ResBit A6.b** bezieht sich auf das Ausgangsport 6 = P10
- SetBit / ResBit A7.b** bezieht sich auf das Ausgangsport 7 = P12
- SetBit / ResBit A8.b** bezieht sich auf das Ausgangsport 8 = P14

siehe auch:
Abschnitt 3.2.1.1: P-Variable

□ Beispiel:



```
; Setzen des Ausgangsports 1 - binäre Schreibweise:  
SetPort A1=11110000B  
; Setzen des Ausgangsports 1 - hexadezimale Schreibweise:  
SetPort A1=0xF0  
oder  
SetPort A1= $F0  
oder  
SetPort A1=F0H
```

[siehe auch:](#)

Abschnitt 3: Sprachbeschreibung, Zahlenformate

□ Beispiel:



```
ISO: ; Bit 5 im Ausgangsport 1 setzen:  
N10 M151
```

```
; Bit 7 im Ausgangsbyte 2 rücksetzen:  
N20 M270
```

```
PAL: ; Bit 5 im Ausgangsbyte 1 setzen:  
N10 SetBit A1.5; oder:  
N10 SetBit A1.5=1
```

```
; Bit 7 im Ausgangsbyte 2 rücksetzen:  
N20 ResBit A2.7; oder:  
N20 SetBit A2.7=0
```

3.1.2.12 Setzen Analog-/PWM-Ausgang

	Setzen Analogausgang Setzen PWM-Ausgang	SetAnalog SetPWM
--	--	-----------------------------------

Syntax:

[Satznummer]?

	SetAnalog Ak = Spannungswert [mV] SetPWM Ak =Tastverhältnis [%]
--	---

Erklärung:

- Setzen des Wertes des analogen Ausgangs mit der übergebenen Kanalnummer. Die Angabe des Analogwertes erfolgt in Millivolt. Der angesprochene Analogausgang muss in der erweiterten IO-Verwaltung deklariert sein.
- Setzen des Wertes des PWM-Signals mit der übergebenen Kanalnummer k [1 ...4]. Die Angabe des Tastverhältnisses erfolgt in Prozent. Der angesprochene PWM-Kanal muss in der erweiterten IO-Verwaltung deklariert sein.

Beispiel:

; Setzen des Analogausgangs 1 auf 2,5 Volt
N10 SetAnalog A1=2500



; Setzen des Tastverhältnisses am PWM-Ausgang 1 auf 50%
N10 SetPWM A1=50 ;Tastverhältnis 50%

3.1.2.13 Aktuelle Achsposition

	Aktueller Wert einer Achsposition	POS_n.A
--	--	--------------------------

Syntax: [Satznummer]?

r_variable = POS_n.A

n=[1,2]

A=[X, Y, Z, A, B, C]

Erklärung: Funktion zur Abfrage der aktuellen Position einer Achse a des Achssystems n. Es sind die Achssysteme 1 und 2 indexierbar. Für a kann einer der Achsbezeichner X, Y, Z, A, B, C eingesetzt werden.

Beispiel: R11=POS1.X ;R11 enthält die Position X im Achssystem 1
R12=POS1.Y ;R12 enthält die Position Y im Achssystem 1
R21=POS2.X ;R11 enthält die Position X im Achssystem 2



3.1.2.14 Aktuelle Systemzeit

	Aktuelle Systemzeit	GetTime
--	----------------------------	----------------

Syntax: [Satznummer]?

**r_variable0=GetTime r_variable1
r_variable2 r_variable3**

Erklärung: Abfrage der aktuellen Systemzeit. Nach Aufruf dieser Funktion stehen in den R-Variablen diese Werte zur Verfügung:

r_variable1: = Stunde

r_variable2: = Minute

r_variable3:= Sekunde

Beispiel: N10 R0=GetTime R11 R12 R13; R11=Stunde, R12=Minute, R13=Sekunde



Hinweis: Nur wenn in R0 nach Ausführung des Befehls GetTime der Wert 0 steht, sind die Werte in R11 (Stunde), R12(Minute) und R13 (Sekunde) richtig.

3.1.2.15 Aktuelles Datum

	Aktuelles Datum	GetDate
--	------------------------	----------------

□ Syntax: [Satznummer]?
`r_variable0=GetDate r_variable1 r_variable2 r_variable3`

□ Erklärung: Abfrage des aktuellen Datums. Nach Aufruf dieser Funktion stehen in den R-Variablen diese Werte zur Verfügung:

r_variable1: = Jahr
 r_variable2: = Monat
 r_variable3: = Tag

□ Beispiel: N10 R0=GetDate R1 R2 R3; R1=Jahr, R2=Monat, R3=Tag

□ Hinweis: Nur wenn in R0 nach Ausführung des Befehls GetDate der Wert 0 steht, sind die Werte in R1 (Jahr), R2(Monat) und R3 (Tag) richtig.

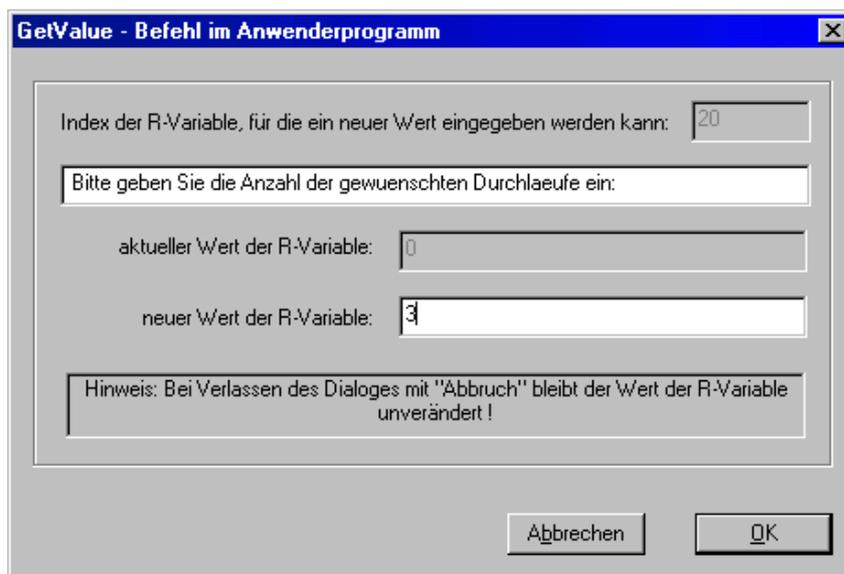
3.1.2.16 Dialogfeld für Anwendertexteingabe

	Dialogfeld für Anwendereingabe	GetValue
--	---------------------------------------	-----------------

□ Syntax: [Satznummer]?
`r_variable=GetValue "<Meldungstext>"`

□ Erklärung: Anzeige eines Dialogfeldes zur Eingabe eines Wertes durch den Anwender

R20=GetValue "Bitte geben Sie die Anzahl der gewünschten Durchläufe ein"



3.1.3 FastVel-Befehl

	Eilganggeschwindigkeit in mm/sec	FASTVEL-Befehl
--	----------------------------------	----------------

□ Syntax:	[Satznummer]? [weiterer Befehl]* [Koordinaten]{0-6} FastVel geschwindigkeit [S-Befehl]? [M-Befehl]*	[Satznummer]? [weiterer Befehl]* [Koordinaten]{0-6} Fastvel geschwindigkeit [S-Befehl]? [Zusatz-Befehl]*
-----------	--	--

□ Erklärung:	<ul style="list-style-type: none"> • geschwindigkeit ist eine Dezimalzahl • mit diesem Befehl wird die Eilganggeschwindigkeit definiert • die Einheit der Eilganggeschwindigkeit ist immer mm / sec bei kartesischen Anlagen bzw. grad / sec bei Anlagen, deren erste Achse eine Rundachse ist • bei Anlagen mit Rundachsen und mindestens einer Linearachse werden die Einstellgeschwindigkeiten der Rundachse(n) der „Führungsgeschwindigkeit“ der Linearachse entsprechend vom Interpolator berechnet.
--------------	--

□ Beispiel:	; Eilganggeschwindigkeit von 100 mm/sec einstellen:
-------------	---



ISO: N10 G0 X100 Y200 Z300 **FASTVEL**
 PAL: N10 **FASTABS** X100 Y200 Z300 **FASTVEL100.0**

□ Verweis:	G0	FASTABS
------------	----	----------------

3.1.4 F-Befehl

F-Befehl	Segmentgeschwindigkeit in mm/sec	VEL-Befehl
----------	----------------------------------	------------

□ Syntax:	[Satznummer]? [weiterer Befehl]* [Koordinaten]{0-6} F geschwindigkeit [S-Befehl]? [M-Befehl]*	[Satznummer]? [weiterer Befehl]* [Koordinaten]{0-6} VEL geschwindigkeit [S-Befehl]? [Zusatz-Befehl]*
-----------	---	--

□ Erklärung:	<ul style="list-style-type: none"> • geschwindigkeit ist eine Dezimalzahl • mit diesem Befehl wird die Segmentgeschwindigkeit (Vorschub) definiert • die Einheit der Segmentgeschwindigkeit ist immer mm / sec bei kartesischen Anlagen • bei Anlagen mit Rundachsen und mindestens einer Linearachse werden die Einstellgeschwindigkeiten der Rundachse(n) der „Führungsgeschwindigkeit“ der Linearachse entsprechend vom Interpolator berechnet.
--------------	---

□ Beispiel: ; Segmentgeschwindigkeit von 100 mm/sec einstellen:



ISO: N10 G1 X100 Y200 Z300 **F100.0**
PAL: N10 MOVEABS X100 Y200 Z300 VEL100.0

□ Verweis: G1, G2, G3, G11, G12, G13 MOVEABS, CWABS, CCWABS, MOVEFRAME, CWHLXABS, CCWHLXABS

3.1.5 S-Befehl

S-Befehl	Spindeldrehzahl definieren in U/min
----------	-------------------------------------

□ Syntax:	[Satznummer]? [weiterer Befehl]* [Koordinaten]{0-6} [F-Befehl]? S drehzahl [M-Befehl]*	[Satznummer]? [weiterer Befehl]* [Koordinaten]{0-6} [F-Befehl]? S drehzahl [Zusatz-Befehl]*
-----------	--	---

□ Erklärung:

- mit diesem Befehl wird die Spindeldrehzahl eingestellt: **drehzahl** ist eine Dezimalzahl und hat die Maßeinheit U/min
- die Drehrichtung der Spindel **clw** (rechtsdrehend) wird mit dem M-Befehl M3 | **SCLW** festgelegt
- die Drehrichtung der Spindel **cclw** (linksdrehend) wird mit dem M-Befehl M4 | **SCCLW** festgelegt

□ Verweis: M3, M4, M5 | SCLW, SCCLW, SOFF

3.1.6 Werkzeugwechsel

T-Befehl T1-Befehl T2-Befehl	Werkzeugwechsel	GetTool GetTool TC1 GetTool TC2
---	------------------------	--

Syntax: [Satznummer]? [Satznummer]?

T **werkzeugnummer**
T1=werkzeugplatz
T2 =werkzeugplatz

GetTool **werkzeugnummer**
GetTool TC1 **werkzeugplatz**
GetTool TC2 **werkzeugplatz**

Erklärung:

- **werkzeugnummer** {1-128}
- **werkzeugplatz** {1-16}
- je Werkzeugwechsler (max. zwei) sind max.128 Werkzeuge für maximal 16 Werkzeugplätze konfigurierbar

Bei Angabe einer Werkzeugnummer ohne den Parameter TC1 bzw TC2 wird immer das Werkzeug mit der in der Werkzeugverwaltung definierten Nummer eingewechselt.

Bei Angabe des Werkzeugplatzes und des zusätzlichen Parameters TC1 oder TC2 zur Definition des Werkzeugwechslers wird das Werkzeug geholt, das sich auf dem angegebenen Werkzeugplatz im entsprechenden Werkzeugwechsler befindet.

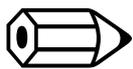
Die Werkzeugverwaltung wird bei diesem Befehl umgangen.

Beispiel:

; Werkzeug mit der in der Werkzeugverwaltung eingerichteten Werkzeugnummer 4 wird geholt

ISO: N10 T 4

PAL: N10 GetTool 4



; Werkzeug vom Werkzeugplatz 4 im Werkzeugwechsler 1 wird geholt

ISO: N10 T1=4

PAL: N10 GetTool TC1 4

3.1.7 Unterprogrammtechnik

Unterprogramm-technik: Mit Hilfe der Unterprogrammtechnik wird es dem Anwender möglich, kompakte und überschaubare Anwenderprogramme zu entwerfen und erfolgreich zu testen.

Bei Anwendung der Unterprogrammtechnik in ProNC sind folgende Randbedingungen zu beachten:

Deklarationszwang von Unterprogrammen:



Unterprogramme sind im Programmtext erst zu vereinbaren (zu deklarieren), ehe im Hauptprogramm ein Unterprogramm aufgerufen werden kann. Der Grund dafür liegt darin, dass der Compiler prüft, ob unerlaubte Unterprogrammaufrufe im Anwenderprogramm stattfinden. Ein unerlaubter Unterprogrammaufruf liegt genau dann vor, wenn das betreffende Unterprogramm noch nicht deklariert wurde.

Unterprogramme können geschachtelt werden. Es ist demnach möglich, dass in einem Unterprogramm ein anderes Unterprogramm aufgerufen wird.

maximale Schachtelungstiefe:



Die maximale Schachtelungstiefe wird mit 10 angegeben. Die Begrenzung auf 10 ist mit Sicherheit sinnvoll und resultiert lediglich daraus, dass der Stack für die Kellerung der Rückkehradressen auf eben diesen Wert limitiert wurde.

3.1.7.1 Deklaration Unterprogramm

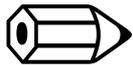
%L-Deklaration	Unterprogramm-Deklaration	%SUBR-Deklaration:
----------------	---------------------------	--------------------

□ Syntax: %L unterprogramm_nummer | %SUBR unterprogramm_nummer

□ Erklärung: **Unterprogramm-Deklaration:**

- ein Unterprogramm wird deklariert (das heißt so viel wie „vereinbart“ oder „bekannt gemacht“)
- **unterprogramm_nummer** ist eine natürliche Zahl
- vor der Kennzeichnung einer Unterprogramm-Deklaration müssen das Sonderzeichen % und der Adressbuchstabe **L** | **Schlüsselwort SUBR** stehen
- die Nummer des (deklarierten) Unterprogramms dient zur eindeutigen Kennzeichnung und darf nur einmal in einem Quellprogramm vorkommen (ein deklariertes Unterprogramm kann aber beliebig oft in einem Anwenderprogramm aufgerufen werden)
- jedes deklarierte Unterprogramm ist mit dem Befehl **M17** | **RETURN** abzuschließen

□ Beispiel:



ISO: ; das Unterprogramm mit der Nummer 11 wird deklariert
; und mit dem entsprechenden M17-Befehl abgeschlossen:
%L11

; das Bit 1 im Ausgangsbyte A1 setzen:
N10 M111

; 1 sec warten:
N20 G4 1000

; das Bit 1 im Ausgangsbyte A1 rücksetzen:
N30 M110

; das Unterprogramm beenden:
N40 **M17**

PAL: ; das Unterprogramm mit der Nummer 11 wird deklariert
; und mit dem entsprechenden **RETURN**-Befehl abgeschlossen:
%SUBR11

; das Bit 1 im Ausgangsbyte A1 setzen:
N10 SETB A1.1

; 1 sec warten:
N20 **TIME** 1000

; das Bit 1 im Ausgangsbyte A1 rücksetzen:
N30 RESB A1.1

; das Unterprogramm beenden:
N40 **RETURN**

□ Verweis: Unterprogrammaufruf: L | Unterprogrammaufruf: SUBR

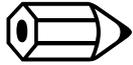
3.1.7.2 Aufruf Unterprogramm

L-Befehl	Unterprogramm-Aufruf (direkter) Unterprogramm-Aufruf (indexiert)	SUBR-Befehl
-----------------	---	--------------------

<input type="checkbox"/> Syntax:	direkter Unterprogramm-Aufruf: [Satznummer]? L unterprogramm_nummer indexierter Unterprogramm-Aufruf: [Satznummer]? L r_variable	direkter Unterprogramm-Aufruf: [Satznummer]? SUBR unterprogramm_nummer indexierter Unterprogramm-Aufruf: [Satznummer]? SUBR_r_variable
----------------------------------	---	---

- Erklärung:
- ein Unterprogramm muss vor seinem Aufruf (seiner Aktivierung) deklariert („vereinbart“) worden sein
 - die Nummer des Unterprogramms dient zur eindeutigen Kennzeichnung, ein deklariertes Unterprogramm kann an beliebig vielen Stellen im Hauptprogramm aufgerufen werden
- direkter Unterprogramm-Aufruf:**
- ein Unterprogramm wird **direkt** aufgerufen, indem nach einer optionalen Satznummer der Adressbuchstabe **L** | **Schlüsselwort SUBR**, gefolgt von einer natürlichen Zahl, im Anwenderhauptprogramm oder einem Unterprogramm programmiert wird
- indexierter Unterprogramm-Aufruf:**
- ein Unterprogramm wird **indexiert** aufgerufen, indem nach einer optionalen Satznummer der Adressbuchstabe **L** | **Schlüsselwort SUBR_**, gefolgt von einer R-Variablen, im Anwenderhauptprogramm oder einem Unterprogramm programmiert wird
 - indexierte Unterprogrammaufrufe erhöhen die Flexibilität der Programmierung erheblich, weil gerade das Unterprogramm aktiviert wird, dessen Nummer mit dem aktuellen Inhalt der entsprechenden R-Variablen übereinstimmt

□ Beispiel:



```

ISO: ; das Unterprogramm mit der Nummer 5 wird deklariert:
%L5 ; Unterprogramm zum Greifen
N10 M8 ; Greifer schließen
N20 G4 1000 ; 1 sec warten, bis geschlossen
N30 M17 ; das Unterprogramm beenden

; das Unterprogramm mit der Nummer 6 wird deklariert:
%L6 ; Unterprogramm zum Loslassen
N10 M9 ; Greifer öffnen
N20 G4 2000 ; 2 sec warten, bis geöffnet
N30 M17 ; das Unterprogramm beenden:

%100 ; Beginn des Hauptprogrammes
N10 ...
N20 ...
N30 L5 ; Unterprogramm L5 aktivieren
N40 ...
N50 L6 ; Unterprogramm L6 aktivieren
N60 ...

; Beispiel zum indexierten Unterprogrammaufruf:
; die drei Unterprogramme %L10, %L11 und %L12 werden
; deklariert:
%L10
N100 R1 = 1 R2 = 2 R3 = 3
N200 M17
%L11
N100 R1 = 5 R2 = 6 R3 = 7
N200 M17
%L12
N100 R1 = 10 R2 = 11 R3 = 12
N200 M17
;
; im Hauptprogramm wird z.B. eine Parametereingabe (über die
; Tastatur) vorgenommen:
N5 G98
; danach hat die R-Variable R1 den Wert der gewünschten
; Unterprogrammnummer:
N10 LR1 ; das Unterprogramm wird aktiviert, dessen
; Unterprogrammnummer gerade mit dem
; aktuellen Inhalt der R-Variablen R1 übereinstimmt

```

Beispiel:



```

PAL: ; das Unterprogramm mit der Nummer 5 wird deklariert:
%SUBR5 ; Unterprogramm zum Greifen
N10 GCLOSE ; Greifer schließen
N20 TIME 1000 ; 1 sec warten, bis geschlossen
N30 RETURN ; das Unterprogramm beenden

; das Unterprogramm mit der Nummer 6 wird deklariert:
%SUBR6 ; Unterprogramm zum Loslassen
N10 GOPEN ; Greifer öffnen
N20 TIME 2000 ; 2 sec warten, bis geöffnet
N30 RETURN ; das Unterprogramm beenden:

%100 ; Beginn des Hauptprogrammes
N10 ...
N20 ...
N30 SUBR5 ; Unterprogramm SUBR5 aktivieren
N40 ...
N50 SUBR6 ; Unterprogramm SUBR6 aktivieren
N60 ...

; Beispiel zum indexierten Unterprogrammaufruf:
; die drei Unterprogramme %SUBR10, %SUBR11 und %SUBR12
; werden
; deklariert:
%SUBR10
N100 R1 = 1 R2 = 2 R3 = 3
N200 RETURN
%SUBR11
N100 R1 = 5 R2 = 6 R3 = 7
N200 RETURN
%SUBR12
N100 R1 = 10 R2 = 11 R3 = 12
N200 RETURN
;
; im Hauptprogramm wird z.B. eine Parametereingabe (über die
; Tastatur vorgenommen:
N5 PARAMETER
; danach hat die R-Variable R1 den Wert der gewünschten
; Unterprogrammnummer:
N10 SUBR_R1 ; das Unterprogramm wird aktiviert, dessen
; Unterprogrammnummer gerade mit dem
; aktuellen Inhalt der R-Variablen R1 übereinstimmt
    
```

□ Verweis:

Unterprogrammdeklaration: **%L**

Unterprogrammdeklaration: **%SUBR**

3.2 Anweisungen: Syntaktische Erweiterungen zur DIN 66025

3.2.1 Variable

In ProNC sind Variablen von elementarer Bedeutung für die Möglichkeit, auf Anwenderniveau Flexibilität programmieren zu können. Die Variablen bilden die Grundlage für die Parameterrechnung.

Variable: Eine Variable muss im Programmtext *benannt* werden können. Das heißt, im Quellprogramm wird eine Variable durch einen *Namen* dargestellt.

In ProNC werden sehr einfache Namen für die verfügbaren Variablen gewählt: einem Großbuchstaben **P**, **Q** oder **R** folgt eine natürliche Zahl n :
 $0 \leq n < 100$ bei P, $0 \leq n < 500$ bei Q, $0 \leq n < 1000$ bei R.

kein Deklarationszwang für Variablen:



In ProNC müssen Variablen *nicht* explizit deklariert werden.

Dies ist deshalb nicht notwendig, weil in jedem Quellprogramm stets **ehnhundert** P-Variablen, **fünfhundert** Q-Variablen und **eintausend** R-Variablen verfügbar sind (die Variablen sind demnach implizit deklariert). Auch die sonst bei Programmiersprachen der EDV übliche Zuordnung von Datentypen zu bestimmten Variablen wird in ProNC nicht benötigt. Der Datentyp definiert immer den Wertebereich einer Variablen. In ProNC sind den verfügbaren Variablen feste Datentypen zugeordnet:

Variable in ProNC	fest zugeordneter Datentyp
P-Variablen	natürliche Zahl: Speicherumfang: 8 Bit = 1 Byte Wertebereich: 0 bis 255
R-Variablen	Gleitkommazahl: Speicherumfang: 8 Byte = 64 Bit Wertebereich: 11 Bit Exponent, 53 Bit Mantisse
Q-Variablen	Struktur von 12 Gleitkommazahlen (X, Y, Z, A, B, C für Achssystem 1, X2, Y2, Z2, A2, B2, C2 für Achssystem 2)

Tabelle 3.2.1: Variablen in ProNC und deren Wertebereiche

Zur Laufzeit eines Anwenderprogramms wird eine Variable immer durch einen **Speicherplatz** und einen **Speicherinhalt** realisiert.

Speicherplatz: *Ort:* Wo wird der aktuelle Wert der Variablen physikalisch gespeichert ?

Speicherinhalt: *Wert:* Welchen Wert repräsentiert die Variable gerade ?

Der Wert einer Variablen kann demnach zu jeder beliebigen Zeit (während der Laufzeit des Anwenderprogramms) geändert werden oder an eine andere Variable gleichen Typs übertragen werden (gilt für P- und R- Variablen).

Laufzeit des Anwenderprogramms: Dies ist genau die Zeit, während der das CNC-Zielprogramm interpretiert wird.

R-Variablen:



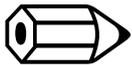
Die wichtigsten Variablen in ProNC sind die R-Variablen. Mit deren Hilfe ist es möglich, Berechnungen durchzuführen und die Ergebnisse der Berechnungen zu speichern („aufzuheben“). Um Entscheidungen zu realisieren, können R-Variablen miteinander oder mit Konstanten verglichen werden.

Eine wesentliche Eigenschaft von ProNC besteht darin, dass alle verfügbaren R-Variablen (R0 bis R999) ihre aktuellen Werte, die sie verkörpern, an Koordinaten oder F-Befehle als Parameter „übergeben“ können. Damit wird eine indirekte Wertangabe möglich.

[siehe auch:](#)
Abschnitt 3.2.2 Parameterrechnung

□ Beispiel: ; NC-Satz mit direkter Angabe von Werten:

```
ISO: N10 G1 G90 X100 Y200 Z-50  
PAL: N10 MOVEABS X100 Y200 Z-50
```



; NC-Satz mit indirekter Angabe von Werten
; mit Hilfe von R-Variablen:

```
ISO: N10 G1 G90 XR1 YR2 ZR3  
PAL: N10 MOVEABS XR1 YR2 ZR3
```

3.2.1.1 P-Variablen

P-Variablen

□ Syntax: **P variablen_index**

□ Erklärung: **Prozess-Variablen P0 bis P99:**

- **0 <= variablen_index < 100**
- Prozess-Variablen besitzen den Wertebereich 0 bis 255 (0x00 bis 0xFF)
- eine P-Variable kann mit Hilfe Bool'scher Verknüpfungen mit anderen P-Variablen oder mit Konstanten (natürliche Zahlen von **0 bis 255** oder Hexa-Dezimalzahlen von **0x00 bis 0xFF**) verknüpft werden
- die Prozessvariablen mit ungeradem Index (P1, P3, P5 bis P15) repräsentieren zu jeder Zeit die aktuellen Eingabeports E1, E2, E3 bis E8:
 $P1 = E1, P3 = E2, P5 = E3, P7 = E4$
 $P9 = E5, P11 = E6, P13 = E7, P15 = E8$

**Merke:**

P-Index für Eingabeport := E-Index * 2 - 1
mit E-Index von 1 ... 8

- die Prozessvariablen mit geradem Index (P0, P2, P4 bis P14) repräsentieren zu jeder Zeit die aktuellen Ausgabeports A1, A2, A3 bis A8:
 $P0 = A1, P2 = A2, P4 = A3, P6 = A4$
 $P8 = A5, P10 = A6, P12 = A7, P14 = A8$

Merke:

P-Index für Ausgabeport := A-Index * 2 - 2
mit A-Index von 1 ... 8

□ Beispiel: **P33** (die P-Variable mit dem Index **33**)
P66 (die P-Variable mit dem Index **66**)
P99 (die P-Variable mit dem Index **99**)



$P0=11110000B$; Ausgabeport A1 schreiben
 $P2=0xF0$; Ausgabeport A2 schreiben

□ Verweis: Abschnitt 3.2.1.3: R-Variablen
 Abschnitt 3.2.2.3 Bool'sche Ausdrücke

3.2.1.2 Q-Variablen

Q-Variablen	
--------------------	--

□ Syntax: **Q variablen_index oder QRvariablen_index**

□ Erklärung: **Q-Variablen** besitzen einen Variablenindex:

- $0 \leq \text{variablen_index} < 500$
- **Q-Variablen werden mit Frames initialisiert, jedes Frame hat einen Namen.**
- **Q-Variablen** repräsentieren eine Struktur, bestehend aus maximal 12 Gleitkommazahlen für die elemente X ... C im Achssystem 1 bzw. X2 ... C2 im Achssystem 2

● auf die Strukturelemente (die Koordinatenwerte der Achsen) kann mit folgender Konstruktion zugegriffen werden:

Q variablen_index : achs_adressbuchstabe

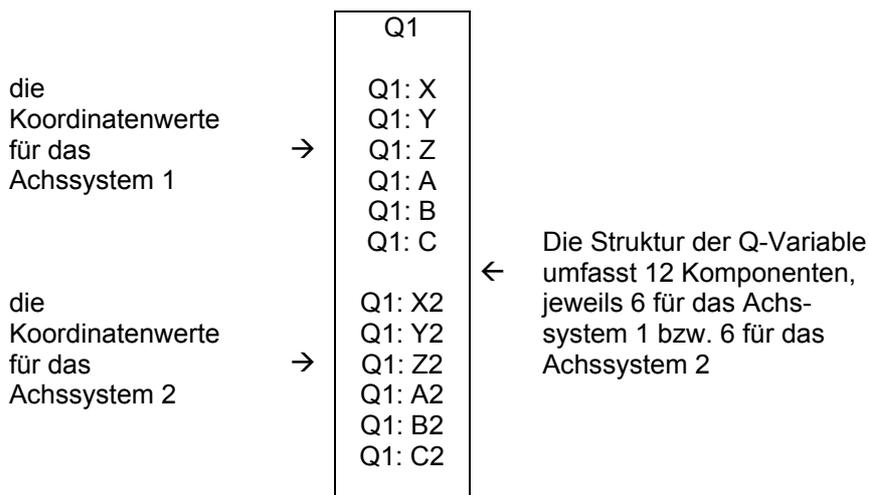


Jede Komponente einer Q-Variablen kann mit Hilfe des Achsbezeichners (Adressbuchstabe) isoliert werden:

- Q variablen_index : X** bezeichnet die **X-Koordinate im Achssystem 1**
- Q variablen_index : Y** bezeichnet die **Y-Koordinate im Achssystem 1**
- Q variablen_index : Z** bezeichnet die **Z-Koordinate im Achssystem 1**
- Q variablen_index : A** bezeichnet die **A-Koordinate im Achssystem 1**
- Q variablen_index : B** bezeichnet die **B-Koordinate im Achssystem 1**
- Q variablen_index : C** bezeichnet die **C-Koordinate im Achssystem 1**



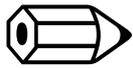
- Q variablen_index : X2** bezeichnet die **X2-Koordinate im Achssystem 2**
- Q variablen_index : Y2** bezeichnet die **Y2-Koordinate im Achssystem 2**
- Q variablen_index : Z2** bezeichnet die **Z2-Koordinate im Achssystem 2**
- Q variablen_index : A2** bezeichnet die **A2-Koordinate im Achssystem 2**
- Q variablen_index : B2** bezeichnet die **B2-Koordinate im Achssystem 2**
- Q variablen_index : C2** bezeichnet die **C2-Koordinate im Achssystem 2**



- in die Frame-Struktur können per Teach-In ermittelte Koordinatenwerte gespeichert werden
- eine Q-Variable kann mit Hilfe einer Zuweisung initialisiert werden z. B. Q1=ZIELPUNKT; ZIELPUNKT ist der Name eines Frames in der aktuellen Geometriedatei

Auf die Koordinatenwerte von Q-Variablen kann indirekt zugegriffen werden. Das heißt, der Index der gewünschten Q-Variable kann mit Hilfe einer R-Variablen angegeben werden.

□ Beispiel:



Q33 (die Q-Variable mit dem Index 33)
Q66 (die Q-Variable mit dem Index 66)
Q99 (die Q-Variable mit dem Index 99)

; der Wert der X-Koordinate innerhalb der Struktur der Q-
; Variable Q15 wird an die R-Variable R14 übertragen
; (gilt für Achssystem 1):
N100 R14 = Q15:X
N200 R15=Q15:X2; für Achssystem 2

wichtige Einzelheit:

; der Wert der Y-Koordinate innerhalb der Struktur der Q-
; Variable, deren Index dem aktuellen Wert von R88 entspricht, wird an
; die R-Variable R14 übertragen:
N200 R14 = QR88:Y

Zum obigen Beispiel folgende Erklärung:

Annahme: R88 besitzt gerade den Wert 5.

Ergebnis: R14 erhält den Wert, der in der Q-Variable Q5 an der Stelle der Y-Koordinate steht.

□ Verweis:

Abschnitt 3.2.1.3: R-Variablen
Abschnitt 3.2.2.4: Zuweisungen
Bedienungsanleitung Pkt. 2.2.2.2 Aufbau der Geometriedatei

3.2.1.3 R-Variablen

R-Variablen	
--------------------	--

Syntax: **R variablen_index oder RRvariablen_index**

Erklärung: **Real-Variablen R0 bis R999:**

● **0 <= variablen_index < 1000**

● Real-Variablen besitzen den Wertebereich einer Gleitkommazahl (doppelte Genauigkeit: 64 Bit)

● alle 1000 verfügbaren R-variablen R0 bis R999 sind in ProNC als Feld (Array) angeordnet

● eine R-Variable kann mit Hilfe arithmetischer Operatoren mit anderen R-Variablen oder mit Konstanten (Dezimalzahlen) verknüpft werden

● R-Variablen ermöglichen die Wertübertragung an eine Koordinate, einen F- oder S-Befehl



● R-Variablen können eine Komponente einer Q-Variable (X, Y, Z, A; B, C bzw. X2, Y2, Z2, A2, B2, C2) aufnehmen

● auf R-Variablen kann indirekt zugegriffen werden, indem als Index der R-Variablen eine weitere R-Variable angegeben wird z. B. RR6

● die R-Variablen R101 bis R106 repräsentieren zu jeder Zeit die aktuellen Istwerte der Achsen 1 bis 6 im Achssystem 1:

**1. Achse : R101, 2. Achse : R102, 3. Achse : R103
4. Achse : R104, 5. Achse : R105, 6. Achse : R106**

bzw. im Achssystem 2:

**1. Achse : R201, 2. Achse : R202, 3. Achse : R203
4. Achse : R204, 5. Achse : R205, 6. Achse : R206**

Bitte beachten Sie, dass in Ihrem Anwenderprogramm diese R-Variablen (R101 bis R106 bzw. R201 bis R206) nicht geschrieben werden.

Beispiel: **Indexierung von R-Variablen:**



R33 (die R-Variable mit dem Index **33**)
R66 (die R-Variable mit dem Index **66**)
R99 (die R-Variable mit dem Index **99**)

Indirekter Zugriff auf R-Variable:

N01 R1 = 1.11 R2 = 2.22 R3 = 3.33 R4 = 4.44 R5 = 5.55

...

N10 R10 = 4 ; den gewünschten Index nach R10

N20 R11 = RR10 ; R11 wird mit dem Wert 4.44 initialisiert

N30 R10 = 5

N40 R11 = RR10 ; R11 wird mit dem Wert 5.55 initialisiert



Initialisierung von R-Variablen mit Werten aus der Variablendatei:

ProNC verfügt über die Fähigkeit, die **aktuellen Werte der R-Variablen R0 bis R299** in die spezielle **Variablendatei "*.var"** zu schreiben, wenn das konkrete **Anwenderprogramm beendet** oder **abgebrochen** wird.

D. h. nach dem Abbruch des Anwenderprogramms beispiel.pal werden die aktuellen Werte der R-Variablen R0 bis R299 in die Datei beispiel.var (im Verzeichnis \CNCWorkbench\NCProg\Dest) geschrieben.

Beim **nächsten Start** werden dann diese Variablen wieder **aus der Variablendatei gelesen**, konvertiert und **indexrichtig** im Arbeitsspeicher **abgelegt**.

Wollen Sie Ihr **Anwenderprogramm** starten, **ohne** dass die Variablen R0 bis R299 **mit den Werten aus der Datei *.var initialisiert werden**, können Sie das entsprechende **Kontrollkästchen** im Dialog NC Interpreter Setup **ausschalten**:



Sie aktivieren diesen Dialog über das Menü **Einstellungen - Interpreter**.

- Verweis:
- Abschnitt 3.2.1.1: P-Variablen
 - Abschnitt 3.2.1.2: Q-Variablen
 - Abschnitt 3.2.1.4: Wertübergabe R-Variable an Koordinate
 - Abschnitt 3.2.2: Parameterrechnung

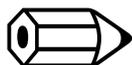
3.2.1.4 Wertübergabe R-Variable an Koordinate

Wertübergabe	
---------------------	--

Syntax: **adressbuchstabe R variablen_index**

- Erklärung:
- **Adressbuchstabe = {X, Y, Z, A, B, C}**
d.h. der Adressbuchstabe ist ein Element der angegebenen Menge
 - **0 <= variablen_index < 1000**
 - eine indirekte Wertangabe bei einem Koordinatenwort ist möglich, wenn anstelle einer Dezimalzahl eine R-Variable angegeben wird
 - der aktuelle Wert der R-Variable wird an die entsprechende Koordinate übergeben

Beispiel:



```

ISO: N10 R1 = 100
        ; der Zielpunkt hat die Koordinaten X=100mm, Y=200mm:
        N20 G1 XR1 Y200

        ; die R-Variable wird neu berechnet:
        N30 R1 = R1 + 50

        ; der Zielpunkt hat jetzt die Koordinaten X=150mm,Y=200mm:
        N40 G1 XR1 Y200

PAL: N10 R1 = 100
        ; der Zielpunkt hat die Koordinaten X=100mm, Y=200mm:
        N20 MOVEABS XR1 Y200

        ; die R-Variable wird neu berechnet:
        N30 R1 = R1 + 50

        ; der Zielpunkt hat jetzt die Koordinaten X=150mm,Y=200mm:
        N40 MOVEABS XR1 Y200
    
```

Verweis:

G0, G1, G2, G3 F-Befehl Abschnitt 3.2.2.2: Funktionen Abschnitt 3.2.2.3: Bool'sche Ausdrücke Abschnitt 3.2.2.4: Zuweisungen Abschnitt 3.2.3.3: Auswahlanweisung Abschnitt 3.2.4.1: Abfrage eines Bediendialogs	FASTABS, MOVEABS, CWABS, CCWABS
---	------------------------------------

3.2.2 Parameterrechnung

3.2.2.1 arithmetische Ausdrücke

arith_ausdruck	arithmetische Ausdrücke
-----------------------	--------------------------------

- Syntax:
1. **arith_ausdruck** **arith_operator** **arith_ausdruck** *oder*
 2. [**arith_ausdruck**] *oder*
 3. **funktion**[**arith_ausdruck**] *oder*
 4. **r_variable** *oder*
 5. **reelle_zahl**
 6. **symb_konstante**
- Erklärung:
- ein **arithmetischer Ausdruck** kann mit Hilfe von arithmetischen Operatoren mit einem zweiten arithmetischen Ausdruck verknüpft werden
 - das Ergebnis dieser Verknüpfung ist wieder ein arithmetischer Ausdruck, dessen Wert mit Hilfe einer **Zuweisung** an eine **R-Variable** übertragen werden kann.



- **arith_operator** ist ein **Zeichen** aus der Menge {+, -, *, /, **MODULO**}, damit sind die arithmetischen Operationen Addition, Subtraktion, Multiplikation, Division und Modulodivision möglich
- um den Vorrang zu bestimmen, können arithmetische Ausdrücke geklammert werden; werden in einem arithmetischen Ausdruck keine Klammern gesetzt, gilt „Punkt vor Strich“
- ein arithmetischer Ausdruck kann Argument einer Funktion sein (trigonometrische und reelle Funktionen, siehe nächster Abschnitt)
- ein arithmetischer Ausdruck kann eine R-Variable sein
- eine **reelle_zahl** ist eine Dezimalzahl
- eine **symb_konstante** kann sein **Pi = 3.142**
- eine **symb_konstante** ist eine symbolische Konstante und kann ein **Wort** sein aus der Menge {**IDOK**, **IDCANCEL**, **IDABORT**, **IDRETRY**, **IDIGNORE**, **IDYES**, **IDNO**}

Diesen symbolischen Konstanten sind folgende Werte zugeordnet:

<u>symb_konstante</u>	<u>Wert</u>
IDOK	1
IDCANCEL	2
IDABORT	3
IDRETRY	4
IDIGNORE	5
IDYES	6
IDNO	7

Beispiel: ; 1. auf der rechten Seite der Zuweisung steht ein komplexer
 ; arithmetischer Ausdruck, dessen Wert wird an die Variable
 ; R10 übertragen:
N10 R10 = PI / 4 * R11 * R11

 ; 2. Gebrauch der Klammer:
N20 R12 = [R13 + R14] * R16

 ; 3. ein arithmetischer Ausdruck als Argument der SIN-Funktion:
N30 R13 = SIN [2.0 * R14]

 ; 4. R-Variable ergibt sich aus R-Variable mit
 ; Vorzeichenumkehr: dies ist eine Zuweisung
N40 R99 = 0.0-R98

 ; 5. R-Variable ergibt sich aus einer Dezimalzahl: dies ist eine
 ; Zuweisung:
N50 R33 = 123.456

 ; 6. Modulodivision
N10 R2 = 7
N15 R3 = 3
N20 R1 = R2 MODULO R3
 ; nach der Abarbeitung der Programmzeile mit der Nummer 20
 ; hat R1 den Wert 1

Die Modulodivision ...

N10 R1 = R2 MODULO 2.0

wird gern zum Test auf Geradzahligkeit einer R-Variablen benutzt. Ist R2 gerade, ist der ganzzahlige Rest in R1 nach der Modulodivision stets 0 und kann entsprechend ausgewertet werden:

```
IF R1 == 0
  ; R2 war gerade ...
ELSE
  ; R2 war ungerade
ENDIF
```

□ Verweis: Abschnitt 3.2.2.2: Funktionen
 Abschnitt 3.2.2.3: Bool'sche Ausdrücke
 Abschnitt 3.2.2.4: Zuweisungen
 Abschnitt 3.2.3.3: Auswahlanweisung
 Abschnitt 3.2.4.1: Abfrage eines Bediendialogs

3.2.2.2 Funktionen

Funktionen	Funktionen zur Berechnung von R-Variablen
-------------------	--

- Syntax: **funktions_bezeichner [arith_ausdruck]**
- Erklärung: • eine Funktion besitzt immer ein Argument; dieses Argument wird in eckige Klammern gesetzt (bei der PAL-Syntax können auch runde Klammern benutzt werden).
- der Funktionsbezeichner definiert die konkrete Funktion näher: es werden trigonometrische und reelle Funktionen unterschieden

	Trigonometrische Funktionen
--	------------------------------------

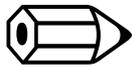
die Argumente von trigonometrischen Funktionen sind immer in Bogenmaß (rad) anzugeben:

$$\alpha \text{ (Radiant)} = \alpha \text{ (Grad)} * \text{Pi} / 180.0$$

SIN SIN (arith_ausdruck)

Operator für die trigonometrische Funktion SINUS.

- Beispiel: R11=PI/2
R12=SIN(R11) ;Ergibt 1.0
R13=SIN(2*R11) ;Ergibt 0

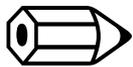


ASIN ASIN (arith_ausdruck)

Operator für die zyklometrische Funktion ARCUSSINUS. Die ASIN-Funktion ist die mathematische Umkehrung der SINUS-Funktion.

- Beispiel: Bsp.: Berechnung des Winkels zwischen Gegenkathete und Hypotenuse:

$$R11=ASIN((R22-R21)/(R32-R31))$$



COS **COS** (arith_ausdruck)

Operator für die trigonometrische Funktion COSINUS.

- Beispiel: R11=PI/2
R12=COS(R11) ;Ergibt 0
R13=COS(2*R11) ;Ergibt -1.0



ACOS **ACOS** (arith_ausdruck)

Operator für die zyklometrische Funktion ARCUSCOSINUS. Die ACOS-Funktion ist die mathematische Umkehrung der COSINUS-Funktion.

□ Beispiel: Berechnung des Winkels zwischen Ankathete und Hypothenuse:

$$R11=ACOS((R25-R24)/(R32-R31))$$



TAN **TAN** (arith_ausdruck)

ATAN Operator für die trigonometrische Funktion TANGENS.

ATAN (arith_ausdruck)

Operator für die zyklometrische Funktion ARCUSTANGENS. Die ATAN-Funktion ist die mathematische Umkehrung der TANGENS-Funktion.

□ Beispiel: N10 R1 = Pi / 4 ; ein Winkel von 45 Grad
 N20 R2 = TAN[R1] ; Ergebnis: R2 erhält den Wert 1.0
 N30 R3 = ATAN[R2] ; Ergebnis: R3 erhält den Wert 0.7855 = Pi / 4



	Reelle Funktionen
--	--------------------------

FABS FABS (arith_ausdruck)

Funktion für die Ermittlung des absoluten Wertes eines vorzeichenbehafteten numerischen Ausdrucks.

□ Beispiel: R2=FABS(R1) Ergibt den Betrag von R1
 R2=FABS(-5.0) Ergibt 5



SQRT SQRT (arith_ausdruck)

Funktion für die Ermittlung der Quadratwurzel eines numerischen Ausdrucks.

□ Beispiel: R2=SQRT(R1) Ergibt die Wurzel aus R1
 R2=SQRT(9.0) Ergibt 3

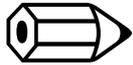


; Satz des Pythagoras:
 ; R10=3, R11=4
 R12 = SQRT [R10 * R10 + R11 * R11]; Ergebnis: R12 enthält den Wert 5.0

FLOOR FLOOR (arith_ausdruck)

Funktion für das Abrunden des übergebenen Wertes. Die Funktion gibt die nächste ganze Zahl zurück, die kleiner gleich dem angegebenen arithmetischen Ausdruck ist.

□ Beispiel: R2=FLOOR(2.5) Ergibt 2
R2=FLOOR(3.9999) Ergibt 3

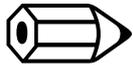


N05 R2 = 5.89
N10 R1 = FLOOR[R2]
; nach der Abarbeitung der Programmzeile mit der Nummer 10
; hat R1 den Wert 5.0

EXP EXP (arith_ausdruck)

Berechnet die Exponentialfunktion mit der Basis e.

□ Beispiel: R22=Exp(R21) ;R-Variable als Argument
R22=Exp(0) ;Ergebnis ist 1
R22=Exp(1) ;Ergebnis ist 2.718282

**LN** LN (arith_ausdruck)

Berechnet den natürlichen Logarithmus des übergebenen Arguments.

□ Beispiel: R22=Ln(R21) ;R-Variable als Argument
R22=Ln(1) ;Ergebnis ist 0
R22=Ln(2.718282) ;Ergebnis ist 1

**LOG** LOG (arith_ausdruck)

Berechnet den dekadischen Logarithmus (Basis 10) des übergebenen Arguments.

□ Beispiel: R22=Log(R21) ;R-Variable als Argument
R22=Log(1) ;Ergebnis ist 0
R22=Log(10) ;Ergebnis ist 1

**SQR** SQR (arith_ausdruck)

Funktion zum Quadrieren des übergebenen Arguments.

□ Beispiel: R22=Sqr(R21) ;R-Variable als Argument
R22=Sqr(2) ;Ergebnis ist 4

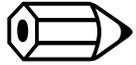


POW

Operator zum Berechnen einer Potenzfunktion. Der Operator POW wird in der folgenden Form verwendet:

Ergebnis = Basis POW Exponent

□ Beispiel:



R22=R23 POW R24 ;Mit R-Variable
R22=R23 POW 3 ;Konstante als Exponent
R22=3 POW R23 ;Konstante Basis
R3=R1 POW R2 ; Ergebnis: $R3=2^3 = 8$

□ Verweis:

Abschnitt 3.2.2.1: arithmetische Ausdrücke
Abschnitt 3.2.2.3: Bool'sche Ausdrücke
Abschnitt 3.2.2.4: Zuweisungen

3.2.2.3 Bool'sche Ausdrücke

bool_ausdruck	Bool'sche Ausdrücke
----------------------	----------------------------

- Syntax: 1. **bool_ausdruck bool_operator bool_ausdruck oder**
 2. **[bool_ausdruck] oder**
 3. **p_variable oder**
 4. **konstante**
- Erklärung: • ein **Bool'scher Ausdruck** kann mit Hilfe von Bool'schen Operatoren mit einem zweiten Bool'schen Ausdruck verknüpft werden
- das Ergebnis dieser Verknüpfung ist wieder ein Bool'scher Ausdruck, dessen Wert mit Hilfe einer **Zuweisung** an eine **P-Variable** übertragen werden kann (siehe auch Abschnitt 3.2.2.4)
- **bool_operator** ist ein **Zeichen** aus der Menge **{&, |, ^}**, damit sind die Bool'schen Verknüpfungen **UND, ODER** bzw. **EXCLUSIV ODER (ANTIVALENZ)** möglich
- um den Vorrang zu bestimmen, können Bool'sche Ausdrücke geklammert werden
- im einfachsten Fall ist ein Bool'scher Ausdruck eine P-Variable oder eine Hexa-Dezimalzahl
- eine bitweise Negation einer P-Variable ist mit der Tilde ~ möglich
- Beispiel: ; 1. auf der rechten Seite der Zuweisung steht ein komplexer
 ; Bool'scher Ausdruck, dessen Wert wird an die Variable **P20**
 ; übertragen:
 N10 **P20** = P11 & 0x11 | P12
-  ; 2. Gebrauch der Klammer:
 N20 P12 = [P13 | P14] & P16
- ; 3. P-Variable ergibt sich aus P-Variable: dies ist eine
 ; Zuweisung, bei der die P-Variable negiert wird:
 N40 P99 = ~P98
- ; 4. P-Variable ergibt sich aus Konstante: dies ist eine Zuweisung:
 N50 P33 = 0xAB ; Angabe der Konstante hexadezimal
 N51 P34 = 10101011B ; Angabe der Konstante binär
- Verweis: Abschnitt 3.2.2.1: arithmetische Ausdrücke
 Abschnitt 3.2.2.2: Funktionen
 Abschnitt 3.2.2.4: Zuweisungen

3.2.2.4 Zuweisungen

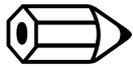
Zuweisung	Zuweisungen von Werten / Variablen an Variable
-----------	--

□ Syntax:

1. **r_variable** = arith_ausdruck *oder*
2. **r_variable** = q_variable : adressbuchstabe *oder*
3. **r_variable** = frame_name: adressbuchstabe *oder*
4. **r_variable** = p_variable *oder*
5. **r_variable** = MessageBox *oder*
6. **r_variable** = GetValue *oder*
7. **r_variable** = Posn.A
8. **r_variable** = USER *oder* USERBAT *oder* USEREXE *oder* USERDLL
9. **p_variable** = bool_ausdruck *oder*
10. **q_variable** = frame_name

- eine Zuweisung kann als Gleichung bezeichnet werden
- die beiden Seiten der Gleichung sind durch das Zeichen = miteinander verbunden
- die linke Seite der Gleichung ist immer eine Variable
- 1. einer **R-Variable** wird das Ergebnis eines arithmetischen Ausdruckes zugewiesen
- 2. einer **R-Variable** wird eine Komponente einer Q-Variable zugewiesen; diese Komponente wird durch einen Adressbuchstaben **adressbuchstaben={X,Y,Z,A,B,C}** für **Achssystem 1** bzw. **{X2,Y2,Z2,A2,B2,C2}** für **Achssystem 2** ausgewählt
- 3. einer **R-Variable** wird eine Komponente eines FRAMES zugewiesen
- 4. einer **R-Variable** wird eine P-Variable zugewiesen; damit wird es möglich, z.B mit einem Eingangsbyte zu rechnen
- 5. einer **R-Variable** wird das Ergebnis eines Bediendialoges zugewiesen
- 6. einer **R-Variable** wird ein vom Bediener einzugebender Wert auf Basis eines Bediendialogs zugewiesen
- 7. einer **R-Variable** wird der aktuelle Wert einer Achsposition zugewiesen
- 8. einer **R-Variable** wird der Returncode der aktuellen USERDLL-Funktion zugewiesen
- 9. einer **P-Variable** wird das Ergebnis eines Bool'schen Ausdruckes zugewiesen
- 10. eine **Q-Variable** wird initialisiert, indem ein durch den Namen **frame_name** ausgewähltes Element der Geometriedatei (dieses Element ist eine Datenstruktur von 12 Gleitkommazahlen) an die Q-Variable übertragen wird
- **frame_name** besteht aus maximal 20 Zeichen (Großbuchstaben oder Ziffern), wobei die ersten vier Zeichen Großbuchstaben sein müssen)

□ Beispiel:



; 1. das Ergebnis des arithmetischen Ausdruckes auf der rechten Seite der
; Zuweisung wird an die Variable **R7** übertragen:

N10 **R7** = R8 * R9 + P1 * R10

; 2. der Wert der **Y**-Komponente der Q-Variable **Q2** wird an die
; R-Variable **R22** übergeben:

N20 **R22** = **Q2:Y**

N21 **R23** = **Q2 : Y2** ; Y-Komponente aus Achssystem 2 an R23

; 3. der Wert der **Z**-Komponente des Frames **PARK_POSITION** wird an die
; R-Variable **R23** übertragen:

N25 **R23** = **PARK_POSITION: Z**

; 4. der Wert der P-Variable **P1** wird an die R-Variable **R10**
; übergeben:

N30 **R10** = **P1**

; 5. der Wert entsprechend des Ergebnisses des Bediendialogs
; wird in der R-Variablen **R30** abgelegt:

N35 **R30** = MessageBox Question YesNo "text"

; 6. die Anzahl der gewünschten Durchläufe wird vom Bediener
; eingegeben und in der Variablen **R35** abgelegt

N40 **R35** = GetValue "Bitte geben Sie die Anzahl der gewünschten
Durchläufe ein"

; 7. die aktuelle Achsposition der Achse Y im Achssystem 1 wird in **R40**
; abgelegt

N45 **R40** = POS1.Y ; für Achssystem 1

N46 R40 = POS2.Y ; für Achssystem 2

; 8. Aufruf einer User-Batch-Datei (DOS) mit drei Parametern, Returncode
; ist in **R50** abgelegt

N50 **R50** = USER [R101 R102 R103]

; 9. das Ergebnis des Bool'schen Ausdruckes auf der rechten Seite
; der Zuweisung wird an die P-Variable **P22** übertragen:

N40 **P22** = P1 & P3 & P5 & P7 & P9

; 10. die Q-Variable **Q4** wird initialisiert:

N50 **Q4** = PALETTE_1

□ Verweis:

Abschnitt 3.2.2.1: arithmetische Ausdrücke

Abschnitt 3.2.2.3: Bool'sche Ausdrücke

Abschnitt 3.2.2.4: Zuweisungen

Abschnitt 3.2.4.1: Abfrage eines Bediendialogs

Abschnitt 3.2.4.2: Aktivierung beliebiger Anwenderprogramme

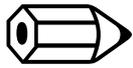
3.2.3 Anweisungen zur Steuerung des Programmablaufes

3.2.3.1 Bedingungen

Bedingungen	Bedingungen zum Testen
-------------	------------------------

- Syntax:
1. **arith_ausdruck** **vergleichts_operator** **arith_ausdruck** *oder*
 2. **bool_ausdruck** *oder*
 3. **NOT** **bool_ausdruck** *oder*
 4. **eingangs_bit** *oder*
 5. **ausgangs_bit**
- Erklärung:
- eine Bedingung ist erfüllt, wenn ein Vergleich erfüllt ist (1.) *oder* ein Bool'scher Ausdruck den Wahrheitswert TRUE besitzt (2.) *oder* ein nicht Bool'scher Ausdruck den Wahrheitswert FALSE besitzt (3.) *oder* ein Eingangs-Bit gesetzt ist (4.) *oder* ein Ausgangs-Bit gesetzt ist (5.)
 - **arith_ausdruck** ist ein *arithmetischer Ausdruck* entsprechend Syntaxvorschrift in Abschnitt 3.2.2.1
 - **bool_ausdruck** ist ein *Bool'scher Ausdruck* entsprechend Syntaxvorschrift in Abschnitt 3.2.2.3
 - **NOT bool_ausdruck** ist das Schlüsselwort **NOT** in Verbindung mit einem *Bool'schen Ausdruck*
 - **vergleichts_operator** = {<, >, !=, ==, <=, >=} ist eine der folgenden Zeichenkombinationen mit der Bedeutung:
 kleiner als: <
 größer als: >
 ungleich: !=
 gleich: ==
 kleiner gleich: <=
 größer gleich: >=
 - **eingangs_bit** ist ein Ausdruck, wie er von der SPS-Programmierung nach Anweisungsliste (AWL) bekannt ist:
E byte_nummer.bit_nummer
 - **ausgangs_bit** ist ein Ausdruck, wie er von der SPS-Programmierung nach Anweisungsliste (AWL) bekannt ist:
A byte_nummer.bit_nummer
- mit: 0 < **byte_nummer** < 9
 und: 0 < **bit_nummer** < 9

□ Beispiel:



- syntaktisch richtiger Vergleich (kleiner als):
R1 < R2

- syntaktisch richtiger Vergleich (größer als):
R3 > R4

- syntaktisch richtiger Vergleich (ungleich):
R5 != 100

- syntaktisch richtiger Vergleich (gleich):
R6 == 123.456

- syntaktisch richtiger Vergleich (größer gleich):
R7 >= 200

- syntaktisch richtiger Vergleich (kleiner gleich):
R8 <= 2 * PI

- Bool'sche Ausdrücke als Bedingung:
P1 & 0x01
P3 ^ 0x55

- syntaktisch richtiges Eingangsbit:
E1.5

- syntaktisch richtiges Ausgangsbit:
A2.8

□ Verweis:

Abschnitt 3.2.2.2: Funktionen
Abschnitt 3.2.2.3: Bool'sche Ausdrücke

3.2.3.2 Verzweigung

IF-Konstruktion	bedingte Verzweigung
------------------------	-----------------------------

Syntax: **IF bedingung
anweisungen
ELSE
anweisungen
ENDIF**

Erklärung: • Programmverzweigung:
1. für den Fall, dass die Bedingung erfüllt ist, werden die Anweisungen **anweisungen** bis zum Schlüsselwort **ELSE** ausgeführt

2. für den Fall, dass die Bedingung nicht erfüllt ist, werden die Anweisungen **anweisungen** zwischen dem Schlüsselwort **ELSE** und dem Schlüsselwort **ENDIF** ausgeführt

• sind im 2. Fall keine Anweisungen auszuführen, kann das Schlüsselwort **ELSE** entfallen

Beispiel: wenn der Wert der Variable R1 größer ist als der Wert der Variable R2, wird das Unterprogramm mit der Nummer 100 aufgerufen, im anderen Fall das Unterprogramm mit der Nummer 200:



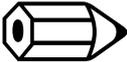
ISO: **IF R1 > R99**
 N10 L100
 ELSE
 N20 L200
 ENDIF

PAL: **IF R1 > R99**
 N10 **SUBR100**
 ELSE
 N20 **SUBR200**
 ENDIF

Verweis: Abschnitt 3.2.3.1: Bedingungen

3.2.3.3 Auswahanweisung

SWITCH-Konstruktion	Verzweigung nach Erfüllung einer Bedingung (Auswahanweisung)
----------------------------	--

- Syntax: **SWITCH** arith_ausdruck
 CASE arith_ausdruck:
 anweisungen
 ENDCASE
 CASE arith_ausdruck:
 anweisungen
 ENDCASE
 DEFAULT
 anweisungen
 ENDCASE
ENDSWITCH
- Erklärung: Die Auswahanweisung erschließt dem Anwender den Komfort höherer Programmiersprachen bezüglich des Strukturblockes "Verzweigung" und ergänzt die IF-Konstruktion effizient. Der dem Token **SWITCH** folgende **arith_ausdruck** repräsentiert zum Zeitpunkt der Programmausführung einen Wert (reelle Zahl). Dieser Wert wird mit den aktuellen Werten der folgenden **CASE**-Zweige verglichen. Bei Übereinstimmung wird der Anweisungsblock **anweisungen** ausgeführt, der Bestandteil des **CASE**-Zweiges ist, dessen Wert nach dem **CASE**-Token mit dem Wert nach dem **SWITCH**-Token übereinstimmt. Der auszuführende Anweisungsblock wird durch den Token **ENDCASE** abgeschlossen. Ist keine Übereinstimmung der Werte gegeben, wird der Block anweisungen nach dem Token **DEFAULT** ausgeführt. Der **DEFAULT**-Zweig innerhalb der **SWITCH**-Konstruktion ist optional.
- Beispiel: N1 R1=P1 ;der aktuelle Eingangsport E1 wird als
 ;Wert (0<=Wert<256) auf die Variable R1 übertragen
-  **SWITCH** R1
 CASE 1:
 R2=5 ;der Variablen R2 den Wert 5.0 zuweisen
 ENDCASE

 CASE 2:
 R2=6 ;der Variablen R2 den Wert 6.0 zuweisen
 ENDCASE

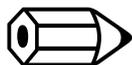
 CASE 3:
 R2=7 ;der Variablen R2 den Wert 7.0 zuweisen
 ENDCASE

 DEFAULT
 R2=10 ;der Variablen R2 den Wert 10.0 zuweisen
 ENDCASE
ENDSWITCH
- Verweis: Abschnitt 3.2.4.1: Abfrage eines Bediendialogs

3.2.3.4 Zählschleife

FOR-Schleife	Zählschleife mit einer Zählvariable
---------------------	--

- Syntax: **FOR r_variable = startwert, endwert, schrittwert**
anweisungen
ENDFOR
- Erklärung: • **r_variable** ist eine R-Variable R0 bis R999
- der R-Variable wird der Anfangswert **startwert** zugewiesen
 - so lange die R-Variable kleiner oder gleich dem Endwert **endwert** ist, werden die Anweisungen bis zum Schlüsselwort **ENDFOR** ausgeführt
 - ist die letzte Anweisung vor dem Schlüsselwort **ENDFOR** ausgeführt, wird die R-Variable um den Schrittwert **schrittwert** erhöht
 - es können maximal fünf FOR-Schleifen ineinander geschachtelt programmiert werden
 - es können maximal 62 FOR-Schleifen hintereinander in einem Quellprogramm programmiert werden
 - aus den beiden vorangehenden Punkten folgt, dass maximal $5 \times 62 = 310$ FOR-Schleifen in einem Quellprogramm programmiert werden können
- Beispiel: ; Anfang der Zählschleife
FOR R0=0,100,1
; der Koordinatenwert für die Mittelpunkts-Koordinate
N10 R8= 50.0 + R1 / 2.0
; der Koordinatenwert für die Ziel-Koordinate
N20 R7 = 100.0 + R1
; Halb-Kreis mit Wertübergabe der Variable R7 an die
; Ziel-Koordinate X und der Variable R8 an die Mittelpunkts-
; Koordinate I
ISO: N30 G2 XR7 IR8
PAL: N30 CWABS XR7 IR8
ENDFOR
- Verweis: Abschnitt 3.2.1: Variablen
Abschnitt 3.2.2: Parameterrechnung



3.2.3.5 Schleife mit Test am Anfang

WHILE-Schleife	Schleife mit Test einer Bedingung am Anfang
-----------------------	--

□ Syntax: **WHILE bedingung
 anweisungen
 ENDWHILE**

- Erklärung:
- so lange die Bedingung **bedingung** erfüllt ist, werden die Anweisungen **anweisungen** ausgeführt
 - ist die Bedingung **bedingung** beim ersten Test nicht erfüllt, wird keine Anweisung ausgeführt
 - es können maximal 5 WHILE-Schleifen ineinander geschachtelt programmiert werden
 - es können maximal 62 WHILE-Schleifen hintereinander in einem Quellprogramm programmiert werden
 - aus den beiden vorangehenden Punkten folgt, dass maximal $5 \times 62 = 310$ WHILE-Schleifen in einem Quellprogramm programmiert werden können

Merke diese Bit-Masken:



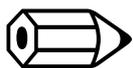
Maske zur Auswahl des Bits 1 aus einem Byte: 0x01
Maske zur Auswahl des Bits 2 aus einem Byte: 0x02
Maske zur Auswahl des Bits 3 aus einem Byte: 0x04
Maske zur Auswahl des Bits 4 aus einem Byte: 0x08

Maske zur Auswahl des Bits 5 aus einem Byte: 0x10
Maske zur Auswahl des Bits 6 aus einem Byte: 0x20
Maske zur Auswahl des Bits 7 aus einem Byte: 0x40
Maske zur Auswahl des Bits 8 aus einem Byte: 0x80

P-Variable mit Maske oder Eingangs-Bit:

P1 & 0x01 entspricht E1.1 ; der Eingang 1 im Eingabeport 1
P1 & 0x02 entspricht E1.2 ; der Eingang 2 im Eingabeport 1
P1 & 0x04 entspricht E1.3 ; der Eingang 3 im Eingabeport 1
P1 & 0x08 entspricht E1.4 ; der Eingang 4 im Eingabeport 1
P1 & 0x10 entspricht E1.5 ; der Eingang 5 im Eingabeport 1
P1 & 0x20 entspricht E1.6 ; der Eingang 6 im Eingabeport 1
P1 & 0x40 entspricht E1.7 ; der Eingang 7 im Eingabeport 1
P1 & 0x80 entspricht E1.8 ; der Eingang 8 im Eingabeport 1

□ Beispiel:



ISO: ; als Bedingung wird getestet, ob das Bit 8 (Eingang 8) im
; Eingangsbyte 2 (Eingabeport 2) gesetzt ist:
; Hinweis: E2.8 ist gleichbedeutend mit P3 & 0x80
WHILE E2.8
; Gerade zum Zielpunkt 100mm, 200mm, -300mm:
N10 G1 X100 Y200 Z-300
; 1 sec warten:
N20 TIME 1000
; Gerade zum Zielpunkt 0mm, 0mm, 0mm:
N30 G1 X0 Y0 Z0
ENDWHILE

```
; es wird so lange die leere WHILE-Schleife ausgeführt,
; wie das Bit 4 im Eingangsbyte 1 gesetzt ist:
; Synchronisation mit einem binären Eingang und Warten
; auf die High-Low-Flanke:
```

```
WHILE E1.4
```

```
; leerer Schleifenkörper
```

```
ENDWHILE
```

```
; es wird so lange die leere WHILE-Schleife ausgeführt,
; wie die Bits 4 oder 5 oder beide im Eingangsbyte 2
; gesetzt sind:
; Synchronisation mit zwei binären Eingängen und Warten,
; bis beide Eingänge E2.4 und E2.5 zu 0 geworden sind:
```

```
WHILE P3 & 0x18
```

```
; leerer Schleifenkörper
```

```
ENDWHILE
```

```
PAL: ; als Bedingung wird getestet, ob das Bit 8 im Eingangsbyte 2
; gesetzt ist:
```

```
; Hinweis: E2.8 ist gleichbedeutend mit P3 & 0x80
```

```
WHILE E2.8
```

```
; Gerade zum Zielpunkt 100mm, 200mm, -300mm:
```

```
N10 MOVEABS X100 Y200 Z-300
```

```
; 1 sec warten:
```

```
N20 TIME 1000
```

```
; Gerade zum Zielpunkt 0mm, 0mm, 0mm:
```

```
N30 MOVEABS X0 Y0 Z0
```

```
ENDWHILE
```

```
; es wird so lange die leere WHILE-Schleife ausgeführt,
; wie das Bit 4 im Eingangsbyte 1 gesetzt ist:
; Synchronisation mit einem binären Eingang und Warten
; auf die High-Low-Flanke:
```

```
WHILE E1.4
```

```
; leerer Schleifenkörper
```

```
ENDWHILE
```

```
; es wird so lange die leere WHILE-Schleife ausgeführt,
; wie die Bits 4 oder 5 oder beide im Eingangsbyte 2
; gesetzt sind:
; Synchronisation mit zwei binären Eingängen und Warten,
; bis beide Eingänge E2.4 und E2.5 zu 0 geworden sind:
```

```
WHILE P3 & 0x18
```

```
; leerer Schleifenkörper
```

```
ENDWHILE
```

- Verweis: Abschnitt 3.2.1: Variablen
 Abschnitt 3.2.2: Parameterrechnung
 Abschnitt 3.2.3.2: Verzweigung
 Abschnitt 3.2.3.6: Schleife mit Test am Ende

3.2.3.6 Schleife mit Test am Ende

DO-Schleife	Schleife mit Test einer Bedingung am Ende
--------------------	--

□ Syntax: **DO**
 anweisungen
 ENDDO bedingung

ODER

□ Syntax: **REPEAT**
 anweisungen
 UNTIL bedingung

□ Erklärung: • die Anweisungen **anweisungen** werden mindestens einmal ausgeführt
 • wenn das Schlüsselwort **ENDDO | UNTIL** erreicht ist, wird die Bedingung **bedingung** getestet
 • ist die Bedingung **bedingung** erfüllt, wird die DO-Schleife beendet und mit der Abarbeitung der nächsten Anweisung bzw. des nächsten NC-Satzes im Programm fortgefahren
 • es können maximal **fünf DO | REPEAT**-Schleifen ineinander geschachtelt programmiert werden
 • es können maximal **62 DO | REPEAT**-Schleifen hintereinander in einem Quellprogramm programmiert werden
 • aus den beiden vorangehenden Punkten folgt, dass maximal **5 x 62 = 310 DO | REPEAT**-Schleifen in einem Quellprogramm programmiert werden können

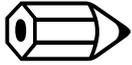
□ Beispiel:



ISO: DO
 ; Gerade zum Zielpunkt 100mm, 200mm, -300mm:
 N10 G1 X100 Y200 Z-300
 ; 1 sec warten :
 N20 G4 1000
 ; Gerade zum Zielpunkt 0mm, 0mm, 0mm:
 N30 G1 X0 Y0 Z0
 ; als Bedingung wird getestet, ob das Bit 3 (der Eingang 3)
 ; im Eingangsbyte 5 (Eingabeport 5) gesetzt ist:
 ; ist E5.3 logisch wahr, wird die DO-Schleife beendet:
ENDDO E5.3

 ; es wird so lange die leere DO-Schleife ausgeführt,
 ; wie das Bit 7 im Eingangsbyte 2 rückgesetzt ist:
 ; Synchronisation mit einem binären Eingang und Warten
 ; auf die Low-High-Flanke:
 ; Hinweis: E2.7 ist gleichbedeutend mit P3 & 0x40
DO
 ; leerer Schleifenkörper
ENDDO E2.7

 ; es wird so lange die leere DO-Schleife ausgeführt,
 ; bis eines der Bits 4 oder 5 oder beide im Eingangsbyte 2
 ; gesetzt werden:
 ; Synchronisation mit **zwei** binären Eingängen und Warten,
 ; bis **einer der beiden** Eingänge E2.4 oder E2.5 zu 1 geworden ist:



```

DO
; leerer Schleifenkörper
ENDDO P3 & 0x18

PAL: DO
; Gerade zum Zielpunkt 100mm, 200mm, -300mm:
N10 MOVEABS X100 Y200 Z-300
; 1 sec warten :
N20 TIME 1000
; Gerade zum Zielpunkt 0mm, 0mm, 0mm:
N30 MOVEABS X0 Y0 Z0
; als Bedingung wird getestet, ob das Bit 3 im Eingangsbyte 5
; gesetzt ist:
; ist E5.3 logisch wahr, wird die DO-Schleife beendet:
ENDDO E5.3

; es wird so lange die leere DO-Schleife ausgeführt,
; wie das Bit 7 im Eingangsbyte 2 rückgesetzt ist:
; Synchronisation mit einem binären Eingang und Warten
; auf die Low-High-Flanke:
; Hinweis: E2.7 ist gleichbedeutend mit P3 & 0x40
DO
; leerer Schleifenkörper
ENDDO E2.7

; es wird so lange die leere DO-Schleife ausgeführt,
; bis eines der Bits 4 oder 5 oder beide im Eingangsbyte 2
; gesetzt werden:
; Synchronisation mit zwei binären Eingängen und Warten,
; bis einer der beiden Eingänge E2.4 oder E2.5 zu 1 geworden ist:
DO
; leerer Schleifenkörper
ENDDO P3 & 0x18
    
```

- Verweis:
- Abschnitt 3.2.1: Variablen
 - Abschnitt 3.2.2: Parameterrechnung
 - Abschnitt 3.2.3.2: Verzweigung
 - Abschnitt 3.2.3.5: Schleife mit Test am Anfang

3.2.4 Anweisungen zur Kommunikation mit externen Geräten

3.2.4.1 Abfrage eines Bediendialogs

MessageBox	Abfrage eines Bediendialogs
-------------------	------------------------------------

- Syntax: **r_variable = MessageBox** [icon] [button] "text"
 [icon] {**INFO** oder **WARNING** oder **QUESTION** oder **ERROR**}
 [button] {**OK** oder **OKCANCEL** oder **YESNO** oder **YESNOCANCEL** oder
RETRYCANCEL oder **ABORTRETRYIGNORE**}
- Erklärung:
- es wird eine Dialogbox mit einer Meldung auf dem Bildschirm ausgegeben
 - das Programm wird solange unterbrochen, bis eine Bediener eingabe erfolgt
 - in Abhängigkeit von der Entscheidung des Bedieners wird in der **R-Variablen** der Rückgabeparameter abgelegt
 - der Anwender kann Symbol, Schaltfläche und Fehlertext im Rahmen der Vorgaben selbst bestimmen
 - umfasst der auszugebende "text" in der Dialogbox mehrere Zeilen muss für jeden Zeilenumbruch ein \n geschrieben werden
 - eine Auswertung der R-Variable kann mit der **SWITCH**-Anweisung erfolgen
- Beispiel:
- ; nach Entscheidung des Bedieners zur Fortsetzung (YES)
 ; oder Beendigung (NO) des Vorgangs wird in
 ; R1 der Rückkehrcode IDYES oder IDNO abgelegt
-  N10 R1 = **MessageBox** ERROR YESNO "Temperaturgrenze erreicht:
 \n\n\n Fortsetzung ?"
 ;
 ; Auswertung des Rückkehrcodes
 SWITCH R1
 Case IDYES:
 Type MessageBox mit JA beendet . . .
 EndCase
 Case IDNO:
 Type MessageBox mit Nein beendet . . .
 EndCase
 ENDSWITCH
- Verweis: Abschnitt 3.2.2.1: arithmetische Ausdrücke
 Abschnitt 3.2.2.4: Zuweisungen
 Abschnitt 3.2.3.3: Auswahlanweisung

3.2.4.2 Aktivierung beliebiger Anwenderprogramme

USER USERBAT USEREXE USERDLL	Aktivierung von Anwenderprogrammen in einem DOS-BATCH-File, als Windows-EXE oder als Funktion einer Windows-DLL
---	--

- Syntax:
- Rx = USER[Ra Rb Rc ... Ri]**
- x, a ... i für beliebige Indizes der R-Variablen oder Achspositionen
 - es können maximal 9 Parameter (Ra bis Ri) übergeben werden
 - x kann den Wert 0 ... 999 annehmen
- Rx = USEREXE name_exe_file**
Rx = USERDLL name_dll_file name_dll_function

- Erklärung:
- In ProNC ist es möglich, zur Programmlaufzeit des ISO/PAL-Anwenderprogramms kundenspezifische Anwenderprogramme (DOS-BATCH-Files, EXE-Files oder DLL-Funktionen) zu aktivieren. Zu diesem Zweck müssen im Verzeichnis \CNCWorkbench\BIN die aufzurufenden Dateien user.bat, name.exe, name.dll installiert werden. Diese Dateien kann der Anwender entsprechend seiner konkreten Aufgabenstellung beliebig erstellen bzw. vorhandene nutzen.

Die Angabe der Bat-, Exe- oder Dll-Datei im Anwenderprogramm erfolgt nur über den Name der Datei ohne Extension.

Hinweis:

Mit der Aktivierung von Batch-Dateien ist es in ProNC sehr einfach möglich, Protokollier- und/oder Druckfunktionen in das laufende Anwenderprogramm zu integrieren.

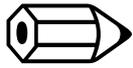
1. R101 bis R106 bzw. POS1.X bis POS1.C sind die aktuellen Istwerte der Achsen 1...6 im Achssystem 1
2. R201 bis R206 bzw. POS2.X bis POS2.C sind die aktuellen Istwerte der Achsen 1...6 im Achssystem 2



Die Anwendungsmöglichkeiten für die Fähigkeit, eigene User-Dateien im ISO/PAL- Anwenderprogramm zu aktivieren, sind u.a.:

- Erstellung von Messdateien und Protokollen
- Drucken beliebiger Programmdateien
- Parametrierung von externen Geräten wie Laser- oder Schweißsteuerungen, z.B. über serielle Schnittstelle
- Kommunikation mit autarken Steuerungen
- Bedienung von OEM-Hardware (z.B. DAC) im Steuer-PC
- Realisierung einer Ferndiagnose (Modem oder Ethernet)
- Integration anwenderspezifischer Dialoge bzw. Visualisierung

□ Beispiele:



①; Aktivierung der User-Batch-Datei user.bat
; mit Übergabe der drei Parameter (Istwerte der Achsen X, Y und Z):
N100 R1 = UserBat[POS1.X POS1Y POS1Z]

Die Batch-Datei user.bat hat z. B. folgenden Inhalt:

```
@echo off  
echo Protokollieren von Istwerten>prn  
echo X=%1 Y=%2 Z=%3>prn
```

②; Aktivierung der User-Exe Wordpad
N200 R0=UserExe Wordpad

③; es soll die Funktion funktion_name
; der von einem Anwender erstellten DLL
; dll_name gestartet werden
; Diese Anwender-DLL muss im BIN-Verzeichnis
; der aktuellen ProNC-Installation
; (z.B. C:\CNCWorkbench\Bin) vorhanden sein
; Die Anwender_DLL muss mindestens die Funktion
; funktion_name bereitstellen

; Aktivierung einer User-Dll
N300 R2=UserDll dll_name funktion_name

□ Hinweis: *Hinweis für C-Programmierer zur Erstellung der/einer Anwender-DLL:*



```
// defines:
#define ERR_NOERROR      (DWORD)0
#define USER_DLLFUNC    __declspec(dllexport) _stdcall

// types:
typedef struct r_var
{
    int nInIt;
    double dValue;
} typ_r_var;

// prototyping:
DWORD USER_DLLFUNC FUNKTION_NAME(LPVOID parameter1,
LPVOID parameter2);

// function body:
DWORD USER_DLLFUNC FUNKTION_NAME(LPVOID parameter1,
LPVOID parameter2)
//-----
// Funktionscode für eine Anwender Dll Funktion ...
// Ein:
// parameter1: Zeiger auf den Namen der aktuellen Variablen-Datei *.var
// parameter2: Zeiger auf die R-Variablen
// Aus:
// Rückgabeparameter
//-----
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());

    DWORD dwRetc;
    int nResponse;
    typ_r_var MeineAktuelleRVariableR0;
    typ_r_var *pR_Variable = (typ_r_var *)parameter2;

    // Ihr Anwender Source Code in C / C++ ...

    // Hinweis:
    // Sie haben Zugriff auf alle 1000 R-Variablen, die in ProNC für die
    //Anwenderprogrammierung verfügbar sind.

    // Die R-Variable R0 wird wie folgt gelesen:
    MeineAktuelleRVariableR0.dValue = pR_Variable[0].dValue;

    // Die R-Variable R0 wird geschrieben mit dem Wert 123.456:
    pR_Variable[0].dValue = 123.456;

    return(ERR_NOERROR);
}
```

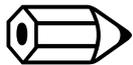
□ Hinweis: Wenn Sie Ihre Anwender-DLL mit einer PASCAL-Programmierungsumgebung oder mit LabView® erstellen wollen, helfen wir Ihnen gern. Bitte setzen Sie sich mit unserem Kundensupport tech-support@isel.com Verkauf / Service, Verkauf / Technische Beratung in Verbindung.



4 Synchronisation auf das Bewegungsende, Integration des Teach-In

4.1 Synchronisation auf das Bewegungsende

Programmierung
eines Bewegungs-
befehles als
"Bewegungsanstoß":



Mit dem Softwaresystem *ProNC* ist es möglich, eine **Parallelität von Bewegung und Programminterpretation zu programmieren**. Zu diesem Zweck ist es erforderlich, syntaktisch einen Bewegungsanstoß im Anwenderprogramm zu formulieren. Der „**Bewegungsanstoß**“ wird mit einer **dem Bewegungsbefehl vorangestellten Raute #** programmiert. Dieser **Bewegungsanstoß bewirkt**, dass der so gekennzeichnete Bewegungsbefehl an die Bewegungssteuerung übergeben wird, die Bewegung gestartet wird, aber das **Ende der Bewegung nicht** wie sonst üblich **abgewartet wird**:

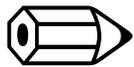
```
N10 #MOVEABS X100 Y200 ; "Bewegungsanstoß"
While (InMotion) ; Synchronisation auf Bewegungsende
  If (POS1.X > 40) and (POS1.X < 60)
    SetBit A1.1=1
  Else
    SetBit A1.1=0
  EndIf
EndWhile
; Programmfortsetzung, wenn Zielposition X=100, Y=200 erreicht ist
```

Die Synchronisation auf das Bewegungsende wird innerhalb der dem Satz N10 ... folgenden Schleife (z.B. While-Schleife) realisiert. Die spezielle Bedingung **InMotion** ist so lange wahr, wie die mit dem Satz N10 „angestoßene“ Bewegung zum Zielpunkt [**X100, Y200**] anhält. **Während** dieser **Bewegung** können z.B. die **Achs-Istwerte** (POS1.X für X, POS1.Y für Y, POS1.Z für Z, ... POS1.C für C) **überwacht** werden, um **bestimmte Ausgänge zu setzen / rückzusetzen**.

Während einer Achsbewegung kann die Bewegung beeinflusst werden.

Dazu das folgende Beispiel:

□ Beispiel:



```

MoveAbs X50.0           ; Startposition in der X-Achse
#MoveRel X1000.0 F5.0   ; Bewegungsanstoß
While (InMotion)
  IF E1.1
    MOverride 50         ; Override auf 50%
    SetPort A1=11110000B
  EndIf
  If E1.2
    MOverride 140       ; Override auf 140%
    SetPort A1=00001111B
  EndIf
  If E1.3
    MStop               ; Bewegung anhalten
  EndIf
  If E1.4
    MStart              ; Bewegung fortsetzen
  EndIf
  If E1.5
    MAbort              ; Bewegung abbrechen
  EndIf
EndWhile
    
```

Die folgenden Befehle zur Bewegungsbeeinflussung sind verfügbar:

```

MOverride wert      ; Bewegungsoverride auf wert
                          ; setzen (0 ≤ wert ≤ 140)
MStop              ; Bewegung anhalten
MStart             ; Bewegung fortsetzen
MAbort            ; Bewegung abbrechen
    
```

□ Hinweis:

Die o. g. Befehle sind für ISO und PAL gleich.

4.2 Integration des Teach-In

Integration des Teach In:

Die Integration des *Teach-In* auf der Ebene des Anwenderprogramms (ISO- oder PAL-Anwenderprogramm) erfolgt durch:

- die Programmierung des Befehles Teach
- die Nutzung von Frame-Variablen (Q-Variablen)
- die Angabe von Frame-Namen zur Definition der Bewegungsziele bei den Bewegungsbefehlen
 - FASTFRAME
 - MOVEFRAME

Frame-Variable initialisieren:

Beispiel:
Q1 = START_FRAESEN



Erläuterung:

In der zum aktuellen Anwenderprogramm **name.cnc** gehörenden Geometriedatei **name.fra** wird eine Frame-Struktur mit dem Namen **START_FRAESEN** gesucht. Ist eine solche Struktur vorhanden, werden die darin gespeicherten Werte (X-Koordinate, Y-Koordinate, Z-Koordinate,

A-Koordinate, B-Koordinate, C-Koordinate) an die Q-Variable **Q1** übertragen. Der Zweck dieser Konstruktion (Initialisierung der Q-Variablen) besteht darin, dass auf die Geometrieinformationen (Koordinatenwerte) innerhalb der Q-Variablen viel schneller zugegriffen werden kann als auf eine benannte Struktur innerhalb der Geometriedatei.

[siehe auch Bedienungsanleitung:](#)

2.2.2 Die Geometriedatei

Zugriff auf Geometrieinformationen in Frame-Variablen:



Auf die nach der Initialisierung der Frame-Variable (Q-Variable) in dieser Variable gespeicherten Geometrieinformationen (Koordinatenwerte aller Achsen) kann wie folgt zugegriffen werden:

Isolation einer bestimmten Komponente der Frame-Variablen:

Beispiel:

R1 = Q1:X

Erläuterung:

Der Wert der **X**-Koordinate innerhalb der Geometrievariable **Q1** wird an die R-Variable **R1** übertragen.

Beispiel:

R2 = Q1:Y

Erläuterung:

Der Wert der **Y**-Koordinate innerhalb der Geometrievariable **Q1** wird an die R-Variable **R2** übertragen.

Beispiel:

R3 = Q1:Z

Erläuterung:

Der Wert der **Z**-Koordinate innerhalb der Geometrievariable **Q1** wird an die R-Variable **R3** übertragen.

Beispiel:

R4 = Q1:A

Erläuterung:

Der Wert der **A**-Koordinate innerhalb der Geometrievariable **Q1** wird an die R-Variable **R4** übertragen.

Beispiel:

R5 = Q1:B

Erläuterung:

Der Wert der **B**-Koordinate innerhalb der Geometrievariable **Q1** wird an die R-Variable **R5** übertragen.

Beispiel:

R6 = Q1:C

Erläuterung:

Der Wert der **C**-Koordinate innerhalb der Geometrievariable **Q1** wird an die R-Variable **R6** übertragen.

Zielvorgabe mit Hilfe von Geometrieinformationen in Frame-Variablen oder direkte Angabe des Frame-Namen:

Auf die nach der Initialisierung der Frame-Variablen in dieser Variablen gespeicherten Geometrieinformationen (Koordinatenwerte aller Achsen) kann wie folgt zugegriffen werden:

□ Beispiel: ; Benennung einer Q-Variable als Zielvorgabe eines Bewegungsbefehles:

ISO: N100 G11 **Q1**
PAL: N100 **MOVEFRAME Q1**

Erläuterung:

Es wird eine Absolut-Bewegung **MOVEFRAME** aller Achsen zu den innerhalb der Q-Variable enthaltenen Koordinatenwerten durchgeführt. Das heißt:

Der Wert der X-Koordinate innerhalb der Q-Variable **Q1** bestimmt die Zielposition der X-Achse.

Der Wert der Y-Koordinate innerhalb der Q-Variable **Q1** bestimmt die Zielposition der Y-Achse.

Der Wert der Z-Koordinate innerhalb der Q-Variable **Q1** bestimmt die Zielposition der Z-Achse.

Der Wert der A-Koordinate innerhalb der Q-Variable **Q1** bestimmt die Zielposition der A-Achse u.s.w.

Alternativ zur Angabe einer Q-Variablen im G10 | **FASTFRAME** bzw. G11 | **MOVEFRAME**-Befehl kann der Name des gewünschten Frames direkt angegeben werden:

Beispiel:

ISO: N100 G11 **START_FRÄSEN**
PAL: N100 **MOVEFRAME START_FRÄSEN**

Erläuterung:

Es wird eine Absolut-Bewegung G11 | **MOVEFRAME** aller Achsen zu den innerhalb des Frames mit dem Namen **START_FRAESEN** enthaltenen Koordinatenwerten durchgeführt.

Das heißt:

Der Wert der X-Koordinate innerhalb des Frames mit dem Namen **START_FRAESEN** bestimmt die Zielposition der X-Achse.

Der Wert der Y-Koordinate innerhalb des Frames mit dem Namen **START_FRAESEN** bestimmt die Zielposition der Y-Achse.

Der Wert der Z-Koordinate innerhalb des Frames mit dem Namen **START_FRAESEN** bestimmt die Zielposition der Z-Achse.

(dementsprechend auch für die Achsen A, B und C)

Geometriedatei - FRAME-Struktur:

Die Initialisierung einer Q-Variable setzt voraus, dass Informationen in einer Geometriedatei gespeichert wurden, die während des Teach-In aktuelle Positionen / Orientierung(en) einer Anlage darstellen. Die Koordinaten werden dabei innerhalb einer festen Struktur, der FRAME-Struktur, abgelegt. Eine Geometriedatei kann beliebig viele FRAME-Strukturen aufnehmen. Jede dieser Strukturen erhält einen Namen (bestehend aus maximal 20 Zeichen). Innerhalb einer Geometriedatei darf ein Name immer nur einmal vorkommen.

4.3 Beispiel für ein einfaches Anwenderprogramm mit Integration des Teach-In

Nachfolgend wird das Anwenderprogramm (in PAL-Syntax) **Mep_p.pal** (Mein erstes Programm) aufgeführt. Dieses einfache Anwenderprogramm demonstriert selbstdokumentierend die Integration des Teach-In auf Anwenderprogrammiersniveau:

Mep_p.pal

Das hier beschriebene PAL-Quellprogramm **Mep_p.pal** wird nach der Installation im Verzeichnis



\\CNCWorkbench\NCProg\PAL\Sample

bereitgestellt.

Das analoge ISO-Quellprogramm finden Sie im Verzeichnis
\\CNCWorkbench\NCProg\ISO\Sample als Mep_p.iso.

```

;=====
;
; Anwenderprogramm für ProNC:
;
; Mep_p.pal: Mein erstes Programm (PAL-Syntax)
;
;=====
;
; keine Unterprogramm-Deklaration benötigt ...
;
;-----
; Beginn des Hauptprogrammes:
ProgBegin
N10 Ref XYZ ; Referenzfahrt in allen Achsen ausführen
N20 Type die geteachten Positionen können noch korrigiert werden ...
N30 Teach
;
; FOR-Schleife: 200 Durchläufe
For R0=1,200,1
;
; Parkposition anfahren mit Eilganggeschwindigkeit:
N100 FastFrame PARK_POSITION
; Arbeitsspindel 1 einschalten (rechtsdrehend), 8000 U/min:
N110 Sclw 1 S1=8000
; Start-Position zum Fräsen im Eilgang anfahren (Spindel läuft hoch):
N120 FastFrame START_FRAESEN
; Fräsbearbeitung mit Segmentgeschwindigkeit von 0.12 m/min:
N130 MoveFrame ENDE_FRAESEN F0.12
; Arbeitsspindel ausschalten:
N140 Soff
; 1 sec = 1000 msec warten in der aktuellen Position:
N150 Time 1000
; testen, ob nach 100 Zyklen eine Referenzfahrt auszuführen ist:
If R0 == 100
N200 Type Referenzfahrt in allen Achsen ...
N210 REF
; Ende der IF-Konstruktion
EndIf
; Ende der Zählschleife
EndFor
;-----
; Programmende
ProgEnd
;=====

```

5 Ausgewählte Lösungen mit ProNC

5.1 isel-XYZ-Anlagen / beliebige kartesische Kinematiken

5.1.1 Lernen

Das folgende Anwenderprogramm (ISO: Lernen.iso / PAL: Lernen.pal) ist gut geeignet, an einer XYZ-Portalanlage wichtige Basisbefehle von **ProNC** wie Geraden- und Kreisinterpolation, Wegbedingungen wie Absolutmaß und Relativmaß (Kettenmaß), Interpolationsebene für die Kreisinterpolation, Wartezeit, Unterprogrammtechnik u.a. kennen zu lernen.

Werkstücknullpunkt per Teach-In: Eine erfolgreiche Abarbeitung des Programms **Lernen.iso / Lernen.pal** setzt voraus, dass der Werkstücknullpunkt per Teach-In in die zugehörige Geometriedatei „Lernen.fra“ übernommen wird.

WS_NULLPUNKT = Werkstücknullpunkt: Die Geometriedatei **Lernen.fra** enthält nur die eine Frame-Struktur mit dem Namen **WS_NULLPUNKT: Werkstücknullpunkt**

Lernen.iso Das entsprechende ISO-Quellprogramm **Lernen.iso** ist nach der Installation im Verzeichnis



\CNCWorkbench\NCProg\ISO\Sample
enthalten.

Lernen.pal Das entsprechende PAL-Quellprogramm **Lernen.pal** ist nach der Installation im Verzeichnis



\CNCWorkbench\NCProg\PAL\Sample
enthalten.

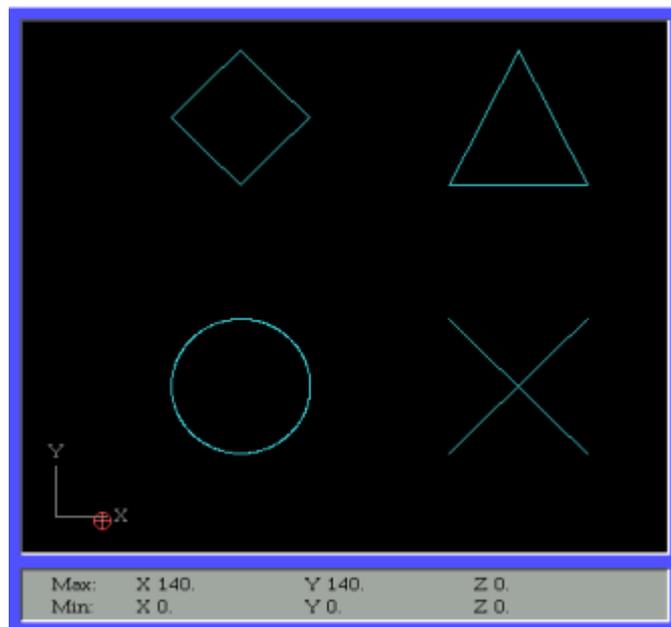
5.1.2 Figuren

Bei der Abarbeitung des folgenden Programmes

ISO: Figures.iso
PAL: Figures.pal

auf einer **isel-XYZ-Anlage** ergibt sich die folgende Fräskontur (bezogen auf die XY-Zeichenebene):

Oben links: Quadrat, auf einer Ecke stehend
Oben rechts: gleichseitiges Dreieck
Unten links: Vollkreis
Unten rechts: X (Höhe = Kreisdurchmesser)



Figures.iso



Das entsprechende ISO-Quellprogramm **Figures.iso** ist nach der Installation im Verzeichnis

\CNCWorkbench\NCProg\ISO\Sample

enthalten.

Figures.pal



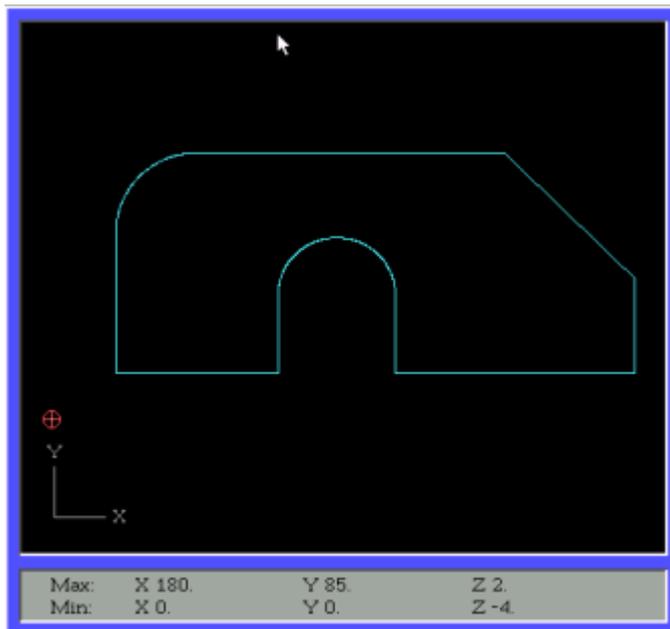
Das entsprechende PAL-Quellprogramm **Figures.pal** ist nach der Installation im Verzeichnis

\CNCWorkbench\NCProg\PAL\Sample

enthalten.

5.1.3 Fräsen einer einfachen Kontur

Bei der Abarbeitung des folgenden Programms auf einer *iseI*-XYZ-Anlage ergibt sich die folgende Fräskontur:



Kontur.iso



Das entsprechende ISO-Quellprogramm **Kontur.iso** ist nach der Installation im Verzeichnis

`\CNCWorkbench\NCProg\ISO\Sample`

enthalten.

Kontur.pal



Das entsprechende PAL-Quellprogramm **Kontur.pal** ist nach der Installation im Verzeichnis

`\CNCWorkbench\NCProg\PAL\Sample`

enthalten.

5.1.4 Bohren

Im folgenden Bohrprogramm werden die Anwendung von verschachtelten FOR-Schleifen sowie die Unterprogrammtechnik demonstriert.

Anwendung von FOR-Schleifen:

Als Schleifenzähler sind R-Variablen zu benutzen. Der Schleifenzähler **R10** kann seinen aktuellen Wert an die X-Koordinate als Zielwert übergeben:

```
For R10=10,100,10 ; Zählschleife für die X-Koordinate
;
; Anfahren der X-Koordinate abhängig vom Wert R10
N40 FASTABS XR10
...
```

Nullpunkt vermessen:

Für ein einwandfreies Funktionieren des Programms auf Ihrer Anlage ist der Nullpunkt des zu bohrenden Werkstückes zu vermessen und im nachstehenden Satz zu aktualisieren:

N20 **FASTABS** X80 Y70 Z-30 ; Anfahren des neuen
Werkstücknullpunktes

Effektiver ist sicherlich ein Vorgehen entsprechend des Programms „Lernen.pal“ im vorhergehenden Abschnitt. Dort wurde der Werkstücknullpunkt „WS_NULLPUNKT“ geteacht und zur Laufzeit des CNC-Zielprogrammes in die Q-Variabale **Q1** übernommen. Nachdem der geteachte Punkt mit dem NC-Satz **MOVEFRAME Q1** angefahren wurde, kann mit dem WPZERO-Befehl ein neuer Nullpunkt eingerichtet werden.

Sie sollten die o.g. Programmänderung in Ihrem Quellprogramm „**Bohren.iso**“ / „**Bohren.pal**“ vornehmen und neu übersetzen. Ist der Nullpunkt per Teach-In in die neu anzulegende Geometriedatei „Bohren.fra“ übernommen, kann das CNC-Zielprogramm „Bohren.cnc“ abgearbeitet werden:

Bohren.iso



Das entsprechende ISO-Quellprogramm **Bohren.iso** ist nach der Installation im Verzeichnis

\CNCWorkbench\NCProg\ISO\Sample

enthalten.

Bohren.pal



Das entsprechende PAL-Quellprogramm **Bohren.pal** ist nach der Installation im Verzeichnis

\CNCWorkbench\NCProg\PAL\Sample

enthalten.

5.1.5 Taschen fräsen

Das folgende Beispielprogramm zum Taschenfräsen basiert auf der Anwendung des Variablenkonzeptes und der Parameterrechnung mit R-Variablen. Es werden die Anwendung von verschachtelten FOR-Schleifen sowie die Unterprogrammtechnik demonstriert.

Taschen.iso



Das zugehörige ISO-Quellprogramm **Taschen.iso** ist nach der Installation im Verzeichnis

\CNCWorkbench\NCProg\ISO\Sample

enthalten.

Taschen.pal



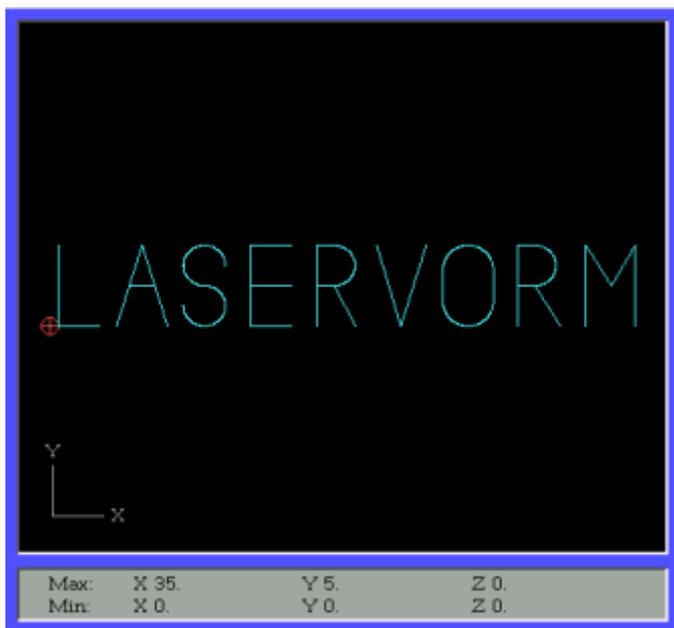
Das zugehörige PAL-Quellprogramm **Taschen.pal** ist nach der Installation im Verzeichnis

\CNCWorkbench\NCProg\PAL\Sample

enthalten.

5.1.6 Schrift mit Laser gravieren

Bei der Abarbeitung des folgenden Programms auf einer *isel*-XYZ-Anlage ergibt sich die folgende Schrift:



Schrift.iso

Das zugehörige ISO-Quellprogramm **Schrift.iso** ist nach der Installation im Verzeichnis



`\CNCWorkbench\NCProg\ISO\Sample`

enthalten.

Schrift.pal

Das zugehörige PAL-Quellprogramm **Schrift.pal** ist nach der Installation im Verzeichnis



`\CNCWorkbench\NCProg\PAL\Sample`

enthalten.

5.1.7 Schweißen

Das folgende Programm realisiert ein automatisiertes Schweißen eines Karosserierahmens. Es werden vier Nähte geschweißt.

Teach-In für Schweiß-Nähte:

Es werden für die vier Schweißnähte acht geteachte Punkte benötigt:



```

N05 Q1 = NAHT_PUNKT1_ANFANG      ; Naht 1 - Startpunkt
N10 Q2 = NAHT_PUNKT1_ENDE        ; Naht 1 - Endpunkt
N15 Q3 = NAHT_PUNKT2_ANFANG      ; Naht 2 - Startpunkt
N20 Q4 = NAHT_PUNKT2_ENDE        ; Naht 2 - Endpunkt
N25 Q5 = NAHT_PUNKT3_ANFANG      ; Naht 3 - Startpunkt
N30 Q6 = NAHT_PUNKT3_ENDE        ; Naht 3 - Endpunkt
N35 Q7 = NAHT_PUNKT4_ANFANG      ; Naht 4 - Startpunkt
N40 Q8 = NAHT_PUNKT4_ENDE        ; Naht 4 - Endpunkt
    
```

Für den Übergänge vom Endpunkt der Naht 1 zum Startpunkt der Naht 2 wird ein Zwischenpunkt benötigt:

N45 Q12= EINS_NACH_ZWEI ; Übergang Naht 1 nach Naht 2

Für den Übergänge vom Endpunkt der Naht 3 zum Startpunkt der Naht 4 wird ein Zwischenpunkt benötigt:

N50 Q34= DREI_NACH_VIER ; Übergang Naht 3 nach Naht 4

Migmag.iso



Das zugehörige ISO-Quellprogramm **Migmag.iso** ist nach der Installation im Verzeichnis

\CNCWorkbench\NCProg\ISO\Sample

enthalten.

Migmag.pal



Das zugehörige PAL-Quellprogramm **Migmag.pal** ist nach der Installation im Verzeichnis

\CNCWorkbench\NCProg\PAL\Sample

enthalten.

6 Zusammenfassung

6 Zusammenfassung

- ProNC:** Das Softwarepaket **ProNC** ist ein Bedien- und Programmiersystem für CNC-Anlagen / CNC-Systeme für Bearbeitungs- und Handling-Applikationen. **ProNC** integriert eine maussensitive Bedienoberfläche entsprechend des SAA-Standards und eine Programmierplattform zur Erstellung, zur Inbetriebnahme und zum Test von ISO- bzw. PAL-Anwenderprogrammen.
- Bediendialog:** Der Bediendialog wird durch Pull-Down-Menüs mit einem funktionell übersichtlichen Bedienbaum realisiert. Klartext-Fehlermeldungen und -informationen sowie die unmittelbare kontextbezogene Verzweigung in die Online-Hilfe unterstützen die Einarbeitung in das Programmsystem effektiv. Bedien- und Laufzeitfehler werden als Klartext im Windows-Stil mitgeteilt.
- isel-Steuerungssoftware:** **ProNC** ist auf PCs unter den Betriebssystemen Win98/Win2000/WinNT4.x/WinXP lauffähig. Die Software wurde als Weiterentwicklung der Systeme **Remote / Pro-DIN bzw. Pro-PAL** implementiert. Dabei wurde die technologieorientierte Syntax der DIN 66025 bzw. des isel-Zwischenformates um problemorientierte Konstruktionen zur Strukturierten Programmierung, zur Parameterrechnung sowie zum Zugriff auf Geometriedateien ergänzt und als flexible, leistungsfähige Programmiersprache definiert (ISO- oder PAL-Syntax).
- ISO- / PAL-Compiler:** Die Möglichkeit der geschachtelten Nutzung von Konstruktionen zur Steuerung des Programmablaufes sowie die Unterprogrammtechnik erfordert einen Compiler mit den Aufgaben der syntaktischen Analyse des Quellprogramms und der Generierung des CNC-Zielprogramms als Eingangssprache für den CNC-Interpreter. Der ISO- / PAL-Compiler wird als selbständige DLL bereitgestellt und bietet dem Anwender eine umfassende Unterstützung bei der Korrektur von Syntaxfehlern.
- Automatikbetrieb:** Im Automatikbetrieb wird der Programmtest durch die Möglichkeit der Aktivierung von Unterbrechungspunkten auf beliebigen NC-Sätzen sowie der Manipulation von aktuellen Werten der R-Variablen (Datentyp: Floating-Point) effizient unterstützt.
- Teach-In:** Das Teach-In kann direkt erfolgen, wenn keine selbsthemmenden Getriebe in der kinematischen Kette vorhanden sind. Dabei werden die entsprechenden Achsen mit stromlosen Motoren per Hand in die gewünschte Lage gebracht und der eingenommene Gelenkvektor wird in der Geometriedatei abgespeichert. Beim indirekten Teach-In wird innerhalb einer komplexen Dialog-Box mit Funktionstasten das Werkzeug am Handflansch in die gewünschte Zielposition / Zielorientierung gefahren.
- Software-Hierarchie** Innerhalb der Hierarchie der *isel*-Steuerungssoftware setzt die Bedien- und Programmieroberfläche **ProNC** auf einer Ebene von Geräte-DLLs für
- Bewegungssteuerung (**Motion Control**)
 - Ein- und Ausgabe (**Input / Output**)
 - Spindelsteuerung (**Spindle**)
 - Werkzeugwechslersteuerung (**Tool Changer**) u.a.
- auf. Durch dieses Hierarchie-Konzept wird es möglich, das Programmpaket für ein

großes Spektrum von isel-Steuerungshardware (z. B. IMC4- C116/C142-, IMS6/IML4-Controller, UPMV4/12-Steuerungen bzw. CVC496-Controller mit CANopen-Interface) zu applizieren, wenn die für die spezielle Hardware angebotenen speziellen Geräte-DLLs untereinander funktionskompatibel sind.

Glossar

button

Schaltfläche innerhalb einer Dialogbox (z. B. MessageBox) für Bedienhandlungen

CNC-Zieldatei

Die CNC-Zieldatei wird vom Compiler aus einer syntaktisch fehlerfreien ISO- oder PAL-Quelldatei erzeugt. Sie ist eine der Eingangsdateien für den CNC-Interpreter.

FRAME

Datenstruktur, Koordinatensystem, bestimmte Matrix

IO

(Input/Output)

Ein- und Ausgabemodul

MCTL

(Motion Control)

Bewegungssteuerungsmodul

SAA-Standard

System Application Architecture

SPN

(Spindle)

Spindelmodule

S-PTP

Synchron Punkt-zu-Punkt (Point-to-Point)

TCH

(Tool Changer)

Werkzeugwechsel

TCP

Tool Center Point: Werkzeugspitze bei Werkzeugmaschinen oder der Greifpunkt bei Handlingsystemen

Index

!		Bewegungsanweisungen.....	6
!=	102	Bewegungssteuerungsmodul	8
%		Bezeichner.....	26
%L -Deklaration.....	81	Bohren (Betriebsart Ausräumen).....	57
%SUBR -Deklaration.....	81	Bohren (Betriebsart Spanbrechen).....	57
&		Bohren mit Verweilzeit.....	57
&, , ^	99	Bohren.iso.....	123
{		Bohren.pal	123
{+, -, *, /, MODULO	93	bool_ausdruck	99
<		Bool'sche Ausdrücke.....	99
<	102	C	
<=	102	CCWABS	39
=		CCWHLXABS	45
==	102	CCWHLXREL	45
>		CCWREL	39
>	102	CNC-Interpreter	126
>=	102	CNC-Zieldatei	13
A		COOLANT OFF	67
Abfrage eines Laserdistanzmeßsystemes	111	COOLANT ON	67
ABORT	63, 64	COS	95
ABS	49, 53, 58	C-Programmierer	114
Absolutmaß	58	CWABS.....	37
Achsbenennung	21	CWHLXABS.....	46
Achsbezeichnung nach DIN 66025.....	22	CWHLXREL.....	46
Achssysteme in ProNC	21	CWREL	37
ACOS	96	D	
adressbuchstabe.....	53	Deklarationszwang	80
Adressbuchstaben	18	DELAY	41
Aktuelle Systemzeit.....	75	Dezimalzahlen	28
Aktueller Wert einer Achsposition	75	Dialogfeld für Anwendereingabe	76
Aktuelles Datum	76	direktes Teach-In.....	7
Analogausgang	74	DLL	8, 114
Anweisung.....	29	DII-Datei	112
Anwenderprogramme.....	112	DLL-Funktionen	112
Argument einer Funktion.....	93	DOS-BATCH-Files.....	112
Argumente von trigonometrischen Funktionen	95	DO-Schleife	109
arith_ausdruck.....	93	Drehsinn	37
arith_operator.....	93	DrillB	57
ASIN	95	DrillID	57
ATAN.....	96	DRILLDEF	55
Ausgangsport	72	DrillIN	57
Automatikbetrieb	126	DrillT	57
B		Drucken beliebiger Programmdatei.....	112
Bahngenerator	50	Dynamic Link Libraries	8
bedingungen	102	E	
Bewegung mit Eilganggeschwindigkeit.....	34	E-Befehl	77
		Eilganggeschwindigkeit	31, 77
		Ein- und Ausgabe.....	8
		Einfaches Bohren	57
		Eingangsport.....	71
		Elemente der Geometriedatei.....	26
		Elemente des NC-Satzes	14

ELSE.....	104	G-Befehle	10
ENDDO	109	Geometriedatei.....	7, 26
ENDFOR.....	106	Geradeninterpolation.....	35
ENDIF	104	GetBit	71
ENDSWITCH	105	GetDate.....	76
ENDWHILE.....	107	GetP	71
ERROR.....	111	GetPort.....	71
EXE-Files.....	112	Getriebeübersetzung.....	9
EXP	97	GetTime	75
F		GetTool	79
FABS	96	GetTool TC1.....	79
FALSE	29	GetTool TC2.....	79
FASTABS	34	GetValue	76
FASTFRAME	42	Gleitkommazahl	85
FASTVEL-Befehl	77	H	
F-Befehl	77	Haftung.....	6
Figures.iso	121	Hand-Modus.....	70
Figures.pal	121	Hauptprogramm	12
FLOOR	97	Helixinterpolation CCWHLX.....	46
Folge von NC-Sätzen	29	Helixinterpolation CWHLX	45
FOR-Schleife	106	Hexadezimalzahlen.....	27
FRAME	26	Hilfs- oder Nebenachsen.....	22
Framedatei.....	7	HOFF.....	70
Frame-Name.....	27	HON	70
funktionen	95	I	
funktions_bezeichner.....	95	IDABORT	93
G		IDCANCEL	93
G0	34	IDIGNORE	93
G1	35	IDNO	93
G10.....	42	IDOK.....	93
G11.....	43	IDRETRY	93
G17.....	48	IDYES.....	93
G18.....	48	IF-Konstruktion.....	104
G19.....	48	INCH.....	52
G2	37	Indexierung von Q-Variablen	42
G3	39	indirektes Teach-In.....	8
G4	41	INFO.....	111
G53	49	Initialisierungsdatei.....	9
G54	49	Initialisierungsdatei des	
G55	49	Bewegungssteuerungsmoduls	9
G56.....	49	Input / Output IO.....	8
G60.....	50	Interpolationsebene.....	31, 48
G64.....	50	ISO-Syntax	7
G70.....	52	K	
G71.....	52	Kettenmaß.....	58
G74.....	53	Kommentare.....	13
G75.....	54	Kommentarfilter	14
G80.....	55	Kommentarzeile	14
G81.....	57	konstante.....	99
G82.....	57	Kontur.iso	122
G83.....	57	Kontur.pal	122
G84.....	57	Koordinaten.....	21
G90.....	58	Koordinaten-Worte.....	11
G91.....	58	Kreisinterpolation ccw	39
G92.....	59	Kreisinterpolation clw	55
G93.....	59	Kreisinterpolation cw	37
G98.....	60	Kreiszahl	28
G99.....	61		

Kühlflüssigkeit	67	POPTION1 ON/POPTION1 OFF	69
L		POPTION2 ON/POPTION2 OFF	69
LAMP OFF	68	POSn.A	75
LAMP ON	68	POW	98
Lampe	68	Programmaufbau	10
Länge eines NC-Satzes	16	Programmbeginn	64
Laufzeit	86	Programmende	64
L-Befehl	82	programmierbarer Abbruch	47
leere Anweisung	25	Programmtext	25
leerer Schleifenkörper	107	Programmunterbrechung	63
Lernen.iso	120	Programmverzweigung	104
Lernen.pal	120	PUMP OFF	68
Lesen Eingangs- /Ausgangsport	71	PUMP ON	68
Linearachsen	22	Pumpe	68
LN	97	P-Variablen	85, 87
LOG	97	PWM-Ausgang	74
M		Q	
M0	63, 64	QUESTION	111
M1	63, 64	QUIT	63, 64
M10	67	Q-Variablen	85, 88
M11	67	R	
M3	65	r_variable	93
M4	65	Radiant	95
M5	65	Real-Variablen	90
M8	67	reelle Funktionen	28
M9	67	REF	53
Manipulation von Technologie-Variablen	60	Referenzpunktfahrt	53
Maßeinheit	52	Reihenfolge der einzelnen Worte in einem Satz	17
M-Befehle	10	REL	58
MessageBox	111	REPEAT	109
METRIC	52	ResBit	72
Mittelpunktskoordinaten	37	rotatorische Achsen	22
Mnemonic	10	R-Variablen	85, 90
Modalität	11	S	
Modulodivision	93	SAA-Standard	7
Motion Control MCTL	8	Satznummer	20
MOVEABS	35	Satzunterdrückung	20
MOVEFRAME	43	S-Befehl	78
MOVEREL	35	SCCLW	65
Mpby	72	Schachtelungstiefe	80
N		Schlüsselworte	25
natürliche oder Dezimalzahlen	18	Schrift.iso	124
Nullpunkt vermessen	122	SCLW	65
Nullpunktregister	59	Segmentgeschwindigkeit	31, 77
Nullpunktverschiebung	31, 49	SetAnalog	74
P		SetBit	72
PAL-Syntax	7	SetP	72
PARAMETER	60	SetPort	72
Parameterrechnung	93	SetPWM	74
PATH	50	Setzen Ausgangsport	72
PATHEND	50	SIN	95
Peripherieoption	69	SOFF	65
Pi93		Sonderzeichen	22
PLANE XY	48	Spannungswert	74
PLANE XZ	48	Spindel ausschalten	65
PLANE YZ	48	Spindel einschalten	65

Spindeldrehzahl	78	Unterbrechungspunkt.....	7
Spindelmodul	8	Unterprogramm-Aufruf	82
Spindle	66	Unterprogramm-Deklaration	81
Spindle CCW	66	Unterprogrammtechnik.....	80
Spindle CW	66	UNTIL	109
Spindle OFF.....	66	USER	112
Spindle ON	66	USERDLL.....	112
Spindle SPN	8	USEREXE	112
S-PTP-Bewegung	35	V	
SQR	97	Variable	26
SQRT	96	VEL-Befehl	77
startwert	106	Verweilzeit	41
Steuerungsmodule.....	8	Vorschub	15
Steuerungsmodulkonzept.....	8	W	
SUBR-Befehl	82	WARNING	111
SWITCH.....	105	Wegbefehle in ProNC	33
SWITCH-Konstruktion	105	Werkstück spannen.....	67
symb_konstante.....	93	Werkstücknullpunktregister	59
T		Werkzeugwechsel	79
T1-Befehl	79	Werkzeugwechselmodul	8
T2-Befehl	79	Wertübergabe	92
TAN.....	96	WHILE	107
Taschen.iso	123	WHILE-Schleife.....	107
Taschen.pal	123	Windows-DLL.....	112
Tastverhältnis	74	Windows-EXE	112
T-Befehl	79	Wort.....	10
TEACH.....	54	WPCLAMP OFF	67
Test-Modus	70	WPCLAMP ON.....	67
TIME	41	WPCLEAR	49
TOFF	70	WPREG1.....	49
Token	28	WPREG1WRITE	59
TON	70	WPREG2.....	49
Tool Changer TCH.....	8	WPREG2WRITE	59
translatorische Achsen	22	WPZERO	49
Trennzeichen	15	Z	
trigonometrische Funktionen	28	Zählschleife	106
TRUE	29	zuweisung	100
TYPE	61		
U			
UND, ODER bzw. EXCLUSIV ODER.....	99		

***isel*GermanyAG**

Bürgermeister-Ebert-Str. 40
D-36124 Eichenzell
Tel.: +49 (0) 6659 / 981-700
Fax: +49 (0) 6659 / 981-776

***isel*GermanyAG**

Untere Röde 2
D-36466 Dermbach
Tel.: +49 (0) 36964 / 84-4
Fax: +49 (0) 36964 / 84-510