SIEMENS

SINUMERIK

SINUMERIK 840D sl / 828D Synchronaktionen

Funktionshandbuch

Gültig für

Steuerung
SINUMERIK 840D sl / 840DE sl
SINUMERIK 828D

Software	Version
CNC-Software	4.7 SP2

10/2015
6FC5397-5BP40-5AA3

Rechtliche Hinweise

Warnhinweiskonzept

Dieses Handbuch enthält Hinweise, die Sie zu Ihrer persönlichen Sicherheit sowie zur Vermeidung von Sachschäden beachten müssen. Die Hinweise zu Ihrer persönlichen Sicherheit sind durch ein Warndreieck hervorgehoben, Hinweise zu alleinigen Sachschäden stehen ohne Warndreieck. Je nach Gefährdungsstufe werden die Warnhinweise in abnehmender Reihenfolge wie folgt dargestellt.

⚠ GEFAHR
bedeutet, dass Tod oder schwere Körperverletzung eintreten **wird**, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

⚠ WARNUNG
bedeutet, dass Tod oder schwere Körperverletzung eintreten **kann**, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

⚠ VORSICHT
bedeutet, dass eine leichte Körperverletzung eintreten kann, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

ACHTUNG
bedeutet, dass Sachschaden eintreten kann, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

Beim Auftreten mehrerer Gefährdungsstufen wird immer der Warnhinweis zur jeweils höchsten Stufe verwendet. Wenn in einem Warnhinweis mit dem Warndreieck vor Personenschäden gewarnt wird, dann kann im selben Warnhinweis zusätzlich eine Warnung vor Sachschäden angefügt sein.

Qualifiziertes Personal

Das zu dieser Dokumentation zugehörige Produkt/System darf nur von für die jeweilige Aufgabenstellung **qualifiziertem Personal** gehandhabt werden unter Beachtung der für die jeweilige Aufgabenstellung zugehörigen Dokumentation, insbesondere der darin enthaltenen Sicherheits- und Warnhinweise. Qualifiziertes Personal ist auf Grund seiner Ausbildung und Erfahrung befähigt, im Umgang mit diesen Produkten/Systemen Risiken zu erkennen und mögliche Gefährdungen zu vermeiden.

Bestimmungsgemäßer Gebrauch von Siemens-Produkten

Beachten Sie Folgendes:

⚠ WARNUNG
Siemens-Produkte dürfen nur für die im Katalog und in der zugehörigen technischen Dokumentation vorgesehenen Einsatzfälle verwendet werden. Falls Fremdprodukte und -komponenten zum Einsatz kommen, müssen diese von Siemens empfohlen bzw. zugelassen sein. Der einwandfreie und sichere Betrieb der Produkte setzt sachgemäßen Transport, sachgemäße Lagerung, Aufstellung, Montage, Installation, Inbetriebnahme, Bedienung und Instandhaltung voraus. Die zulässigen Umgebungsbedingungen müssen eingehalten werden. Hinweise in den zugehörigen Dokumentationen müssen beachtet werden.

Marken

Alle mit dem Schutzrechtsvermerk ® gekennzeichneten Bezeichnungen sind eingetragene Marken der Siemens AG. Die übrigen Bezeichnungen in dieser Schrift können Marken sein, deren Benutzung durch Dritte für deren Zwecke die Rechte der Inhaber verletzen kann.

Haftungsausschluss

Wir haben den Inhalt der Druckschrift auf Übereinstimmung mit der beschriebenen Hard- und Software geprüft. Dennoch können Abweichungen nicht ausgeschlossen werden, so dass wir für die vollständige Übereinstimmung keine Gewähr übernehmen. Die Angaben in dieser Druckschrift werden regelmäßig überprüft, notwendige Korrekturen sind in den nachfolgenden Auflagen enthalten.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
DEUTSCHLAND

Dokumentbestellnummer: 6FC5397-5BP40-5AA3
℗ 10/2015 Änderungen vorbehalten

Vorwort

SINUMERIK-Dokumentation

Die SINUMERIK-Dokumentation ist in folgende Kategorien gegliedert:

- Allgemeine Dokumentation
- Anwender-Dokumentation
- Hersteller/Service-Dokumentation

Weiterführende Informationen

Unter dem Link www.siemens.com/motioncontrol/docu finden Sie Informationen zu folgenden Themen:

- Dokumentation bestellen / Druckschriftenübersicht
- Weiterführende Links für den Download von Dokumenten
- Dokumentation online nutzen (Handbücher/Informationen finden und durchsuchen)

Bei Fragen zur Technischen Dokumentation (z. B. Anregungen, Korrekturen) senden Sie bitte eine E-Mail an folgende Adresse:

docu.motioncontrol@siemens.com

My Documentation Manager (MDM)

Unter folgendem Link finden Sie Informationen, um auf Basis der Siemens Inhalte eine OEM-spezifische Maschinen-Dokumentation individuell zusammenzustellen:

www.siemens.com/mdm

Training

Informationen zum Trainingsangebot finden Sie unter:

- www.siemens.com/sitrain
  SITRAIN - das Training von Siemens für Produkte, Systeme und Lösungen der Automatisierungstechnik
- www.siemens.com/sinutrain
  SinuTrain - Trainingssoftware für SINUMERIK

FAQs

Frequently Asked Questions finden Sie in den Service&Support Seiten unter Produkt Support.
http://support.automation.siemens.com

SINUMERIK

Informationen zu SINUMERIK finden Sie unter folgendem Link:

www.siemens.com/sinumerik

Zielgruppe

Die vorliegende Druckschrift wendet sich an:

- Projekteure
- Technologen (von Maschinenherstellern)
- Inbetriebnehmer (von Systemen/Maschinen)
- Programmierer

Nutzen

Das Funktionshandbuch beschreibt die Funktionen, so dass die Zielgruppe die Funktionen kennt und auswählen kann. Es befähigt die Zielgruppe, die Funktionen in Betrieb zu nehmen.

Standardumfang

In der vorliegenden Dokumentation ist die Funktionalität des Standardumfangs beschrieben. Ergänzungen oder Änderungen, die durch den Maschinenhersteller vorgenommen werden, werden vom Maschinenhersteller dokumentiert.

Es können in der Steuerung weitere, in dieser Dokumentation nicht erläuterte Funktionen ablauffähig sein. Es besteht jedoch kein Anspruch auf diese Funktionen bei der Neulieferung bzw. im Servicefall.

Ebenso enthält diese Dokumentation aus Gründen der Übersichtlichkeit nicht sämtliche Detailinformationen zu allen Typen des Produkts und kann auch nicht jeden denkbaren Fall der Aufstellung, des Betriebes und der Instandhaltung berücksichtigen.

Technical Support

Landesspezifische Telefonnummern für technische Beratung finden Sie im Internet unter http://www.siemens.com/automation/service&support

Inhaltsverzeichnis

Grundlegende Sicherheitshinweise 1

1.1 Allgemeine Sicherheitshinweise

⚠ WARNUNG

Lebensgefahr durch Nichtbeachtung von Sicherheitshinweisen und Restrisiken

Durch Nichtbeachtung der Sicherheitshinweise und Restrisiken in der zugehörigen Hardware-Dokumentation können Unfälle mit schweren Verletzungen oder Tod auftreten.

- Halten Sie die Sicherheitshinweise der Hardware-Dokumentation ein.
- Berücksichtigen Sie bei der Risikobeurteilung die Restrisiken.

⚠ WARNUNG

Lebensgefahr durch Fehlfunktionen der Maschine infolge fehlerhafter oder veränderter Parametrierung

Durch fehlerhafte oder veränderte Parametrierung können Fehlfunktionen an Maschinen auftreten, die zu Körperverletzungen oder Tod führen können.

- Schützen Sie die Parametrierungen vor unbefugtem Zugriff.
- Beherrschen Sie mögliche Fehlfunktionen durch geeignete Maßnahmen (z. B. NOT-HALT oder NOT-AUS).

1.2 Industrial Security

Hinweis

Industrial Security

Siemens bietet Produkte und Lösungen mit Industrial Security-Funktionen an, die den sicheren Betrieb von Anlagen, Lösungen, Maschinen, Geräten und/oder Netzwerken unterstützen. Sie sind wichtige Komponenten in einem ganzheitlichen Industrial Security-Konzept. Die Produkte und Lösungen von Siemens werden unter diesem Gesichtspunkt ständig weiterentwickelt. Siemens empfiehlt, sich unbedingt regelmäßig über Produkt-Updates zu informieren.

Für den sicheren Betrieb von Produkten und Lösungen von Siemens ist es erforderlich, geeignete Schutzmaßnahmen (z. B. Zellenschutzkonzept) zu ergreifen und jede Komponente in ein ganzheitliches Industrial Security-Konzept zu integrieren, das dem aktuellen Stand der Technik entspricht. Dabei sind auch eingesetzte Produkte von anderen Herstellern zu berücksichtigen. Weitergehende Informationen über Industrial Security finden Sie unter dieser Adresse (http://www.siemens.com/industrialsecurity).

Um stets über Produkt-Updates informiert zu sein, melden Sie sich für unseren produktspezifischen Newsletter an. Weitere Informationen hierzu finden Sie unter dieser Adresse (http://support.automation.siemens.com).

⚠ WARNUNG

Gefahr durch unsichere Betriebszustände wegen Manipulation der Software

Manipulationen der Software (z. B. Viren, Trojaner, Malware, Würmer) können unsichere Betriebszustände in Ihrer Anlage verursachen, die zu Tod, schwerer Körperverletzung und zu Sachschäden führen können.

- Halten Sie die Software aktuell.
  Informationen und Newsletter hierzu finden Sie unter dieser Adresse (http://support.automation.siemens.com).
- Integrieren Sie die Automatisierungs- und Antriebskomponenten in ein ganzheitliches Industrial Security-Konzept der Anlage oder Maschine nach dem aktuellen Stand der Technik.
  Weitergehende Informationen finden Sie unter dieser Adresse (http://www.siemens.com/industrialsecurity).
- Berücksichtigen Sie bei Ihrem ganzheitlichen Industrial Security-Konzept alle eingesetzten Produkte.

Kurzbeschreibung 2

Allgemein

Eine Synchronaktion besteht aus einer Reihe von zusammengehörenden Anweisungen innerhalb eines Teileprogramms, die synchron zu den Bearbeitungssätzen, zyklisch im Interpolatortakt ausgeführt werden.

Eine Synchronaktion gliedert sich im Wesentlichen in zwei Teile, dem optionalen Bedingungs- und dem obligatorischen Aktionsteil. Über den Bedingungsteil kann der Ausführungszeitpunkt der Aktionen von einem bestimmten Systemzustand abhängig gemacht werden. Die Bedingungen werden zyklisch im Interpolatortakt ausgewertet. Die Aktionen sind dann eine Reaktion auf vom Anwender definierbare Systemzustände. Ihre Ausführung ist dabei nicht an Satzgrenzen gebunden.

Des Weiteren kann die Gültigkeit der Synchronaktion (satzweise, modal oder statisch) und die Häufigkeit der Ausführung der Aktionen (einmalig, wiederholt) festgelegt werden.

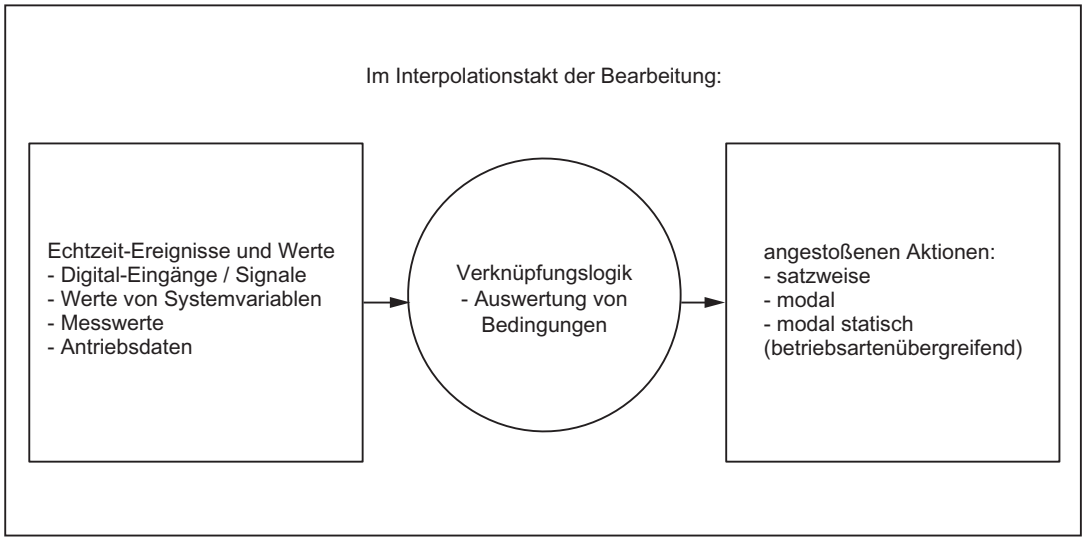
Beispiele von zulässigen Aktionen

- Ausgabe von Hilfsfunktionen an die PLC
- Schreiben und Lesen von Hauptlaufvariablen
- Verfahren von Positionierachsen
- Aktivierung von Synchronprozeduren wie z. B.:
  - Einlesesperre
  - Restweglöschen
  - Vorlaufstopp beenden
- Aktivierung von Technologiezyklen
- Berechnung von Funktionswerten
- Werkzeugkorrekturen
- Kopplungen aktivieren / deaktivieren
- Messen
- Sperren / Freigeben von Synchronaktionen

Beispiele von nicht zulässigen Aktionen

- Verfahren von Bahnachsen

Synchronaktionen schematisch



Ausführliche Beschreibung 3

3.1 Definition einer Synchronaktion

Eine Synchronaktion wird in einem Satz eines Teileprogramms definiert. Innerhalb dieses Satzes dürfen keine weiteren Befehle programmiert werden, die nicht Bestandteil der Synchronaktion sind.

Komponenten einer Synchronaktion

Eine Synchronaktion besteht aus folgenden Komponenten:

Gültigkeit, Ident-Nr. (Seite 14) (optional)	Bedingungsteil (optional)			Aktionsteil		
	Häufigkeit (Seite 15)	G-Funktion (Seite 16) (optional)	Bedingung (Seite 17)	Schlüsselwort	G-Funktion (Seite 17) (optional)	Aktionen (Seite 18)
--- [1] ID=<Nr> IDS=<Nr>	--- [1] WHENEVER FROM WHEN EVERY	G...	Logischer Ausdruck	DO	G...	Aktion 1 ... Aktion n

[1] nicht programmiert

Syntax

Beispiele:

1. `DO <Aktion 1...n>`
2. `<Häufigkeit> [<G-Funktion>] <Bedingung> DO <Aktion 1...n>`
3. `ID=<Nr> <Häufigkeit> [<G-Funktion>] <Bedingung> DO <Aktion 1...n>`
4. `IDS=<Nr> <Häufigkeit> [<G-Funktion>] <Bedingung> DO <Aktion 1...n>`

3.2 Komponenten von Synchronaktionen

3.2.1 Gültigkeit, Identifikationsnummern (ID, IDS)

Gültigkeit

Mit der Gültigkeit wird festgelegt, wann und wo die Synchronaktion bearbeitet wird:

Gültigkeit	Bedeutung

--- [1]	Satzweise Synchronaktion
	Eine satzweise Synchronaktion wirkt:
	• solange der auf die Synchronaktion folgende Hauptlaufsatz aktiv ist
	• nur in der Betriebsart AUTOMATIK
	Beispiel:
	Die Synchronaktion aus N10 wirkt solange N20 aktiv ist.
	`N10 WHEN $A_IN[1]==TRUE DO $A_OUTA[1]=10`
	`N20 G90 F1000 X100`

ID=<ID-Nummer>	Modale Synchronaktion Eine modale Synchronaktion wirkt: • solange, bis das Teileprogramm beendet ist • nur in der Betriebsart AUTOMATIK Wertebereich: siehe unten Absatz "Identifikationsnummer" > "Wertebereich" Beispiel: `N20 ID=1 EVERY $A_IN[1]==TRUE DO $A_OUTA[1]=10`
IDS=<ID-Nummer>	Statische Synchronaktion Eine statische Synchronaktion wirkt: • zeitlich unbegrenzt in allen Betriebsarten Wertebereich: siehe unten Absatz "Identifikationsnummer" > "Wertebereich" Beispiel: `N30 IDS=1 EVERY $A_IN[1]==TRUE DO $A_OUTA[1]=10`

[1] nicht programmiert

Hinweis

Statische Synchronaktionen

Statische Synchronaktionen (`IDS`) können in einem ASUP definiert und durch Aktivierung des ASUP über das PLC-Anwenderprogramm zu einem beliebigen Zeitpunkt aktiviert werden.

Identifikationsnummer ID/IDS

Wertebereich

Die Identifikationsnummern ID/IDS liegen in verschiedenen Nummernbändern. Die Nummernbänder sind verschiedenen Anwender zugeordnet.

ID / IDS	Anwender	Verzeichnis
1 ... 999	Endanwender "Safety Integrated"	Beliebiges Verzeichnis /_N_CST_DIR/_N_SAFE_SPF
1000 ... 1199	Maschinenhersteller	/_N_CMA_DIR
1200 ... 1399	Siemens	/_N_CST_DIR

Parallelisierung

Sollen in einem Kanal mehrere Synchronaktionen parallel aktiv sein, müssen deren Identifikationsnummern ID/IDS voneinander verschieden sein. Synchronaktionen mit der gleichen Identifikationsnummer lösen sich innerhalb eines Kanals ab.

Bearbeitungsreihenfolge

Modale und statische Synchronaktionen werden in der Reihenfolge ihrer Identifikationsnummern ID/IDS bearbeitet.

Satzweise Synchronaktionen werden nach der Bearbeitung der modalen Synchronaktionen in der Reihenfolge ihrer Programmierung bearbeitet.

Koordinierung über Teileprogramme und Synchronaktionen

Anhand der Identifikationsnummern ID/IDS können Synchronaktionen über Teileprogramme und Synchronaktionen koordiniert werden (siehe Kapitel "Koordinierung über Teileprogramm und Synchronaktion (LOCK, UNLOCK, RESET, CANCEL) (Seite 118)").

Koordinierung über PLC

Synchronaktionen mit Identifikationsnummern ID/IDS im Bereich von 1 bis 64 können über die NC/PLC-Nahtstelle vom PLC-Anwenderprogramm aus koordiniert werden (siehe Kapitel "Koordinierung über PLC (Seite 118)").

3.2.2 Häufigkeit (WHENEVER, FROM, WHEN, EVERY)

Über die Häufigkeit wird angegeben, wie oft die Bedingung abgefragt und bei erfüllter Bedingung die Aktion ausgeführt werden soll. Die Häufigkeit ist Bestandteil der Bedingung.

Häufigkeit	Bedeutung
--- [1)]	Ist keine Häufigkeit angegeben, wird die Aktion zyklisch in jedem Interpolatortakt ausgeführt.
WHENEVER	Ist die Bedingung erfüllt, wird die Aktion zyklisch in jedem Interpolatortakt ausgeführt.
FROM	Nachdem die Bedingung einmal erfüllt wurde, wird die Aktion zyklisch in jedem Interpolatortakt ausgeführt, solange die Synchronaktion aktiv ist.

Häufigkeit	Bedeutung
WHEN	Wird die Bedingung erfüllt, wird die Aktion ein Mal ausgeführt und anschließend die Bedingung nicht mehr geprüft.
EVERY	Bei jedem Zustandsübergang der Bedingung von FALSE nach TRUE (steigende Flanke), wird die Aktion ein Mal ausgeführt.

1) nicht programmiert

Siehe auch

Technologiezyklen (Seite 112)

3.2.3 G-Funktion (Bedingung)

Definierter Ausgangszustand

Bezogen auf den Teileprogrammablauf können Synchronaktionen, abhängig von der Erfüllung der Bedingung, zu beliebigen Zeitpunkten ausgeführt werden. Es wird daher empfohlen, in einer Synchronaktion **vor** der Bedingung und/oder im Aktionsteil das erforderliche Maßsystem (inch oder metrisch) festzulegen. Dadurch wird für die Auswertung der Bedingung und der Ausführung der Aktion eine definierte Ausgangsstellung, unabhängig vom aktuellen Teileprogrammzustand, erzeugt.

G-Funktionen

Folgende G-Funktionen sind zulässig:

- G70 (Inch-Maßangabe für geometrische Angaben (Längen))
- G71 (Metrische Maßangabe für geometrische Angaben (Längen))
- G700 (Inch-Maßangabe für geom. und technologische Angaben (Längen, Vorschub))
- G710 (Metrische Maßangabe für geom. und technologische Angaben (Längen, Vorschub))

Hinweis

In Synchronaktionen sind außer G70, G71, G700 und G710 keine weiteren G-Funktionen zulässig.

Gültigkeit

Eine im Bedingungsteil programmierte G-Funktion gilt auch für den Aktionsteil, wenn im Aktionsteil selbst keine eigene G-Funktion programmiert wurde.

Eine im Aktionsteil programmierte G-Funktion gilt nur innerhalb des Aktionsteils.

3.2.4 Bedingung

Die Ausführung der Aktion kann von der Erfüllung einer Bedingung abhängig gemacht werden. Solange die Synchronaktion aktiv ist, wird die Bedingung zyklisch im Interpolatortakt überprüft. Ist keine Bedingung angegeben, wird die Aktion zyklisch in jedem Interpolatortakt ausgeführt.

Als Bedingung können alle Operationen programmiert werden, die als Ergebnis einen Wahrheitswert (TRUE / FALSE) liefern:

- Vergleiche von Systemvariablen mit Konstanten
- Vergleiche von Systemvariablen mit Systemvariablen
- Vergleiche von Systemvariablen mit Rechenergebnissen
- Verknüpfung von Vergleichen durch Boole'sche Ausdrücke

Beispiele

Vergleiche

Programmcode

```
ID=1 WHENEVER $AA_IM[X] > $$AA_IM[Y] DO ...
ID=2 WHENEVER $AA_IM[X] > (10.5 * SIN(45)) DO ...
```

Boole'sche Verknüpfungen

Programmcode

```
ID=1 WHENEVER ($A_IN[1]==1) OR ($A_IN[3]==0) DO ...
```

Siehe auch

Lesen und Schreiben (Seite 19)

Beispiele für Bedingungen in Synchronaktionen (Seite 131)

Systemvariable für Synchronaktionen (Seite 19)

3.2.5 G-Funktion (Aktion)

Definierter Ausgangszustand

Bezogen auf den Teileprogrammablauf können Synchronaktionen, abhängig von der Erfüllung der Bedingung, zu beliebigen Zeitpunkten ausgeführt werden. Daher ist es sinnvoll, in einer Synchronaktion im Aktionsteil das erforderliche Maßsystem (inch oder metrisch) festzulegen. Dadurch wird für die Ausführung der Aktion eine definierte Ausgangsstellung, unabhängig vom aktuellen Teileprogrammzustand, erzeugt.

G-Funktionen

Folgende G-Funktionen sind zulässig:

- `G70` (Inch-Maßangabe für geometrische Angaben (Längen))
- `G71` (Metrische Maßangabe für geometrische Angaben (Längen))
- `G700` (Inch-Maßangabe für geom. und technologische Angaben (Längen, Vorschub))
- `G710` (Metrische Maßangabe für geom. und technologische Angaben (Längen, Vorschub))

Gültigkeit

Eine im Bedingungsteil programmierte G-Funktion gilt auch für den Aktionsteil, wenn im Aktionsteil selbst keine eigene G-Funktion programmiert wurde.

Eine im Aktionsteil programmierte G-Funktion gilt nur innerhalb des Aktionsteils.

3.2.6 Aktion (DO)

Der Aktionsteil einer Synchronaktion wird eingeleitet mit dem Schlüsselwort `DO`.

Im Aktionsteil können eine oder mehrere Aktionen programmiert werden. Diese werden ausgeführt, wenn die Bedingung erfüllt ist. Sind mehrere Aktionen in einer Synchronaktion programmiert, werden alle im selben Interpolatortakt ausgeführt.

Beispiel:

Ist der Istwert der Y-Achse größer oder gleich 35.7, wird die Hilfsfunktion M135 an PLC ausgegeben und gleichzeitig der digitale Ausgang 1 = 1 gesetzt.

Programmcode

```
WHEN $AA_IM[Y] >= 35.7 DO M135 $A_OUT[1]=1
```

Technologiezyklus

Als Aktion kann ein Technologiezyklus aufgerufen werden (siehe Kapitel "Technologiezyklen (Seite 112)").

3.3 Systemvariable für Synchronaktionen

Die Systemvariablen des NCK sind im Listenhandbuch "Systemvariable" mit ihren jeweiligen Eigenschaften aufgelistet. Systemvariablen die in Synchronaktionen gelesen bzw. geschrieben werden können, sind in der entsprechenden Zeile (Read bzw. Write) der Spalte "SA" (Synchronaktion) mit "X" gekennzeichnet.

Hinweis

Systemvariable die in Synchronaktionen verwendet werden, werden implizit immer hauptlaufsynchron gelesen und geschrieben.

Literatur

Eine umfassende Beschreibung zu den in diesem Funktionshandbuch aufgeführten Systemvariablen findet sich in:

- Listenhandbuch Systemvariable

3.3.1 Lesen und Schreiben

Das Lesen und Schreiben von Variablen erfolgt in Synchronaktionen bis auf wenige Ausnahmen im Hauptlauf. Ausnahmen sind:

- Anwenderdefinierte Variablen: LUD, GUD
- Maschinendaten: $M...
- Settingdaten: $S...
- R-Paramter: R<Nummer> oder R[<Index>]

Diese Variablen werden bereits im Vorlauf gelesen und geschrieben.

Systemvariablen

Allgemein gilt für alle Systemvariablen die in Synchronaktionen verwendet werden können, dass diese im Hauptlauf gelesen/geschrieben werden. Diese Systemvariablen sind im Listenhandbuch "Systemvariable" in der Spalte "SA" (Synchronaktion) in der Zeile "Read" und/oder "Write" mit "X" gekennzeichnet.

Literatur:
Listenhandbuch Systemvariable

Systematik der Bezeichner

Die Bezeichner der Systemvariablen, die im Hauptlauf gelesen/geschrieben werden, haben folgende Systematik:

$A...	Aktuelle Hauptlaufdaten
$V...	Servo-Daten

$R...	Im Hauptlauf zu lesender/schreibender R-Parameter
$$M...	Im Hauptlauf zu lesende/schreibende Maschinendaten
$$S...	Im Hauptlauf zu lesende/schreibende Settingdaten

3.3.2 Operatoren und Rechenfunktionen

Operatoren

Rechenoperatoren

Systemvariable vom Typ REAL oder INT können durch folgende Operatoren miteinander verknüpft werden:

Operator	Bedeutung
+	Addition
-	Subtraktion
*	Multiplikation
/	Division, **Achtung**: INT / INT = **REAL**
DIV	Integer-Division, **Achtung**: INT / INT = **INT**
MOD	Modulo-Division, (nur für Typ INT) liefert Rest einer INT-Division Beispiel: 3 MOD 4 = 3

Hinweis

Es können nur Variablen gleichen Typs verknüpft werden.

Vergleichsoperatoren

Operator	Bedeutung
==	gleich
>	ungleich
<	kleiner
>	größer
<=	kleiner oder gleich
>=	größer oder gleich

Boole'sche Operatoren

Operator	Bedeutung
NOT	NICHT
AND	UND
OR	ODER
XOR	Exclusiv-ODER

Bitweise logische Operatoren

Operator	Bedeutung
B_OR	bitweise ODER
B_AND	bitweise UND
B_XOR	bitweise exklusives ODER
B_NOT	bitweise Negierung

Priorität der Operatoren

Die Operatoren haben bei der Abarbeitung in der Synchronaktion folgende Prioritäten (höchste Priorität: 1):

Prio.	Operatoren	Bedeutung
1	NOT, B_NOT	Verneinung, bitweise Verneinung
2	*, /, DIV, MOD	Multiplikation, Division
3	+, -	Addition, Subtraktion
4	B_AND	bitweise UND
5	B_XOR	bitweise exklusives ODER
6	B_OR	bitweise ODER
7	AND	UND
8	XOR	exklusives ODER
9	OR	ODER
10	<<	Verkettung von Strings, Ergebnistyp STRING
11	==, <>, <, >, >=, <=	Vergleichsoperatoren

Hinweis

Es wird dringend empfohlen, bei der Verwendung von mehreren Operatoren in einem Ausdruck, die einzelnen Operatoren durch das Setzen von Klammern "(…)" eindeutig zu priorisieren.

Beispiel einer Bedingung mit einem Ausdruck mit mehreren Operatoren:

Programmcode

```
... WHEN ($AA_IM[X] > WERT) AND ($AA_IM[Y] > WERT1) DO ...
```

Rechenfunktionen

Operator	Bedeutung
Sin()	Sinus
COS()	Cosinus
TAN()	Tangens
ASIN()	Arcussinus
ACOS()	Arcuscosinus
ATAN2(,)	Arcustangens2
SQRT()	Quadratwurzel

Operator	Bedeutung
ABS()	Betrag
POT()	2. Potenz (Quadrat)
TRUNC()	ganzzahliger Teil Genauigkeiten bei Vergleichsbefehlen einstellbar mit TRUNC
ROUND()	Runden auf eine ganze Zahl
LN()	natürlicher Logarithmus
EXP()	Exponentialfunktion

Eine ausführliche Beschreibung der Funktionen findet sich in:
Literatur
Programmierhandbuch Arbeitsvorbereitung, Kapitel "Flexible NC-Programmierung" ff.

Indizierung

Der Index einer Systemvariablen vom Typ "Feld von …" kann wiederum eine Systemvariable sein. Der Index wird dabei ebenfalls im Hauptlauf im Interpolatortakt ausgewertet.

Beispiel

Programmcode

```
... WHEN ... DO $AC_PARAM[ $AC_MARKER[1] ] = 3
```

Einschränkungen

- Eine Schachtelung der Indizierung mit weiteren Systemvariablen ist nicht erlaubt.
- Der Index darf nicht über Vorlaufvariablen gebildet werden. Das folgende Beispiel ist somit **nicht** erlaubt, da $P_EP eine Vorlaufvariable ist:
  $AC_PARAM[1] = $P_EP[$AC_MARKER[0]]

3.3.3 Typwandlungen

Bei Wertzuweisungen und Parameterübergaben mit unterschiedlichen Datentypen erfolgt eine implizite Typwandlung zwischen folgenden Datentypen:

- REAL
- INT
- BOOL

Hinweis

Konvertierung von REAL nach INT

Bei der Konvertierung von REAL nach INT wird bei einem Nachkommawert ≧ 0.5 auf die nächste größere Ganzzahl aufgerundet. Bei einem Nachkommawert < 0.5 wird auf die nächste kleinere Ganzzahl abgerundet. Verhalten entsprechend der Funktion `ROUND`.

Liegt der REAL-Wert außerhalb des INT-Wertebereichs, wird ein Alarm angezeigt und die Konvertierung wird nicht durchgeführt.

Konvertierung von REAL oder INT nach BOOL

- Wert <> 0 → TRUE
- Wert == 0 → FALSE

Beispiele

Konvertierung: **INT** $AC_MARKER → **REAL** $AC_PARAM

Programmcode

```
$AC_MARKER[1]=561
ID=1 WHEN TRUE DO $AC_PARAM[1] = $AC_MARKER[1]
```

Konvertierung: **REAL** $AC_PARAM → **INT** $AC_MARKER

Programmcode

```
$AC_PARAM[1]=561.0
ID=1 WHEN TRUE DO $AC_MARKER[1] = $AC_PARAM[1]
```

Konvertierung: **INT** $AC_MARKER → **BOOL** $A_OUT

Programmcode

```
$AC_MARKER[1]=561
ID=1 WHEN $A_IN[1]==TRUE DO $A_OUT[0] = $AC_MARKER[1]
```

Konvertierung: **REAL** $R401 → **BOOL** $A_OUT

Programmcode

```
R401 = 100.542
WHEN $A_IN[0]==TRUE DO $A_OUT[2] = $R401
```

Konvertierung: **BOOL** $A_OUT → **INT** $AC_MARKER

Programmcode

```
ID=1 WHEN $A_IN[2]==TRUE DO $AC_MARKER[4] = $A_OUT[1]
```

Konvertierung: **BOOL** $A_OUT → **REAL** $R10

```
Programmcode
WHEN $A_IN[3]==TRUE DO $R10 = $A_OUT[3]
```

3.3.4 Merker-/Zähler ($AC_MARKER)

Die Variablen $AC_MARKER[<Index>] sind kanalspezifische Felder von Systemvariablen zur Verwendung als Merker oder Zähler.

Datentyp: INT (Integer)
<Index>: Feldindex: 0, 1, 2, ... (max. Anzahl - 1)

Anzahl pro Kanal

Die maximale Anzahl der $AC_MARKER Variablen pro Kanal ist einstellbar über das Maschinendatum:

MD28256 $MC_MM_NUM_AC_MARKER = <Maximale Anzahl>

Speicherort

Der Speicherort der $AC_MARKER Variablen kann kanalspezifisch festgelegt über das Maschinendatum:

MD28257 $MC_MM_BUFFERED_AC_MARKER = <Wert>

Wert	Speicherort
0	Dynamischer Speicher (Standardeinstellung)
1	Statischer Speicher

Hinweis

Datensicherung und Speicherplatz

- Die im statischen Speicher angelegten $AC_MARKER Variablen können kanalspezifisch über die Datensicherung gesichert werden. Datenbaustein: _N_CH<Kanalnummer>_ACM
- Es muss darauf geachtet werden, dass im gewählten Speicherbereich der erforderliche Speicherplatz verfügbar ist. Ein Feldelement benötigt 4 Byte Speicherplatz.

Reset-Verhalten

Das Reset-Verhalten ist abhängig vom Speicherort der $AC_PARAM Variablen:

- Dynamischer Speicher: Initialisierung mit dem Wert "0"

Statischer Speicher: Beibehalten des aktuellen Wertes

3.3.5 Parameter ($AC_PARAM)

Die Variablen $AC_PARAM[<Index>] sind kanalspezifische Felder von Systemvariablen zur Verwendung als allgemeine Zwischenspeicher.

Datentyp: REAL
<Index>: Feldindex: 0, 1, 2, ... (max. Anzahl - 1)

Anzahl pro Kanal

Die maximale Anzahl der $AC_PARAM Variablen pro Kanal ist einstellbar über das Maschinendatum:

MD28254 $MC_MM_NUM_AC_PARAM = <Maximale Anzahl>

Speicherort

Der Speicherort der $AC_PARAM Variablen kann kanalspezifisch festgelegt werden über das Maschinendatum:

MD28255 $MC_MM_BUFFERED_AC_PARAM = <Wert>

Wert	Speicherort
0	Dynamischer Speicher (Standardeinstellung)
1	Statischer Speicher

Hinweis

Datensicherung und Speicherplatz

- Die im statischen Speicher angelegten $AC_PARAM Variablen können kanalspezifisch über die Datensicherung gesichert werden. Datenbaustein: _N_CH<Kanalnummer>_ACP
- Es muss darauf geachtet werden, dass im gewählten Speicherbereich der erforderliche Speicherplatz verfügbar ist. Ein Feldelement benötigt 4 Byte Speicherplatz.

Reset-Verhalten

Das Reset-Verhalten ist abhängig vom Speicherort der $AC_PARAM Variablen:

- Dynamischer Speicher: Initialisierung mit dem Wert "0"
- Statischer Speicher: Beibehalten des aktuellen Wertes

3.3.6 R-Parameter ($R)

Ob R-Parameter als Vor- oder Hauptlaufvariable behandelt werden, hängt davon ab ob sie mit oder ohne $-Zeichen geschrieben werden. Die Schreibweise ist prinzipiell frei wählbar. Für die Verwendung in Synchronaktionen sollten R-Parameter aber als Hauptlaufvariable, also mit $-Zeichen, verwendet werden:

- $R[<Index>]
- $R<Nummer>

Datentyp: REAL
<Index>: Feldindex: 0, 1, 2, ...
<Nummer>: Nummer des R-Parameters: 0, 1, 2, ...
Die Schreibweisen mit Index oder Nummer sind gleichwertig.

Parametrierbare Anzahl pro Kanal

Die maximale Anzahl von R-Parametern pro Kanal ist einstellbar über das Maschinendatum:

MD28254 $MC_MM_NUM_AC_PARAM = <Maximale Anzahl>

Reset-Verhalten

R-Parameter werden persistent im statischen Speicher der NC gespeichert. Daher behalten R-Parameter über alle Reset-Arten hinaus ihre Werte bei:

- Power On-Reset
- NC-Reset
- Teileprogrammende-Reset

Beispiel

Wertzuweisung an R10 im Aktionsteil der Synchronaktion und anschließende Auswertung im Teileprogramm

Programmcode	**Kommentar**
`WHEN $A_IN[1]==1 DO $R[10]=$AA_IM[Y]`	`; Zuweisung`
`G1 X100 F150`	
`STOPRE`	
`IF R[10] > 50 ...`	`; Auswertung im Teileprogramm`

3.3.7 Maschinen- und Settingdaten ($$M, $$S)

MD und SD lesen und schreiben

Bei der Verwendung von Maschinen- und Settingdaten in Synchronaktionen muss unterschieden werden, ob diese während der Bearbeitung der Synchronaktion unverändert bleiben, oder durch parallele Prozesse verändert werden.

Daten die **unverändert** bleiben, darf die NC bereits im **Vorlauf** lesen bzw. schreiben.

Daten die **verändert** werden, darf die NC erst im **Hauptlauf** lesen bzw. schreiben.

Datenzugriff im Vorlauf

Maschinen- und Settingdaten die in Synchronaktionen bereits im Vorlauf gelesen bzw. geschrieben werden dürfen, werden mit den gleichen Bezeichnern wie im Teileprogramm programmiert: $M... bzw. $S...

Programmcode

```
; Die Umkehrposition der Z-Achse $SA_OSCILL_REVERSE_POS2[Z]
; bleibt über den gesamten Bearbeitungszeitraum unverändert
ID=2 WHENEVER $AA_IM[z] < $SA_OSCILL_REVERSE_POS2[Z]-6 DO $AA_OVR[X]=0
```

Datenzugriff im Hauptlauf

Für Maschinen- und Settingdaten die in Synchronaktionen erst im Hauptlauf gelesen bzw. geschrieben werden sollen, wird dem Bezeichner ein zusätzliches Zeichen "$" vorangestellt: $$M… bzw. $$S…

Programmcode

```
; Die Umkehrposition der Z-Achse $SA_OSCILL_REVERSE_POS2[Z]
; kann durch Bedienhandlung zu jedem Zeitpunkt verändert werden
ID=1 WHENEVER $AA_IM[z] < $$SA_OSCILL_REVERSE_POS2[Z] DO $AA_OVR[X] = 0
```

Schreiben im Hauptlauf

Folgende Voraussetzungen müssen für das Schreiben im Hauptlauf gegeben sein:

- Das zum Zeitpunkt des Schreibens vorliegende Zugriffsrecht muss zum Schreiben berechtigen.
- Das Maschinen- bzw. Settingdatum muss die Eigenschaft "sofort wirksam" haben.

Programmcode

```
; Die Schaltposition der SW-Nocken $SN_SW_CAM_... darf, abhängig
; vom aktuellen Sollwert der X-Achse im WKS $AA_IW[X], erst im
; Hauptlauf geschrieben werden
ID=2 WHEN $AA_IW[X] > 10 DO $$SN_SW_CAM_PLUS_POS_TAB_1[0] = 20
                            $$SN_SW_CAM_MINUS_POS_TAB_1[0] = 20
```

Eine vollständige Übersicht der Eigenschaften der Maschinen- und Settingdaten findet sich in:

Literatur

- Listenhandbuch: Listen (Buch 1)
- Listenhandbuch: Ausführliche Maschinendatenbeschreibung

3.3.8 Timer ($AC_TIMER)

Die Variablen $AC_TIMER[<Index>] sind kanalspezifische Felder von Systemvariablen.

Datentyp:	REAL
<Index>:	Feldindex: 0, 1, 2, ... (max. Anzahl - 1)
Einheit:	Sekunde

Anzahl pro Kanal

Die maximale Anzahl der $AC_TIMER Variablen pro Kanal ist einstellbar über das Maschinendatum:

MD28258 $MC_MM_NUM_AC_TIMER = <Maximale Anzahl>

Funktion

Starten

Das Starten eines Timers erfolgt durch Zuweisung eines Wertes ≥ 0:

$AC_TIMER[<Index>] = <Startwert>; mit Startwert ≥ 0

Inkrementieren

Pro Interpolatortakt wird der Wert des Timers um die Dauer eingestellten Interpolatortaktes (MD10071 IPO_CYCLE_TIME) inkrementiert.

$AC_TIMER[<Index>] += <Interpolatortakt>

Stoppen

Das Stoppen eines Timers erfolgt durch Zuweisung eines Wertes < 0:

$AC_TIMER[<Index>] = <Abschaltwert>; mit Abschaltwert < 0

Bei der Zuweisung eines Abschaltwerts, wird nur das weitere Inkrementieren des Timers gestoppt. Der Abschaltwert wird dabei nicht zugewiesen. Nach dem Stoppen des Timers bleibt der letzte gültige Wert erhalten und kann weiterhin gelesen werden.

Hinweis

Der aktuelle Wert eines Timers kann bei laufendem und gestopptem Timer gelesen werden.

Beispiel

Ausgabe des Istwerts der X-Achse als Spannungswert über den Analogausgang $A_OUTA[3], 500 ms nach dem Erkennen des digitalen Eingangs $A_IN[1]:

```
Programmcode                                    Kommentar
WHEN $A_IN[1]==1 DO $AC_TIMER[1]=0              ; Timer starten, Anfangswert 0
WHEN $AC_TIMER[1]>=0.5 DO $A_OUTA[3]=$AA_IM[X] $AC_TIMER[1]=-1
```

3.3.9 FIFO-Variablen ($AC_FIFO)

Über $AC_FIFO Variablen werden innerhalb der R-Parameter spezielle von der NC verwaltete Datenstrukturen bereitgestellt. Diese sind als Ringpuffer organisiert, die nach dem FIFO-Prinzip (First In, First Out) arbeiten.

Syntax

```
$AC_FIFO<Nummer>[<Index>]
$AC_FIFO[<Nummer>, <Index>]
```

Datentyp: Entsprechend R-Parameter: REAL
<Nummer>: Nummer der $AC_FIFO-Variablen: 1, 2, 3, ... max. Anzahl
<Index>: Feldindex: 0, 1, 2, ... (max. Anzahl - 1)

Bedeutung der Feldindizes

Eine $AC_FIFO Variable umfasst neben den Feldelementen für die Anwenderdaten auch mehrere Feldelementen zur Verwaltung der Daten. Über den Index kann auf jedes einzelne Feldelement zugegriffen werden.

Die Feldelemente mit den Indizes 0 … 5 dienen der Verwaltung der $AC_FIFO Variable:

Index	Bedeutung
0	Index 0 hat folgende Sonderbedeutung: Mit dem Index 0 wird nicht auf das Feldelement 0 zugegriffen. Schreiben: der "jüngste" Wert wird in der Variablen abgelegt Lesen: der "älteste" Wert wird aus der Variablen gelesen
1	Schreiben/Lesen: das "älteste" Feldelement wird angesprochen
2	Schreiben/Lesen: das "jüngste" Feldelement wird angesprochen
3	Lesen: liefert die Summe der Werte aller Anwenderdaten Voraussetzung: siehe unten Absatz "Summenbildung über alle Anwenderdaten"
4	Lesen: liefert die Anzahl der vorhandenen Daten Das Rücksetzen einer $AC_FIFO-Variablen auf den Ausgangszustand erfolgt mit: `$AC_FIFO<Nummer>[4] = 0`
5	Lesen: liefert den aktuellen Schreibindex, relativ zum Anfang der $AC_FIFO Variablen

Die Feldelemente ab Index 6 beinhalten die Anwenderdaten:

Index	Bedeutung
6	Schreiben/Lesen: das 1. Feldelement für Anwenderdaten wird angesprochen
7	Schreiben/Lesen: das 2. Feldelement für Anwenderdaten wird angesprochen
n	Schreiben/Lesen: das n. Feldelement für Anwenderdaten wird angesprochen

Hinweis

Überschreiben von Anwenderdaten

Aufgrund der Ringpufferstruktur werden die ältesten Anwenderdaten überschrieben, sobald alle freien Feldelemente einer $AC_FIFO Variablen belegt sind.

Projektierung

Anzahl pro Kanal

Anzahl Feldelemente

Die maximale Anzahl von Feldelementen pro $AC_FIFO Variable ist einstellbar über das Maschinendatum:

MD28264 $MC_LEN_AC_FIFO = <Maximale Anzahl von Feldelementen>

Anfang des $AC_FIFO Variablenbereichs

Der R-Parameter ab dem der Bereich der $AC_FIFO Variablen beginnen soll ist einstellbar über das Maschinendatum:

MD28262 $MC_START_AC_FIFO = <Nummer des Start-R-Parameters>

R-Parameter oberhalb des Start-R-Parameters können im Teileprogramm nicht geschrieben werden.

Gesamtanzahl von R-Parametern im Kanal

Die Gesamtanzahl von R-Parametern im Kanal ist einstellbar über das Maschinendatum:

MD28050 $MC_MM_NUM_R_PARAM = <Maximale Anzahl>

Die im Maschinendatum eingestellte Anzahl von R-Parametern im Kanal muss mindestens so groß sein, wie die Anzahl der für die $AC_FIFO Variablen benötigten R-Parameter:

<Maximale Anzahl> ≥ MD28262 $MC_START_AC_FIFO + MD28260 $MC_NUM_AC_FIFO * (MD28264 $MC_LEN_AC_FIFO + 6)

Summenbildung über alle Anwenderdaten

Die Summe der Werte aller Anwenderdaten wird über den Index [4] nur bereitgestellt, wenn die Funktion über das Maschinendatum aktiviert wurde:

MD28266 MODE_AC_FIFO, Bit 0 = <Wert>

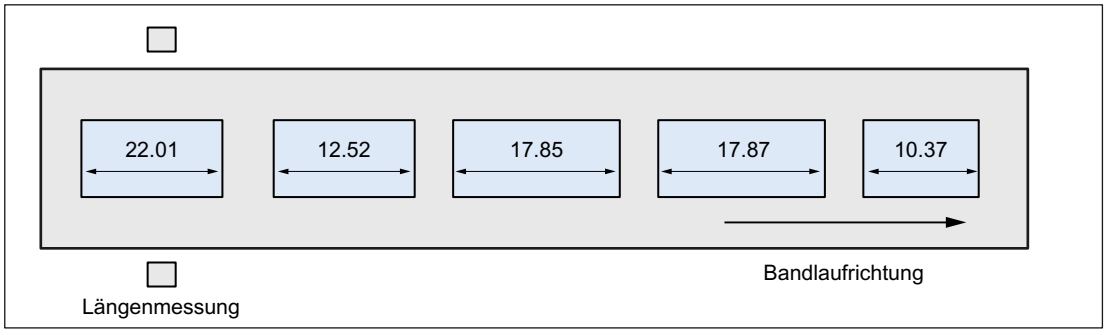
Wert	Bedeutung
0	Die Summe über die Werte aller Anwenderdaten wird **nicht** bereitgestellt
1	Die Summe über die Werte aller Anwenderdaten wird bereitgestellt

Speicherort und Reset-Verhalten

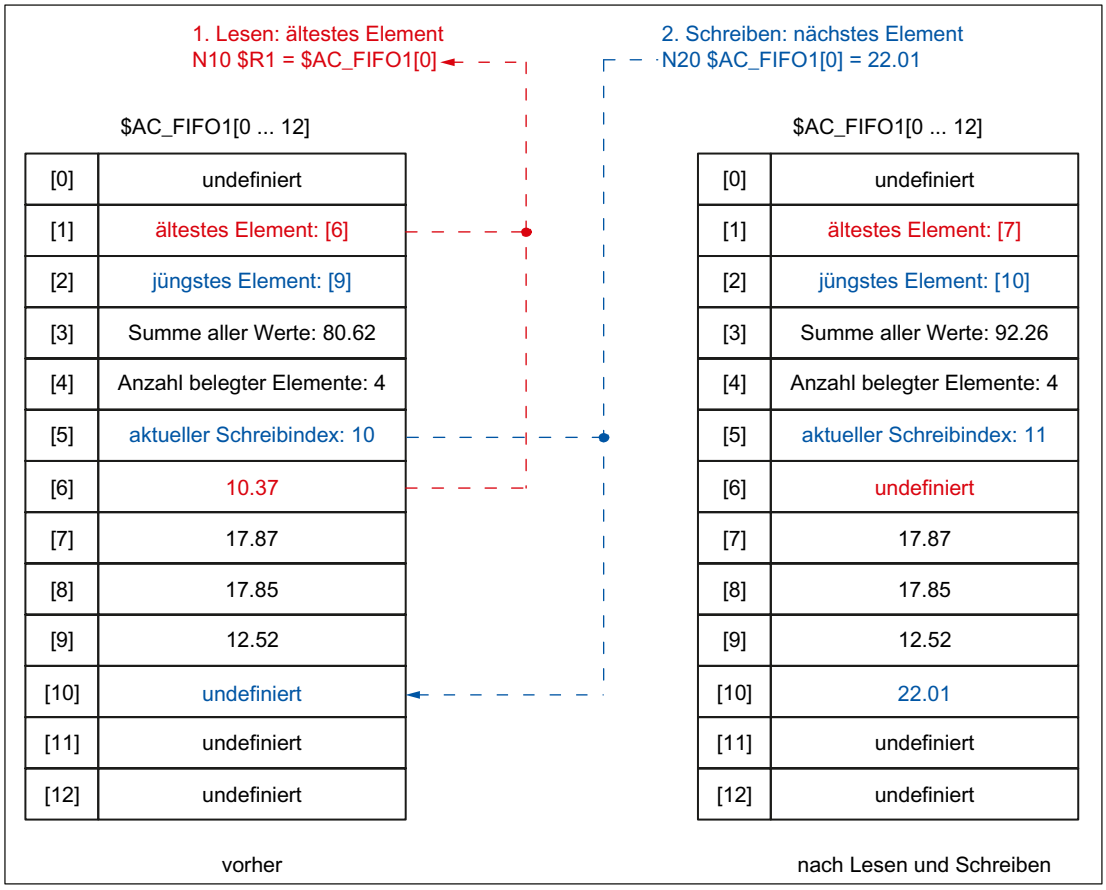
Die $AC_FIFO Variablen basieren auf den R-Parametern. Die dort gemachten Aussagen gelten daher auch für die $AC_FIFO Variablen. Siehe Kapitel "R-Parameter ($R) (Seite 26)".

Beispiel

Serielles Ermitteln der Länge von Werkstücken, die von einem Laufband an einer automatischen Messstation vorbei bewegt werden.



Die Messergebnisse werden über Synchronaktionen in die Systemvariable $AC_FIFO1 geschrieben bzw. aus ihr gelesen.



3.3.10 Bahntangentenwinkel ($AC_TANEB)

Über die kanalspezifische Systemvariable $AC_TANEB (**T**angent **AN**gle at **E**nd of **B**lock) kann der Winkel zwischen der Tangente im Endpunkt des aktuellen Satzes und der Tangente im Anfangspunkt des Folgesatzes gelesen werden.

Datentyp: REAL

Der Tangentenwinkel wird stets positiv im Bereich 0.0° bis 180.0° angegeben.

Kann der Tangentenwinkel nicht ermittelt werden, wird der Wert -180.0° ausgegeben.

Verwendung nur bei programmierten Sätzen

Es wird empfohlen, den Tangentenwinkel nur bei programmierten Sätzen, nicht bei vom System generierten Zwischensätzen, zu lesen. Eine Unterscheidung kann über die Systemvariable $AC_BLOCKTYPE erfolgen:

$AC_BLOCKTYPE == 0 (programmierter Satz)

Beispiel:

```
Programmcode
ID=2 EVERY $AC_BLOCKTYPE==0 DO $R1=$AC_TANEB
```

3.3.11 Override ($A...OVR)

Aktueller Override

Kanalspezifischer Override

Über die kanalspezifisch Systemvariable $AC_OVR kann der Bahnvorschub verändert werden.

Datentyp: REAL
Einheit: %
Wertebereich: 0.0 bis Maschinendatum

- bei **binär**codiertem Korrekturschalter
  MD12100 $MN_OVR_FACTOR_LIMIT_BIN
- bei **gray**codiertem Korrekturschalters
  MD12030 $MN_OVR_FACTOR_FEEDRATE[**30**]

Die Systemvariable $AC_OVR muss in jedem Interpolatortakt neu geschrieben werden, ansonsten wirkt der Wert "100%".

Kanalspezifischer Eilgang-Override

Bei G0-Sätzen (Eilgang) kann der Eilgangvorschub zusätzlich zur Systemvariable $AC_OVR auch über das Settingdatum SD42122 $SC_OVR_RAPID_FACTOR beeinflusst werden.

Voraussetzung: Freigabe der Eilgangbeeinflussung über die Bedienoberfläche.

Achsspezifischer Override

Über die achsspezifische Systemvariable $AA_OVR kann der axiale Vorschub verändert werden:

Datentyp: REAL
Einheit: %
Wertebereich: 0.0 bis Maschinendatum

- bei **binär**codiertem Korrekturschalter
  MD12100 $MN_OVR_FACTOR_LIMIT_BIN
- bei **gray**codiertem Korrekturschalters
  MD12030 $MN_OVR_FACTOR_FEEDRATE[**30**]

Die Systemvariable $AA_OVR muss in jedem Interpolatortakt neu geschrieben werden, ansonsten wirkt der Wert "100%".

PLC-Override

Kanalspezifischer Override

Über die kanalspezifische Systemvariable $AC_PLC_OVR kann der über die Maschinensteuertafel eingestellte kanalspezifisch Override (DB21, ... DBB4) gelesen werden:

Datentyp: REAL
Einheit: %
Wertebereich: 0.0 bis Maximalwert

Achsspezifischer Override

Über die achsspezifische Systemvariable $AA_PLC_OVR kann der über die Maschinensteuertafel eingestellte achsspezifische Override (DB31, ... DBB0) gelesen werden:

Datentyp: REAL
Einheit: %
Wertebereich: 0.0 bis Maximalwert

Wirksamer Override

Wirksamer kanalspezifischer Override

Über die kanalspezifische Systemvariable $AC_TOTAL_OVR kann der wirksame kanalspezifische Override gelesen werden:

Datentyp: REAL
Einheit: %
Wertebereich: 0.0 bis Maximalwert

Wirksamer achsspezifischer Override

Über die achsspezifische Systemvariable $AA_TOTAL_OVR kann der wirksame achssspezifische Override gelesen werden:

Datentyp: REAL
Einheit: %
Wertebereich: 0.0 bis Maximalwert

3.3.12 Auslastungsauswertung ($AN_IPO..., $AN/AC_SYNC..., $AN_SERVO)

Über folgende Systemvariable können die Werte der aktuellen, maximalen und durchschnittlichen Systemauslastung aufgrund von Synchronaktionen gelesen werden:

NC-spezifische Systemvariable	Bedeutung
$AN_IPO_ACT_LOAD	aktuelle Rechenzeit der Interpolatorebene (inkl. Synchronaktionen aller Kanäle)
$AN_IPO_MAX_LOAD	längste Rechenzeit der Interpolatorebene (inkl. Synchronaktionen aller Kanäle)
$AN_IPO_MIN_LOAD	kürzeste Rechenzeit der Interpolatorebene (inkl. Synchronaktionen aller Kanäle)
$AN_IPO_LOAD_PERCENT	aktuelle Rechenzeit der Interpolatorebene im Verhältnis zum Interpolator-Takt (%).
$AN_SYNC_ACT_LOAD	aktuelle Rechenzeit für Synchronaktionen über alle Kanäle
$AN_SYNC_MAX_LOAD	längste Rechenzeit für Synchronaktionen über alle Kanäle
$AN_SYNC_TO_IPO	prozentualer Anteil der Synchronaktionen an der gesamten Rechenzeit der Interpolatorebene (über alle Kanäle)
$AN_SERVO_ACT_LOAD	aktuelle Rechenzeit des Lagereglers
$AN_SERVO_MAX_LOAD	längste Rechenzeit des Lagereglers
$AN_SERVO_MIN_LOAD	kürzeste Rechenzeit des Lagereglers

Kanalspezifische Systemvariable	Bedeutung
$AC_SYNC_ACT_LOAD	aktuelle Rechenzeit für Synchronaktionen im Kanal
$AC_SYNC_MAX_LOAD	längste Rechenzeit für Synchronaktionen im Kanal
$AC_SYNC_AVERAGE_LOAD	durchschnittliche Rechenzeit für Synchronaktionen im Kanal

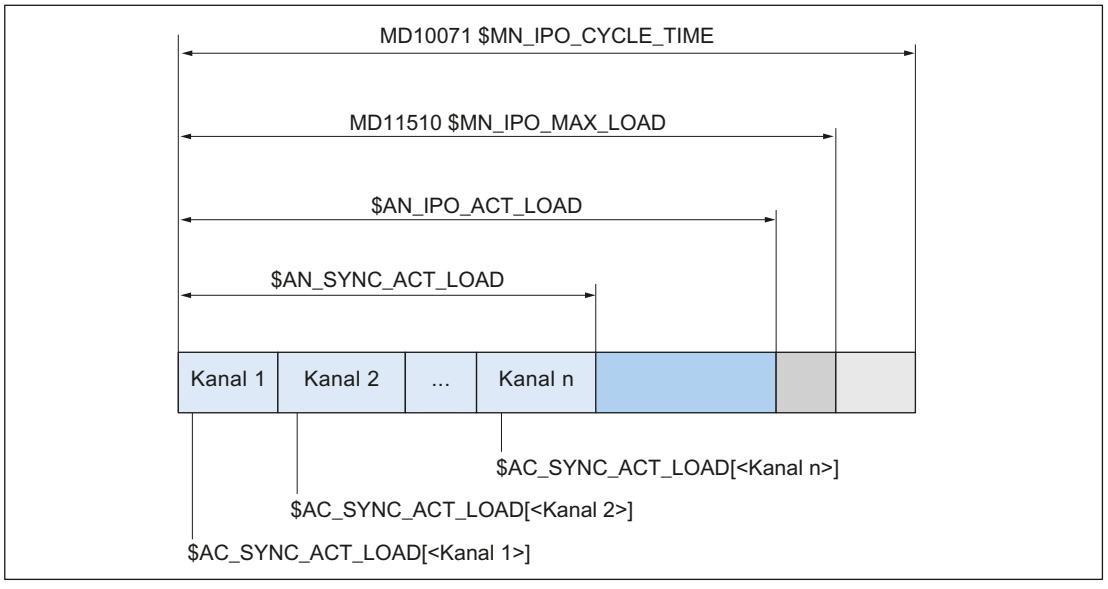


Bild 3-1 Rechenzeitanteile der Synchronaktionen am Interpolatortakt

Aktivierung

Die Systemvariablen enthalten nur gültig Werte, wenn die Diagnosefunktion "Auslastungsauswertung über Synchronaktionen" aktiv ist.

Dazu muss das folgende Maschinendatum größer Null gesetzt werden:

MD11510 $MN_IPO_MAX_LOAD **> 0** (Maximal erlaubte Interpolator-Auslastung)

Bei aktiver Funktion wird im Bedienbereich "Diagnose" > "Systemauslastung" die Zeile "Zeitbedarf der Synchronaktionen" mit den aktuellen Werten angezeigt.

Hinweis

Die Systemvariablen enthalten immer die Werte des vorhergehenden Interpolatortakts.

Überlastgrenze

Mit dem über MD11510 $MN_IPO_MAX_LOAD eingestellten Wert wird eine Überlastgrenze festgelegt:

MD11510 $MN_IPO_MAX_LOAD = <maximale erlaubte Auslastung in % vom Interpolatortakt>

Wird der im Maschinendatum eingestellte Wert überschritten, wird die folgende Systemvariable gesetzt:

$AN_IPO_LOAD_LIMIT = TRUE

Wird der eingestellte Wert wieder unterschritten, wird die Systemvariable wieder zurückgesetzt:

$AN_IPO_LOAD_LIMIT = FALSE

Anwendung

Über die Systemvariable $AN_IPO_LOAD_LIMIT kann eine anwenderspezifische Strategie zur Vermeidung eines Ebenenüberlaufs realisiert werden.

Rücksetzen von Min/Max-Werten

Durch Schreiben von beliebigen Werten werden folgende Systemvariable für Min/Max-Werte wieder zurückgesetzt:

Systemvariable	Bedeutung
$AN_SERVO_MAX_LOAD	längste Rechenzeit des Lagereglers
$AN_SERVO_MIN_LOAD	kürzeste Rechenzeit des Lagereglers
$AN_IPO_MAX_LOAD	längste Rechenzeit der Interpolatorebene (inkl. Synchronaktionen aller Kanäle)
$AN_IPO_MIN_LOAD	kürzeste Rechenzeit der Interpolatorebene (inkl. Synchronaktionen aller Kanäle)
$AN_SYNC_MAX_LOAD	längste Rechenzeit für Synchronaktionen über alle Kanäle
$AC_SYNC_MAX_LOAD	längste Rechenzeit für Synchronaktionen im Kanal

Beispiel

```
Programmcode                                                    Kommentar
$MN_IPO_MAX_LOAD=80                                             ; Überlastgrenze
;
; Initialisierung der Min/Max-Werte
N01 $AN_SERVO_MAX_LOAD=0
N02 $AN_SERVO_MIN_LOAD=0
N03 $AN_IPO_MAX_LOAD=0
N04 $AN_IPO_MIN_LOAD=0
N05 $AN_SYNC_MAX_LOAD=0
N06 $AC_SYNC_MAX_LOAD=0
;
; Alarm 63111 bei Überschreitung der Überlastgrenze
N10 IDS=1 WHENEVER $AN_IPO_LOAD_LIMIT==TRUE DO M4711 SETAL(63111)
;
; Alarm 63222 bei Überschreitung des Rechenzeitanteils
; der Synchronaktionen über alle Kanäle von 30% des Interpolatortakts
N20 IDS=2 WHENEVER $AN_SYNC_TO_IPO > 30 DO SETAL(63222)
;
N30 G0 X0 Y0 Z0
...
N999 M30
```

3.3.13 Arbeitsfeldbegrenzung ($SA_WORKAREA_...)

Für die in Synchronaktionen verfahrbaren Kommandoachsen wirkt bezüglich der programmierbaren Arbeitsfeldbegrenzung `G25` / `G26` nur die Aktivierung über die Settingdaten:

- $SA_WORKAREA_PLUS_ENABLE
- $SA_WORKAREA_MINUS_ENABLE

Ein Ein/Ausschalten der Arbeitsfeldbegrenzung über die Befehle `WALIMON` / `WALIMOF` im Teileprogramm wirkt nicht auf die über Synchronaktionen verfahrenen Kommandoachsen.

3.3.14 SW-Nockenpositionen und -zeiten ($$SN_SW_CAM_...)

Über folgende Settingdaten können die Werte der SW-Nockenpositionen und -zeiten gelesen und geschrieben werden:

NC-spezifisches Settingdatum	Bedeutung
$SN_SW_CAM_MINUS_POS_TAB_1[0..7]	Positionen Minusnocken
$SN_SW_CAM_MINUS_POS_TAB_2[0..7]	Positionen Minusnocken
$SN_SW_CAM_PLUS_POS_TAB_1[0..7]	Positionen Plusnocken

NC-spezifisches Settingdatum	Bedeutung
$SN_SW_CAM_PLUS_POS_TAB_2[0..7]	Positionen Plusnocken
$SN_SW_CAM_MINUS_TIME_TAB_1[0..7]	Vorhalte- bzw. Verzögerungszeit Minusnocken
$SN_SW_CAM_MINUS_TIME_TAB_2[0..7]	Vorhalte- bzw. Verzögerungszeit Minusnocken
$SN_SW_CAM_PLUS_TIME_TAB_1[0..7]	Vorhalte- bzw. Verzögerungszeit Plusnocken
$SN_SW_CAM_PLUS_TIME_TAB_2[0..7]	Vorhalte- bzw. Verzögerungszeit Plusnocken

Hinweis

Das Setzen eines Softwarenockens über Synchronaktionen darf nicht unmittelbar vor Erreichen des Nockens geschehen. Bis zum Erreichen des Nockens müssen mindestens noch 3 Interpolationstakte zur Verfügung stehen.

Eine ausführliche Beschreibung der Funktion "Softwarenocken" findet sich in:

Literatur

Funktionshandbuch Erweiterungsfunktionen; Softwarenocken, Wegschaltsignale (N3)

Beispiele

Programmcode

```
; Veränderung einer Nockenposition
ID=1 WHEN $AA_IW[x] > 0 DO $$SN_SW_CAM_MINUS_POS_TAB_1[0] = 50.0
...
; Veränderung einer Vorhaltezeit
ID=1 WHEN $AA_IW[x] > 0 DO $$SN_SW_CAM_MINUS_TIME_TAB_1[0] = 1.0
```

Siehe auch

Maschinen- und Settingdaten ($$M, $$S) (Seite 27)

3.3.15 Weglängenauswertung/Maschinenwartung ($AA_TRAVEL..., $AA_JERK...)

Über die unten aufgeführten Systemvariablen können die Daten der Weglängenauswertung, z.B. zur Maschinenwartung, gelesen werden.

Aktivierung

Die Aktivierung zur Aufzeichnung der Daten der Weglängenauswertung erfolgt über:

MD18860 $MN_MM_MAINTENANCE_MON = 1

Über das folgende achsspezifische Maschinendatum können die Daten angewählt werden, die für die spezifische Achse aufgezeichnet werden:

MD33060 $MA_MAINTENANCE_DATA[<Achse>], Bit n = 1

Bit	Bedeutung
0	Aufzeichnung von Gesamtverfahrstrecke, Gesamtverfahrzeit und Anzahl der Verfahrvorgänge der Achse
1	Aufzeichnung von Gesamtverfahrstrecke, Gesamtverfahrzeit und Anzahl der Verfahrvorgänge bei großer Geschwindigkeit der Achse
2	Aufzeichnung der gesamten Summe des Rucks der Achse, der Zeit in der die Achse mit Ruck verfahren wird, und der Anzahl der Verfahrvorgänge mit Ruck.

Systemvariable

Systemvariable	Bedeutung	n
$AA_TRAVEL_DIST	Gesamtverfahrweg: Summe aller Sollpositionsänderungen im MKS in [mm] bzw. [Grad].	0
$AA_TRAVEL_TIME	Gesamtverfahrzeit: Summe der IPO-Takte von Sollpositionsänderungen im MKS in [s] (Auflösung: 1 IPO-Takt)	
$AA_TRAVEL_COUNT	Gesamtanzahl der Verfahrvorgänge: Ein Verfahrvorgang einer Maschinenachse ist durch folgende Zustandsfolge bezogen auf die Sollposition gekennzeichnet: Stillstand > Verfahren > Stillstand	
$AA_TRAVEL_DIST_HS	Gesamtverfahrweg bei großer Achsgeschwindigkeit [1)]	1
$AA_TRAVEL_TIME_HS	Gesamtverfahrzeit bei großer Achsgeschwindigkeit [1)]	
$AA_TRAVEL_COUNT_HS	Gesamtanzahl der Verfahrvorgänge bei großer Achsgeschwindigkeit [3)]	
$AA_JERK_TOT	Gesamtsumme des Rucks der Achse: Summe aller Sollwerte des Rucks in [m/s^3] bzw. [Grad/ s^3].	2
$AA_JERK_TIME	Gesamtverfahrzeit mit Ruck: Summe der IPO-Takte von Sollwertänderungen des Rucks in [s] (Auflösung: 1 IPO-Takt)	
$AA_JERK_COUNT	Gesamtanzahl der Verfahrvorgänge mit Ruck	
[1)] Istgeschwindigkeit der Maschinenachse ≥ 80% der parametrierten maximalen Achsgeschwindigkeit (MD32000 MAX_AX_VELO)		

Literatur

Eine ausführliche Beschreibung der Funktion findet sich in:

Funktionshandbuch Sonderfunktionen, Kapitel "Weglängenauswertung (W6)

3.3.16 Polynomkoeffizienten, -parameter ($AC_FCT...)

Funktion

Über die Funktion FCTDEF können Polynome maximal 3. Grades definiert werden:

$f(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + a_3 \cdot x^3$

Hinweis

Die Definition muss in einem **Teileprogramm** erfolgen.

Syntax

`FCTDEF(<Poly_Nr>,<Lo_Limit>,<Up_Limit>,a0,a1,a2,a3)`

Bedeutung

Parameter	Bedeutung
`<Poly_Nr>`:	Nummer der Polynomfunktion
`<Lo_Limit>`:	Untergrenze der Funktionswerte
`<Up_Limit>`:	Obergrenze der Funktionswerte
`a0, a1, a2, a3`:	Polynomkoeffizienten

Hinweis

Nicht benötigte Polynomkoeffizienten (a_2, a_3) können bei der Programmierung der Funktion `FCTDEF(...)` weggelassen werden.

Systemvariable

Aus Synchronaktionen heraus kann auf die Polynomkoeffizienten und -parameter über folgende Systemvariable lesend und schreibend zugegriffen werden:

Systemvariable	Bedeutung
`$AC_FCTLL[<Poly_Nr>]`:	Untergrenze für Funktionswert
`$AC_FCTUL[<Poly_Nr>]`:	Obergrenze für Funktionswert
`$AC_FCT0[<Poly_Nr>]`:	a_0
`$AC_FCT1[<Poly_Nr>]`:	a_1
`$AC_FCT2[<Poly_Nr>]`:	a_2
`$AC_FCT3[<Poly_Nr>]`:	a_3
`<Poly_Nr>`:	Die bei der Definition angegebene Nummer der Polynomfunktion (siehe oben: Syntax)

Teileprogramm

Beim Schreiben der Systemvariablen im Teileprogramm, muss für ein satzsynchrones Schreiben explizit Vorlaufstopp `STOPRE` programmiert werden.

Hinweis

Satzsynchrones Schreiben im Teileprogramm

Damit im Teileprogramm die Systemvariablen satzsynchron geschrieben werden, ist nach dem Schreiben der Systemvariablen der Befehl `STOPRE` (Vorlaufstopp) zu verwenden.

Synchroaktion

Beim Schreiben der Systemvariablen in Synchronaktionen sind diese sofort wirksam.

Verwendung

Der Funktionswert f(x) des Polynoms kann in Synchronaktionen als Eingangswert für z.B. folgende Funktionen verwendet werden:

- "Polynomauswertung (SYNFCT) (Seite 61)"
- "Online-Werkzeugkorrektur (FTOC) (Seite 67)"

Beispiel: Lineare Abhängigkeit

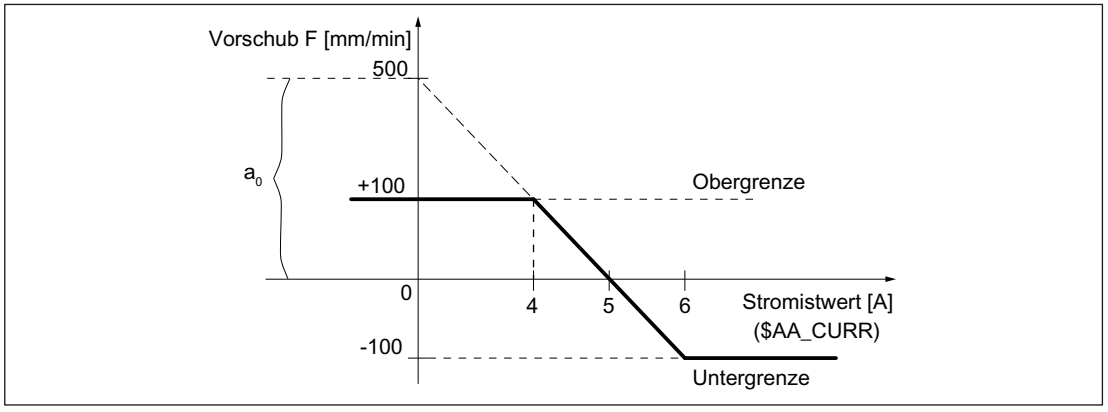


Bild 3-2 Beispiel lineare Abhängigkeit

Parameter	Bedeutung
`<Poly_Nr>`:	Nummer des Polynoms z.B. = **1**
`<Lo_Limit>`:	Untergrenze der Funktionswerte = **-100**
`<Up_Limit>`:	Obergrenze der Funktionswerte = **100**
a_0:	Achsenabschnitt auf der Ordinate (Vorschub): (5 - 4) / 100 = 5 / a_0 a_0 = 100 * 5 / (5 - 4) = **500**

Parameter	Bedeutung
a_1:	Steigung der Geraden: a_1 = 100 / (4 - 5) = **-100**
a_2:	= **0** (kein quadratischer Anteil)
a_3:	= **0** (kein kubischer Anteil)

Programmcode

```
FCTDEF(1, -100, 100, 500, -100, 0, 0)
; oder in verkürzter Schreibweise ohne Parameter a2 und a3
FCTDEF(1, -100, 100, 500, -100)
```

3.3.17 Überlagerte Bewegungen ($AA_OFF)

Überlagerte Bewegungen

Über die Systemvariable $AA_OFF kann ein Positionsoffset in einer Kanalachse vorgegeben werden, der sofort herausgefahren wird:

$AA_OFF[<Kanalachse>] = <Positionsoffset>

Über das folgende Maschinendatum kann eingestellt werden, ob der Positionsoffset der Systemvariablen zugewiesen oder in ihr aufsummiert (integriert) wird:

MD36750 $MA_AA_OFF_MODE, Bit 0 = <Wert>

<Wert>	Bedeutung
0	Zuweisung: $AA_OFF = <Positionsoffset>
1	Aufsummierung (Integration): $AA_OFF += <Positionsoffset>

Begrenzung der Überlagerungsgeschwindigkeit

Die maximal zulässige Geschwindigkeit, mit welcher der Positionsoffset herausgefahren wird, ist einstellbar über das Maschinendatum:

MD32070 $MA_CORR_VELO (Achsgeschwindigkeit für Überlagerung)

Axiale Ruckbegrenzung

Durch Setzen des folgenden Maschinendatums kann für die $AA_OFF-Überlagerung eine axiale Ruckbegrenzung aktiviert werden:

MD32420 $MA_JOG_AND_POS_JERK_ENABLE (Grundeinstellung der axialen Ruckbegrenzung) = **1**

Der axiale Ruck wird auf den in MD32430 $MA_JOG_AND_POS_MAX_JERK (Axialer Ruck) eingestellten Wert begrenzt.

Hinweis

Bei der überlagerten $AA_OFF-Bewegung kann keine vorausschauende Geschwindigkeitsführung erfolgen. Dies kann besonders bei taktweiser Vorgabe (über Synchronaktionen) von $AA_OFF-Überlagerungswerten zu einem unstetigen Geschwindigkeitsverlauf führen. Es wird in solchen Fällen empfohlen, nach Möglichkeit die Ruckbegrenzung zu deaktivieren.

Obergrenze des Korrekturwerts

Über das folgende Settingdatum kann der Wert von $AA_OFF begrenzt werden:

SD43350 $SA_AA_OFF_LIMIT (Obergrenze des Korrekturwerts $AA_OFF bei Abstandsregelung)

Der Status der Begrenzung kann über folgende Systemvariable gelesen werden:

$AA_OFF_LIMIT[<Achse>] == <Wert>

Wert	Bedeutung
-1	Begrenzung des Korrekturwerts in negativer Richtung erfolgt.
1	Begrenzung des Korrekturwerts in positiver Richtung erfolgt.
0	Keine Begrenzung des Korrekturwerts

Reset-Verhalten

Bei statischen Synchronaktionen (`IDS = <Nummer> DO $AA_OFF = <Wert>`) führt das Abwählen des in $AA_OFF wirkenden Positionsoffsets zu einer sofortigen erneuten überlagerten Bewegung. Das Reset-Verhalten bezüglich $AA_OFF kann daher über folgendes Maschinendatum eingestellt werden:

MD36750 $MA_AA_OFF_MODE, Bit 1 = <Wert>

<Wert>	Bedeutung
0	Der Positionsoffset in $AA_OFF wird bei RESET abgewählt
1	Der Positionsoffset in $AA_OFF bleibt über RESET hinaus erhalten

Betriebsart JOG

Das Ausführen einer überlagerten Bewegung aufgrund von $AA_OFF kann auch für die Betriebsart JOG freigegeben werden:

MD36750 $MA_AA_OFF_MODE, Bit 2 = <Wert>

<Wert>	Bedeutung
0	Betriebsart JOG: überlagerte Bewegung aufgrund von $AA_OFF gesperrt
1	Betriebsart JOG: überlagerte Bewegung aufgrund von $AA_OFF freigegeben

Ein Betriebsartenwechsel in die Betriebsart JOG darf erst erfolgen, wenn der aktuelle Positionsoffset herausgefahren wurde. Andernfalls wird folgender Alarm angezeigt:

Alarm "16907 Aktion ... nur im Stopp-Zustand möglich"

Randbedingungen

- Interruptroutinen und ASUP
  Bei Aktivierung einer Interruptroutine bleiben modale Bewegungssynchronaktionen erhalten und sind auch im ASUP wirksam. Erfolgt der Unterprogrammrücksprung nicht mit `REPOS`, so wirken im Hauptprogramm die im asynchronen Unterprogramm geänderten modalen Synchronaktionen weiter.
- `REPOS`
  Im Restsatz gelten die Synchronaktionen wie im Unterbrechungssatz. Änderungen an den modalen Synchronaktionen im ASUP sind im unterbrochenen Programm nicht wirksam. Die mit `FCTDEF` programmierten Polynomkoeffizienten werden durch ASUP und `REPOS` nicht beeinflusst.
  Im ASUP wirken die Polynomkoeffizienten aus dem aufrufenden Programm. Im aufrufenden Programm wirken die Polynomkoeffizienten aus dem ASUP weiter.
- Programmende
  Die mit `FCTDEF` programmierten Polynomkoeffizienten wirken über Programmende hinaus.
- Satzsuchlauf: Aufsammeln der Polynomkoeffizienten
  Bei Satzsuchlauf mit Berechnung werden die Polynomkoeffizienten in den Systemvariablen aufgesammelt.
- Satzsuchlauf: Abwahl von aktiven überlagerten Bewegungen
  Bei Satzsuchlauf werden die Befehle `CORROF` und `DRFOF` aufgesammelt und in einem Aktionssatz ausgegeben. Dabei werden im letzten Satz der `CORROF` oder `DRFOF` enthält, alle abgewählten DRF-Verschiebungen aufgesammelt.
  Die Befehle zur Abwahl von überlagerten Bewegungen `CORROF(<Achse>, "AA_OFF")` werden beim Satzsuchlauf nicht aufgesammelt. Will ein Anwender diesen Suchlauf weiterhin nutzen, so ist dies mit dem Satzsuchlauf via Programmtest "SERUPRO" möglich.
  Literatur:
  Funktionshandbuch Grundfunktionen; BAG, Kanal, Programmbetrieb (K1),
- Abwahl des Positionsoffsets bei aktiven Synchronaktionen
  Ist bei der Abwahl des Positionsoffsets über den Befehl `COROFF(<Achse>, "AA_OFF")` eine Synchronaktion aktiv, wird der Alarm 21660 angezeigt. Gleichzeitig wird $AA_OFF abgewählt und nicht wieder gesetzt. Wird die Synchronaktion später im Satz nach `CORROF` aktiv, bleibt $AA_OFF gesetzt und es wird ein Positionsoffset interpoliert.

Literatur:

Programmierhandbuch Grundlagen

Hinweis

Abhängig davon, in welchem Koordinatensystem eine Hauptlaufvariable definiert ist (BKS oder WKS), werden Frames eingerechnet oder nicht.

Entfernungen werden immer im eingestellten Grundsystem (metrisch oder inch) berechnet. Eine Umschaltung mit `G70` oder `G71` hat keine Auswirkung.

`DRF`-Verschiebungen, externe Nullpunktverschiebungen usw. werden nur bei Hauptlaufvariablen berücksichtigt, die im MKS definiert sind.

3.3.18 Online-Werkzeuglängenkorrektur ($AA_TOFF)

Funktion

In Verbindung mit einer aktiven Orientierungstransformation oder einem aktiven Werkzeugträger können während der Bearbeitung in Echtzeit Werkzeuglängenkorrekturen ausgeführt werden. Die Veränderung der effektiven Werkzeuglänge durch die Online-Werkzeuglängenkorrektur führt bei Orientierungsänderungen zu veränderten Ausgleichsbewegungen der an der Transformation beteiligten Achsen. Die resultierenden Geschwindigkeiten können dabei, abhängig von der Maschinenkinematik und den aktuellen Achspositionen, sowohl größer als auch kleiner werden.

Geschwindigkeit und Beschleunigung, mit der die über Systemvariable $AA_TOFF vorgegebenen Werkzeuglängenkorrekturen herausgefahren werden, können über folgende Maschinendaten vorgegeben werden:

- MD21194 $MC_TOFF_VELO (Geschwindigkeit, Online Korrektur in Werkzeugrichtung)
- MD21196 $MC_TOFF_ACCEL (Beschleunigung, Online Korrektur in Werkzeugrichtung)

Weitere Informationen zur Aktivierung der Funktion siehe:

Literatur:

Programmierhandbuch Arbeitsvorbereitung; Kapitel "Transformationen "TOFFON, TOFFOF""

Anwendungen in Synchronaktionen

In Synchronaktionen können über die Systemvariable $AA_TOFF Werkzeuglängenkorrekturen in allen drei Dimensionen aufgeschaltet werden. Als Index werden die drei Geometrieachsnamen X, Y, Z verwendet. Alle drei Korrekturrichtungen können gleichzeitig aktiv sein.

Die Korrekturen sind bei einer aktiven Orientierungstransformation oder bei einem aktiven, orientierbaren Werkzeugträger in den jeweiligen Werkzeugrichtungen wirksam. Vor dem Ein- bzw. Ausschalten einer Transformation muss eine überlagerte Bewegung mit `TOFFOF` ausgeschaltet werden.

Nach Abwahl der Werkzeuglängenkorrektur in einer Dimension, ist der Wert der Systemvariablen $AA_TOFF in dieser Dimension gleich 0.

Wirkungsweise der Korrektur in Werkzeugrichtung

Die Werkzeuglängenkorrekturen verändern nicht die Werkzeugparameter, sondern sie werden innerhalb der Transformation oder des orientierbaren Werkzeugträgers so verrechnet, dass sich Korrekturen im Werkzeugkoordinatensystem ergeben.

Über das folgende Maschinendatum kann für jede Dimension festgelegt werden, ob die in $AA_TOFF angegebene Werkzeuglängenkorrektur absolut oder inkrementell (integrierend) verrechnet werden soll:

MD21190 $MC_TOFF_MODE (Wirkungsweise der Korrektur in Werkzeugrichtung)

Der aktuelle Wert der Werkzeuglängenkorrektur kann über die Systemvariable $AA_TOFF_VAL gelesen werden.

Hinweis

Eine Auswertung der Variablen $AA_TOFF_VAL ist nur in Verbindung mit einer aktiven Orientierungstransformation oder einem aktiven Werkzeugträger sinnvoll.

Beispiele

Anwahl der Online-Werkzeuglängenkorrektur

Maschinendaten für Online-Werkzeuglängenkorrektur:

- MD21190 $MC_TOFF_MODE = 1
- MD21194 $MC_TOFF_VEL[0] = 10000
- MD21194 $MC_TOFF_VEL[1] = 10000
- MD21194 $MC_TOFF_VEL[2] = 10000
- MD21196 $MC_TOFF_ACC[0] = 1
- MD21196 $MC_TOFF_ACC[1] = 1
- MD21196 $MC_TOFF_ACC[2] = 1

Online-Werkzeuglängenkorrektur im Teileprogramm aktivieren:

Programmcode

```
N5 DEF REAL XOFFSET
; Orientierungstransformation aktivieren
N10 TRAORI
; Werkzeuglängenkorrektur in Z-Richtung aktivieren
N20 TOFFON(Z)
; Werkzeuglängenkorrektur in Z-Richtung: 10 mm
N30 WHEN TRUE DO $AA_TOFF[Z] = 10
G4 F5
...
; Statische Synchronaktion: Werkzeuglängenkorrektur in X-Richtung
; entsprechend der Position der X2-Achse im WKS
N50 ID=1 DO $AA_TOFF[X] = $AA_IW[X2]
```

Programmcode

```
G4 F5
...
; Merke: aktuelle gesamt Werkzeuglängenkorrektur in X-Richtung
N100 XOFFSET = $AA_TOFF_VAL[X]
; die Werkzeuglängenkorrektur in X-Richtung zu 0 zurückfahren
N120 TOFFON(X, -XOFFSET)
G4 F5
```

Abwahl der Online-Werkzeuglängenkorrektur

Programmcode

```
; Orientierungstransformation aktivieren
N10 TRAORI
; Werkzeuglängenkorrektur in X-Richtung aktivieren
N20 TOFFON(X)
; Werkzeuglängenkorrektur in X-Richtung: 10 mm
N30 WHEN TRUE DO $AA_TOFF[X] = 10
G4 F5
...
; Werkzeuglängenkorrektur in X-Richtung löschen
; Es wird keine Achse verfahren. Zur aktuellen Position im WKS wird
; der Positionsoffset entsprechend der aktuellen Orientierung
; hinzugerechnet.
N80 TOFFOF(X)
N90 TRAFOOF
```

Aktivierung und Deaktivierung im Teileprogramm

Die Online-Werkzeuglängenkorrektur wird im Teileprogramm mit `TOFFON` aktiviert und mit `TOFFOF` deaktiviert. Bei der Aktivierung kann für die jeweilige Korrekturrichtung ein Offsetwert, z. B. `TOFFON(Z,25)` angegeben werden, der sofort herausgefahren wird. Der Status der Online-Werkzeuglängenkorrektur wird an der NC/PLC-Nahtstelle über folgende Signale angezeigt:

- DB21, ... DBX318.2 (`TOFF` aktiv)
- DB21, ... DBX318.3 (`TOFF` Bewegung aktiv)

Hinweis

Die Online-Werkzeuglängenkorrektur bleibt solange inaktiv, bis sie über `TOFFON` im Teileprogramm wieder angewählt wird.

Verhalten bei Reset und Power On

Das Verhalten bei Reset kann eingestellt werden über das Maschinendatum:

MD21190 $MC_TOFF_MODE, Bit 0 = <Wert> (Wirkungsweise der Korrektur in Werkzeugrichtung)

Wert	Bedeutung
0	Der Werkzeuglängenoffset $AA_TOFF wird bei Reset abgewählt
1	Der Werkzeuglängenoffset $AA_TOFF bleibt bei Reset erhalten

Dies ist immer bei statischen Synchronaktionen `IDS=<Nummer> DO $AA_TOFF[n]=<Wert>` notwendig, da es sonst erneut zu einer sofortigen Werkzeuglängenkorrektur kommen würde.

Ebenso kann eine Transformation oder ein orientierbarer Werkzeugträger **nach** Reset über folgendes Maschinendatum abgewählt werden:

MD20110 $MC_RESET_MODE_MASK (Grundstellung nach Reset)

Auch hier muss die Werkzeuglängenkorrektur gelöscht werden.

Wenn über Reset hinweg eine Werkzeuglängenkorrektur aktiv bleiben soll, und ein Transformationswechsel oder ein Wechsel des orientierbaren Werkzeugträgers stattfindet, wird der Alarm 21665 "Kanal %1 $AA_TOFF[ ] rückgesetzt" ausgegeben. Die Werkzeuglängenkorrektur wird dabei auf den Wert 0 gesetzt.

Nach Power On werden alle Werkzeuglängenoffsets auf den Wert 0 gesetzt.

Die Funktion ist nach POWER ON deaktiviert.

Verhalten bei Betriebsartenwechsel

Die Werkzeuglängenkorrektur bleibt über Betriebsartenwechsel hinaus aktiv. Die Korrektur wird in allen Betriebsarten außer JOG und REF ausgeführt.

Wird beim Betriebsartenwechsel eine Werkzeuglängenkorrektur aufgrund von $AA_TOFF[ ] herausgefahren, erfolgt die Betriebsartenumschaltung erst, nach dem Herausfahren der Werkzeuglängenkorrektur. Es wird der Alarm 16907 "Kanal %1 Aktion %2 <ALNX> nur im Stopp-Zustand möglich" angezeigt.

Verhalten bei REPOS

Die Werkzeuglängenkorrektur ist in der Betriebsart `REPOS` aktiv.

Randbedingungen

Bei vorhandenem Werkzeuglängenoffset sind folgende Randbedingungen zu beachten:

- Eine Transformation ist mit `TRAFOOF` auszuschalten
- Vor der Aktivierung einer Transformation im Teileprogramm muss ein aktiver Werkzeuglängenoffset mit `TOFFOF` gelöscht werden.
- Beim Umschalten von `CP` nach `PTP` wird eine Transformation abgeschaltet. Ein Werkzeuglängenoffset muss **vor** der Umschaltung gelöscht werden. Ist bei einem Wechsel auf achsspezifisches Handverfahren in der Betriebsart JOG eine Werkzeuglängenkorrektur aktiv, wird der Wechsel nach `PTP` nicht ausgeführt. `CP` bleibt solange aktiv, bis die Werkzeuglängenkorrektur über `TOFFOF` gelöscht wurde.

- Vor einem Achstausch einer Geometrieachse muss ein aktiver Werkzeuglängenoffset in Richtung der Geometrieachse über `TOFFOF` gelöscht werden.
- Vor einem Ebenenwechsel muss ein aktiver Werkzeuglängenoffset über `TOFFOF` gelöscht werden.
- Die `TOFFON` bzw. `TOFFOF` werden bei einem Satzsuchlauf nicht mit aufgesammelt und im Aktionssatz nicht ausgegeben.

3.3.19 Aktueller Satz im Interpolator ($AC_BLOCKTYPE, $AC_BLOCKTYPEINFO, $AC_SPLITBLOCK)

Über die folgenden Systemvariablen können in Synchronaktionen Informationen zum aktuell im Hauptlauf abgearbeiteten Satz gelesen werden.

$AC_BLOCKTYPE und $AC_BLOCKTYPEINFO

Die Systemvariable $AC_BLOCKTYPE enthält den Satztyp bzw. die Kennung für die Funktion, die den Satz erzeugt hat.

Die Systemvariable $AC_BLOCKTYPEINFO enthält neben dem Satztyp (tausender Stelle) die funktionsspezifische Ursache für die Generierung des Zwischensatzes.

$AC_BLOCKTYPE		$AC_BLOCKTYPEINFO	
Wert	**Bedeutung: Aktueller Satz wurde generiert aufgrund ...**	**Wert**	**Bedeutung**
0	Programmierter Satz!	-	-
1	NC als Zwischensatz	1000	Enthält keine weiteren Informationen
2	Fasen / Runden	2001	Gerade
		2002	Kreis
3	Weiches An/Abfahren (WAB)	3001	Anfahren mit Gerade
		3002	Anfahren mit Viertelkreis
		3003	Anfahren mit Halbkreis
4	Werkzeugkorrektur	4001	Anfahrsatz nach STOPRE
		4002	Verbindungssätze bei nicht gefundenem Schnittpunkt
		4003	Punktförmiger Kreis an Innenecken (nur bei TRACYL)
		4004	Umfahrungskreis (bzw. Kegelschnitt) an Außenecken
		4005	Anfahrsätze bei Korrekturunterdrückung
		4006	Anfahrsätze bei erneuter WRK-Aktivierung
		4007	Satzaufspaltung wegen zu hoher Krümmung
		4008	Ausgleichssätze beim 3D-Stirnfräsen (Werkzeug-Vektor parallel zum Flächen-Vektor)
5	Überschleifen	5001	Überschleifkontur durch G641
		5002	Überschleifkontur durch G642
		5003	Überschleifkontur durch G643
		5004	Überschleifkontur durch G644

$AC_BLOCKTYPE		$AC_BLOCKTYPEINFO	
Wert	**Bedeutung: Aktueller Satz wurde generiert aufgrund ...**	**Wert**	**Bedeutung**
6	Tangentiales Nachführen (TLIFT)	6001	Lineare Bewegung der Tangentialachse ohne Abhebebewegung
		6002	Nichtlineare Bewegung der Tangentialachse (Polynom) ohne Abhebebewegung
		6003	Abhebebewegung: Tangentialachs- und Abhebebewegung starten gleichzeitig
		6004	Abhebebewegung: Tangentialachse startet erst, wenn bestimmte Abhebeposition erreicht wird
7	Wegaufteilung	7001	programmierte Wegaufteilung ohne das Stanzen oder Nibbeln aktiv ist
		7002	programmierte Wegaufteilung mit aktivem Stanzen oder Nibbeln
		7003	automatisch intern generierte Wegaufteilung
8	Compile-Zyklus	x	x: ID der Compile-Zyklen-Applikation, die den Satz erzeugt hat
9	Bahnrelative Orientierungsinterpolation (ORIPATH / ORIROTC)	9000	Interpolation der Werkzeugorientierung bei ORIPATH
		9001	Interpolation der Drehung des Werkzeugs bei ORIROTC
10	Polbehandlung bei Orientierungstransformation	10000	Vorausschauende Positionierung der Polachse
		10001	Durchfahren des Polkegels

$AC_SPLITBLOCK

Über die Systemvariable $AC_SPLITBLOCK kann ermittelt werden, ob ein intern generierter Satz oder ein programmierter aber durch die NC verkürzter Satz vorliegt.

$AC_SPLITBLOCK	
Wert	**Bedeutung:**
0	Programmierter Satz. Ein durch den Kompressor generierter Satz wird ebenfalls als programmierter Satz behandelt.
1	Intern generierter Satz oder ein verkürzter Originalsatz
3	Letzter Satz in einer Kette von intern generierten Sätzen oder verkürzten Originalsätzen

Beispiel

Synchronaktionen zum Zählen von Überschleifsätzen.

Über die Abfrage der Systemvariable $AC_TIMEC == 0 (Interpolations-Takte seit Satzanfang) wird sicher gestellt, dass der Satztyp nur einmal am Satzanfang ermittelt wird.

```
Programmcode                    Kommentar
$AC_MARKER[0]=0                 ;   Zähler für alle Überschleifsätze
$AC_MARKER[1]=0                 ;   Zähler für G641-Überschleifsätze
$AC_MARKER[2]=0                 ;   Zähler für G642-Überschleifsätze
...
; Synchronaktion zum Zählen aller Überschleifsätze
```

```
Programmcode                    Kommentar
ID=1 WHENEVER ($AC_TIMEC==0) AND ($AC_BLOCKTYPE==5) DO
     $AC_MARKER[0] = $AC_MARKER[0] + 1
...
; Synchronaktion zum Zählen der G641-Überschleifsätze
ID=2 WHENEVER ($AC_TIMEC==0) AND ($AC_BLOCKTYPEINFO==5001) DO
     $AC_MARKER[1] = $AC_MARKER[1]+1
...
; Synchronaktion zum Zählen der G642-Überschleifsätze
ID=3 WHENEVER ($AC_TIMEC==0) AND ($AC_BLOCKTYPEINFO==5002) DO
     $AC_MARKER[2] = $AC_MARKER[2] + 1
...
```

3.3.20 Initialisierung von Feldvariablen (SET, REP)

Funktion

Über die Befehle `SET` und `REP` können Feldvariable auch in Synchronaktionen initialisiert werden.

Eine ausführliche Beschreibung der Befehle findet sich in:

Literatur

Programmierhandbuch Arbeitsvorbereitung, Kapitel "Flexible NC-Programmierung" > "Variablen" > "Definition und Initialisierung von Feldvariablen (DEF, SET, REP)"

Beispiel

```
Programmcode
PROC MAIN
N10 DEF REAL SYG_IS[3,2]
...
WHEN TRUE DO SYG_IS[0,0]=REP(0.0,3)
WHEN TRUE DO SYG_IS[1,1]=SET(3,4,5)
...
```

Randbedingungen

- Es können nur Feldvariable initialisiert werden, die in Synchronaktionen schreibbar sind.

3.4 Anwenderdefinierte Variablen für Synchronaktionen

Synchronaktionsfähige GUD-Variablen

Neben spezifischen Systemvariablen können in Synchronaktionen auch vordefinierte globale Synchronaktions-Anwendervariablen (Synchronaktions-GUD) verwendet werden. Die Anzahl der dem Anwender zur Verfügung stehenden Synchronaktions-GUD wird Datentyp- und Zugriffs-spezifisch über folgende Maschinendaten parametriert:

- MD18660 $MM_NUM_SYNACT_GUD_**REAL**[<x>] = <Anzahl>
- MD18661 $MM_NUM_SYNACT_GUD_**INT**[<x>] = <Anzahl>
- MD18662 $MM_NUM_SYNACT_GUD_**BOOL**[<x>] = <Anzahl>
- MD18663 $MM_NUM_SYNACT_GUD_**AXIS**[<x>] = <Anzahl>
- MD18664 $MM_NUM_SYNACT_GUD_**CHAR**[<x>] = <Anzahl>
- MD18665 $MM_NUM_SYNACT_GUD_**STRING**[<x>] = <Anzahl>

Über den Index <x> wird der Datenbaustein (Zugriffrechte), über den Wert <Anzahl> die Anzahl von Synchronaktions-GUD des jeweiligen Datentyps (REAL, INT, ...) angegeben. Im jeweiligen Datenbaustein wird daraufhin für jeden Datentyp ein 1-dimensionale Feldvariable mit folgendem Namens-Schema angelegt: SYG_<Datentyp><Zugriffsrecht>[<Index>]:

Index <x>		Datentyp (MD18660 ... MD18665)					
	Baustein	REAL	INT	BOOL	AXIS	CHAR	STRING
0	SGUD	SYG_RS[i]	SYG_IS[i]	SYG_BS[i]	SYG_AS[i]	SYG_CS[i]	SYG_SS[i]
1	MGUD	SYG_RM[i]	SYG_IM[i]	SYG_BM[i]	SYG_AM[i]	SYG_CM[i]	SYG_SM[i]
2	UGUD	SYG_RU[i]	SYG_IU[i]	SYG_BU[i]	SYG_AU[i]	SYG_CU[i]	SYG_SU[i]
3	GUD4	SYG_R4[i]	SYG_I4[i]	SYG_B4[i]	SYG_A4[i]	SYG_C4[i]	SYG_S4[i]
4	GUD5	SYG_R5[i]	SYG_I5[i]	SYG_B5[i]	SYG_A5[i]	SYG_C5[i]	SYG_S5[i]
5	GUD6	SYG_R6[i]	SYG_I6[i]	SYG_B6[i]	SYG_A6[i]	SYG_C6[i]	SYG_S6[i]
6	GUD7	SYG_R7[i]	SYG_I7[i]	SYG_B7[i]	SYG_A7[i]	SYG_C7[i]	SYG_S7[i]
7	GUD8	SYG_R8[i]	SYG_I8[i]	SYG_B8[i]	SYG_A8[i]	SYG_C8[i]	SYG_S8[i]
8	GUD9	SYG_R9[i]	SYG_I9[i]	SYG_B9[i]	SYG_A9[i]	SYG_C9[i]	SYG_S9[i]
mit i = 0 bis (<Anzahl> - 1) Baustein: _N_DEF_DIR/_N_ ... _DEF, z.B.für SGUD ⇒ _N_DEF_DIR/_N_**SGUD**_DEF							

Eigenschaften

Synchronaktions-GUD haben folgende Eigenschaften:

- Synchronaktions-GUD können in Synchronaktionen und Teileprogrammen / Zyklen gelesen und geschrieben werden
- Auf Synchronaktions-GUD kann über BTSS zugegriffen werden
- Synchronaktions-GUD werden auf der Bedienoberfläche des HMI im Bedienbereich "Parameter" angezeigt

- Synchronaktions-GUD können auf HMI im Wizard, in der Variablenansicht und im Variablenprotokoll verwendet werden
- Die Feldgröße bei Synchronaktions-GUD vom Typ STRING ist fest auf 32 (31 Zeichen + \0) definiert.
- Auch wenn manuell keine Definitionsdateien für globale Anwenderdaten (GUD) angelegt wurden, können über Maschinendaten definierte Synchronaktions-GUD im jeweiligen GUD-Baustein von HMI aus gelesen werden.

Hinweis

Anwendervariablen (GUD, PUD, LUD) können nur dann mit dem gleichen Namen wie Synchronaktions-GUD definiert werden (`DEF ... SYG_xy`), wenn keine Synchronaktions-GUD mit gleichem Namen parametriert sind (MD18660 - MD18665). Diese vom Anwender definierten GUD können **nicht** in Synchronaktionen verwendet werden.

Zugriffsrechte

Die in einer GUD-Definitionsdatei definierten Zugriffsrechte bleiben weiterhin gültig und beziehen sich nur auf die in dieser GUD-Definitionsdatei definierten GUD-Variablen.

Löschverhalten

Wird der Inhalt einer bestimmten GUD-Definitionsdatei neu aktiviert, wird zunächst der alte GUD-Datenbaustein im aktiven Filesystem gelöscht. Die projektierten Synchronaktions-GUD werden dabei ebenfalls zurückgesetzt. Dieser Vorgang ist auch über HMI im Bedienbereich "Dienste" > "Anwenderdaten (GUD) definieren und aktivieren" möglich.

3.5 Sprachelemente für Synchronaktionen und Technologiezyklen

Folgende Sprachelemente können in Synchronaktionen und Technologiezyklen verwendet werden:

Feste Adressen	
L	Unterprogrammnummer
F	Vorschub
S [1) 2)]	Spindel
M [1) 2)]	M-Funktion
H [1)]	H-Funktion
1) Kapitel: "Ausgabe von M-, S- und H-Hilfsfunktionen an die PLC (Seite 60)"	
2) Kapitel: "Verfahren von Spindeln(M, S, SPOS) (Seite 85)"	

Feste Adressen mit Achserweiterung: Verschiedenes	
POS	Positionierachse Kapitel: "Verfahren von Achsen, auf Position (POS) (Seite 72)"
POSA	Satzübergreifende Positionierachse

Feste Adressen mit Achserweiterung: Verschiedenes	
SPOS	Spindelpositionierung Kapitel: "Verfahren von Spindeln(M, S, SPOS) (Seite 85)"
MOV [1)]	Positionierachse Kapitel: "Verfahren von Achsen, endlos (MOV) (Seite 77)"
FA	Axialer Vorschub Kapitel: "Axialer Vorschub (FA) (Seite 78)"
OVRA	Axialer Override
ACC	Axiale Beschleunigung
MEASA	Axiales Messen mit Restweglöschen
MEAWA	Axiales Messen ohne Restweglöschen Kapitel: "Messen (MEAWA, MEAC) (Seite 105)"
MEAC	Zyklisches Messen Kapitel: "Messen (MEAWA, MEAC) (Seite 105)"
SCPARA	Parametersatzumschaltung
VELOLIMA	Axiale Geschwindigkeits- / Drehzahlbegrenzung
ACCLIMA	Axiale Beschleunigungsbegrenzung
JERKLIMA	Axiale Ruckbegrenzung
1) nicht in Technologiezyklen zulässig	

Einstellbare Adressen: Fahren auf Festanschlag [1)]	
FXS	Fahren auf Festanschlag einschalten
FXST	Momentgrenze für Fahren auf Festanschlag
FXSW	Überwachungsfenster für Fahren auf Festanschlag
FOCON	Fahren mit begrenztem Moment / Kraft einschalten
FOCOF	Fahren mit begrenztem Moment / Kraft ausschalten
1) Kapitel: "Fahren auf Festanschlag (FXS, FXST, FXSW, FOCON, FOCOF, FOC) (Seite 108)"	

Einstellbare Adressen: Kopplungen > Generische Kopplung [1)]	
CPBC	Satzwechselkriterium bei aktiver Kopplung
CPDEF	Koppelmodul anlegen
CPDEL	Koppelmodul löschen
CPFMOF	Verhalten der Folgeachse beim Ausschalten der Kopplung
CPFMON	Verhalten der Folgeachse beim Einschalten der Kopplung
CPFMSON	Synchronisationsmodus beim Einkoppeln
CPFPOS	Synchronposition der Folgeachse beim Einschalten
CPFRS	Bezugssystem für das Koppelmodul der Folgeachse
CPLCTID	Nummer der Kurventabellen für die Kopplung der Folgeachse
CPLDEF	Definition des Bezugs: Leitachse zu Folgeachse
CPLDEL	Aufheben des Bezugs: Leitachse zu Folgeachse
CPLDEN	Koppelfaktor: Zähler
CPLNUM	Koppelfaktor: Nenner

Einstellbare Adressen: Kopplungen > Generische Kopplung [1]	
CPLDYPRIO	Priorität der Leitachse bei der dynamische Begrenzung
CPLDYVLL	Begrenzung der überlagerten Bewegung der Leitachse: Untergrenze
CPLDYVLU	Begrenzung der überlagerten Bewegung der Leitachse: Obergrenze
CPLINSC	Skalierfaktor für den Eingangswert der Leitachse
CPLINTR	Verschiebewert für den Eingangswert der Leitachse
CPLOF	Kopplung Leitachse zu Folgeachse: Ausschalten
CPLON	Kopplung Leitachse zu Folgeachse: Einschalten
CPLOUTSC	Skalierung des Ausgangswertes
CPLOUTTR	Verschiebung des Ausgangswertes
CPLPOS	Synchronposition der Leitachse beim Einschalten
CPLSETVAL	Kopplungsart der Folgeachse zur Leitachse
CPMALARM	Alarmverhalten definieren
CPMPRT	Startverhalten bei Programmtest definieren
CPMRESET	Reset-Verhalten definieren
CPMSTART	Startverhalten definieren
CPMPVDI	Verhalten bzgl. NC/PLC-Nahtstellensignalen definieren
CPOF	Ausschalten der Kopplung zu allen definierten Leitachsen
CPON	Einschalten der Kopplung zu allen definierten Leitachsen
CPSETTYPE	Grundlegende Kopplungseigenschaften definieren
CPSYNCOP	Positionssynchronlauf "grob"
CPSYNCOP2	Positionssynchronlauf 2 "grob"
CPSYNFIP	Positionssynchronlauf "fein"
CPSYNFIP2	Positionssynchronlauf 2 "fein"
CPSYNCOV	Geschwindigkeitssynchronlauf"grob"
CPSYNFIV	Geschwindigkeitssynchronlauf"fein"
1) Kapitel: "Kopplungen (CP..., LEAD..., TRAIL..., CTAB...) (Seite 101)"	

G-Funktionen: Maßsystem einstellen [1]	
G70	Maßsystem inch
G71	Maßsystem metrisch
G700	Maßsystem inch
G710	Maßsystem metrisch
1) Kapitel: "Maßsystem einstellen (G70, G71, G700, G710) (Seite 75)"	

Vordefinierte Unterprogramme: Verschiedenes	
POLFA	Axiale Rückzugsposition für Einzelachse
POLFC	Axiale Rückzugsposition für Kanalachsen
STOPREOF	Vorlaufstopp aufheben Kapitel:"Vorlaufstopp aufheben (STOPREOF) (Seite 70)"
RDISABLE	Einlesesperre Programmierte Einlesesperre (RDISABLE) (Seite 69)"

Vordefinierte Unterprogramme: Verschiedenes	
DELDTG	Restweglöschen Kapitel:"Restweglöschen (DELDTG) (Seite 70)"
LOCK	Synchronaktion sperren
UNLOCK	Synchronaktion freigeben
RESET	Technologiezyklus rücksetzen
ICYCON	Technologiezyklus: einen Satz pro Interpolatortakt
ICYCOF	Technologiezyklus: alle Sätze in einem Interpolatortakt
SYNFCT	Polynomfunktion auswerten Kapitel:"Polynomauswertung (SYNFCT) (Seite 61)"
FTOC	Werkzeugfeinkorrektur Kapitel:"Online-Werkzeugkorrektur (FTOC) (Seite 67)"
SOFTENDSA	Softwareendschalter
PROTA	Status eines Schutzbereichs ändern
SETM	Marke der Kanalkoordinierung setzen Kapitel:"Kanalsynchronisation (SETM, CLEARM) (Seite 110)"
CLEARM	Marke der Kanalkoordinierung löschen Kapitel:"Kanalsynchronisation (SETM, CLEARM) (Seite 110)"
RET	Unterprogrammrücksprung
GET	Achse anfordern Kapitel:"Achstausch (GET, RELEASE, AXTOCHAN) (Seite 79)"
RELEASE	Achse freigeben Kapitel:"Achstausch (GET, RELEASE, AXTOCHAN) (Seite 79)"
AXTOCHAN	Achse an einen anderen Kanal übergeben Kapitel:"Achstausch (GET, RELEASE, AXTOCHAN) (Seite 79)"
AXCTSWEC	Rücknahme der Freigabe zur Achscontainer-Drehung Kapitel:"Freigabe für Achscontainer-Drehung zurücknehmen (AXCTSWEC) (Seite 86)"
SETAL	Anwenderspezifischen Alarm anzeigen Kapitel:"Anwenderspezifische Fehlerreaktionen (SETAL) (Seite 111)"
IPOBRKA	Satzwechselkriterium: Bremsrampe
ADISPOSA	Toleranzfenster zum Bewegungsendekriterium

Vordefinierte Unterprogramme: Kopplung > Mitschleppen [1)]	
TRAILON	Mitschleppen ein
TRAILOF	Mitschleppen aus
1) Kapitel:"Kopplungen (CP..., LEAD..., TRAIL..., CTAB...) (Seite 101)"	

Vordefinierte Unterprogramme: Kopplungen > Leitwertkopplung [1)]	
LEADON	Leitewertkopplung ein
LEADOF	Leitwertkopplung aus
1) Kapitel:"Kopplungen (CP..., LEAD..., TRAIL..., CTAB...) (Seite 101)"	

Vordefinierte Unterprogramme: Kopplungen > Drehmomentkopplung (Master/Slave)	
MASLON	Kopplung ein
MASLOF	Kopplung aus
MASLDEF	Kopplung definieren
MASLDEL	Kopplung löschen
MASLOFS	Kopplung mit Slave-Spindel aus

Vordefinierte Funktionen: Kopplung > Kurventabellen [1)]	
CTAB	Berechnet über die Kurventabelle die Folgeachseposition anhand der Leitachsposition
CTABINV	Berechnet über die Kurventabelle die Leitachsposition anhand der Folgeachseposition
CTABID	Ermittelt die Tabellennummer der Kurventabelle
CTABLOCK	Kurventabelle sperren
CTABUNLOCK	Kurventabelle freigeben
CTABISLOCK	Ermittelt den Sperrstatus der Kurventabelle
CTABEXISTS	Prüft ob die Kurventabelle existiert
CTABMEMTYP	Ermittelt den Speicherort der Kurventabelle (statischer / dynamischer Speicher)
CTABPERIOD	Ermittelt die Periodizität der Kurventabelle
CTABNO	Ermittelt die Anzahl an Kurventabellen
CTABNOMEM	Ermittelt speicherortspezifisch die Anzahl der vorhandenen Kurventabellen
CTABSEG	Ermittelt speicherortspezifisch die Anzahl der bereits verwendeter Kurvensegmente
CTABSEGID	Ermittelt tabellenspezifisch die Anzahl der bereits verwendeter Kurvensegmente
CTABFSEG	Ermittelt tabellenspezifisch die Anzahl der noch möglichen Kurvensegmente
CTABMSEG	Ermittelt speicherortspezifisch die Anzahl der maximal möglichen Kurvensegmente
CTABPOL	Ermittelt speicherortspezifisch die Anzahl der bereits verwendeter Polynome
CTABPOLID	Ermittelt tabellenspezifisch die Anzahl der bereits verwendeter Polynome
CTABFPOL	Ermittelt tabellenspezifisch die Anzahl der noch möglichen Polynome
CTABMPOL	Ermittelt speicherortspezifisch die Anzahl der maximal möglichen Polynome
CTABTSV	Ermittelt den Folgewert am Anfang der Tabelle
CTABTEV	Ermittelt den Folgewert am Ende der Tabelle
CTABTSP	Ermittelt den Leitwert am Anfang der Tabelle
CTABTEP	Ermittelt den Leitwert am Ende der Tabelle
CTABTMIN	Ermittelt den minimalen Folgewert der Tabelle
CTABTMAX	Ermittelt den maximalen Folgewert der Tabelle

Vordefinierte Funktionen: Kopplung > Kurventabellen [1)]	
CTABFNO	Ermittelt speicherortspezifisch die Anzahl der noch möglichen Kurventabellen
CTABSSV	Ermittelt für die Folgeachsen den Startwert eines Tabellensegments
CTABSEV	Ermittelt für die Folgeachsen den Endwert eines Tabellensegments
1) Kapitel:"Kopplungen (CP..., LEAD..., TRAIL..., CTAB...) (Seite 101)"	

Vordefinierte Funktionen: Arithmetik	
SIN	Sinus
ASIN	Arcus-Sinus
COS	Cosinus
ACOS	Arcus-Cosinus
TAN	Tangens
ATAN2	Arcus-Tangens2
SQRT	Quadratwurzel
POT	2. Potenz (Quadrat)
TRUNC	Ganzzahliger Teil
ROUND	Runden auf nächste ganze Zahl
ROUNDUP	Aufrunden eines Eingabewertes auf die nächste ganze Zahl
ABS	Betrag
LN	Natürlicher Logarithmus
EXP	Exponentialfunktion
MINVAL	Kleinerer von zwei Werten
MAXVAL	Größer von zwei Werten
BOUND	Prüfen auf definierten Wertebereich

Vordefinierte Funktionen: Aktuelle Maschinendatenwerte	
GETMDACT	Ermittelt den aktuellen Wert des Maschinendatums
GETMDPEAK	Ermittelt den seit letztem RESETPEAK maximalen im Maschinendatum vorgekommenen Wert
GETMDLIM	Ermittelt den maximalen oder minimalen Grenzwert des Maschinendatums
RESETPEAK	Setzt für GETMDPEAK den maximalen Wert wieder zurück

Vordefinierte Funktionen: Formatwandlungen	
ITOR	INT → REAL
RTOI	REAL → INT
RTOB	REAL → BOOL
BTOR	BOOL → REAL
ITOB	INT → BOOL
BTOI	BOOL → INT

Vordefinierte Funktionen: Safety Integrated	
SIRELAY	Aktivieren der mit SIRELIN, SIRELOUT und SIRELTIME parametrierten Sicherheitsfunktionen

Vordefinierte Funktionen: Verschiedenes	
POSRANGE	Achsposition innerhalb des Toleranzbereiches um die Referenzposition Kapitel:"Position im vorgegebenen Referenzbereich (POSRANGE) (Seite 76)"
PRESETON	Istwertsetzen für eine Achse Kapitel:"Istwertsetzen mit Verlust des Referenzierstatus (PRESETON) (Seite 89)"

Literatur

Ausführliche Beschreibungen zu den nicht in diesem Handbuch beschriebenen Sprachelementen finden sich in:

- Programmierhandbuch Grundlagen
- Programmierhandbuch Arbeitsvorbereitung

3.6 Sprachelemente nur für Technologiezyklen

Folgende Sprachelemente dürfen nur in Technologiezyklen verwendet werden:

Sprunganweisungen	
IF	Verzweigung
GOTO	Springe auf Label, Suchrichtung erst vorwärts, dann rückwärts
GOTOF	Springe auf Label, Suchrichtung vorwärts
GOTOB	Springe auf Label, Suchrichtung rückwärts

Programmende	
M02	Programmende
M17	Programmende
M30	Programmende
RET	Programmende

Literatur

Ausführliche Beschreibungen zu den nicht in diesem Handbuch beschriebenen Anweisungen finden sich in:

- Programmierhandbuch Grundlagen
- Programmierhandbuch Arbeitsvorbereitung

3.7 Aktionen in Synchronaktionen

3.7.1 Ausgabe von M-, S- und H-Hilfsfunktionen an die PLC

Ausgabezeitpunkt

Aus Synchronaktionen können Hilfsfunktionen vom Typ M, S und H ausgegeben werden. Die Ausgabe an die PLC erfolgt sofort, d. h. unmittelbar in dem Interpolatortakt in dem die Aktion ausgeführt wird.

Eventuell über Maschinendaten eingestellte Ausgabezeitpunkte für Hilfsfunktionen sind bei der Ausgabe aus Synchronaktionen unwirksam:

- MD11110 $MN_AUXFU_GROUP_SPEC (Hilfsfunktionsgruppenspezifikation)
- MD22200 $MC_AUXFU_M_SYNC_TYPE (Ausgabezeitpunkt der M-Funktionen)
- MD22210 $MC_AUXFU_S_SYNC_TYPE (Ausgabezeitpunkt der S-Funktionen)
- MD22230 $MC_AUXFU_H_SYNC_TYPE (Ausgabezeitpunkt der H-Funktionen)

Maximale Anzahl

Allgemein

Aus dem Teileprogramm und den aktiven Synchronaktionen eines Kanals heraus, dürfen in Summe gleichzeitig, d. h. in einem OB40-Zyklus der PLC, maximal 10 Hilfsfunktionen ausgegeben werden.

Synchronaktionsspezifisch

Die maximal zulässige Anzahl von Hilfsfunktionen im Aktionsteil eine Synchronaktion beträgt:

- M-Funktionen: 5
- S-Funktionen: 3
- H-Funktionen: 3

Satzweise Synchronaktionen

In satzweisen Synchronaktionen (ohne Angabe von `ID` oder `IDS`) dürfen Hilfsfunktionen nur im Zusammenhang mit der Häufigkeitsangabe `WHEN` oder `EVERY` ausgegeben werden.

Vordefinierte M-Funktionen

Vordefinierte M-Funktionen dürfen im Allgemeinen nicht in Synchronaktionen ausgegeben werden.

Ausnahmen: `M3`, `M4`, `M5`, `M40`, `M41`, `M42`, `M43`, `M44`, `M45`, `M70` und `M17`

Siehe auch

Häufigkeit (WHENEVER, FROM, WHEN, EVERY) (Seite 15)

3.7.2 Schreiben und Lesen von Systemvariablen

Die Systemvariablen des NCK sind im Listenhandbuch "Systemvariable" mit ihren jeweiligen Eigenschaften aufgelistet. Systemvariablen die im Aktionsteil von Synchronaktionen gelesen bzw. geschrieben werden können, sind in der entsprechenden Zeile (Read bzw. Write) der Spalte "SA" (Synchronaktion) mit "X" gekennzeichnet.

Hinweis

Systemvariable die in Synchronaktionen verwendet werden, werden implizit immer hauptlaufsynchron gelesen und geschrieben.

Literatur:
Listenhandbuch Systemvariable

3.7.3 Polynomauswertung (SYNFCT)

Anwendung

Mit der Funktion SYNFCT kann im Hauptlauf eine Variable gelesen, diese über ein Polynom bewertet und das Ergebnis in eine andere Variable geschrieben werden.
Anwendungsbeispiele:

- Vorschub in Abhängigkeit von der Antriebsauslastung
- Position in Abhängigkeit von einem Sensorsignal
- Laser-Leistung in Abhängigkeit von der Bahngeschwindigkeit

Syntax

```
SYNFCT(<Poly_Nr>,<SysVar_Out>,<SysVar_In>)
```

Bedeutung

Parameter	Bedeutung
<Poly_Nr>:	Nummer des mit FCTDEF definierten Polynoms: $f(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + a_3 \cdot x^3$
<SysVar_Out>:	Systemvariable, Ausgang: <SysVar_Out> = f(x)
<SysVar_In>:	Systemvariable, Eingang: x = <SysVar_In>
Zu FCTDEF siehe Kapitel "Polynomkoeffizienten, -parameter ($AC_FCT...) (Seite 39)"	

Beispiel: Additive Überlagerung des Bahnvorschubs

Zum programmierten Vorschub (F-Wort) wird ein Überlagerungswert addiert:

$F_{wirksam} = F_{programmiert} + F_{AC}$

<SysVar_Out>	Bedeutung
$AC_VC	additive Bahnvorschubkorrektur
$AA_VC[Achse]	additive axiale Vorschubkorrektur

Eingangswert sei der Stromistwert $AA_CURR der X-Achse.

Der Arbeitspunkt wird auf 5 A festgelegt.

Der Vorschub darf ±100 mm/min verändert werden, wobei die Abweichung des axialen Stromes ±1 A betragen darf.

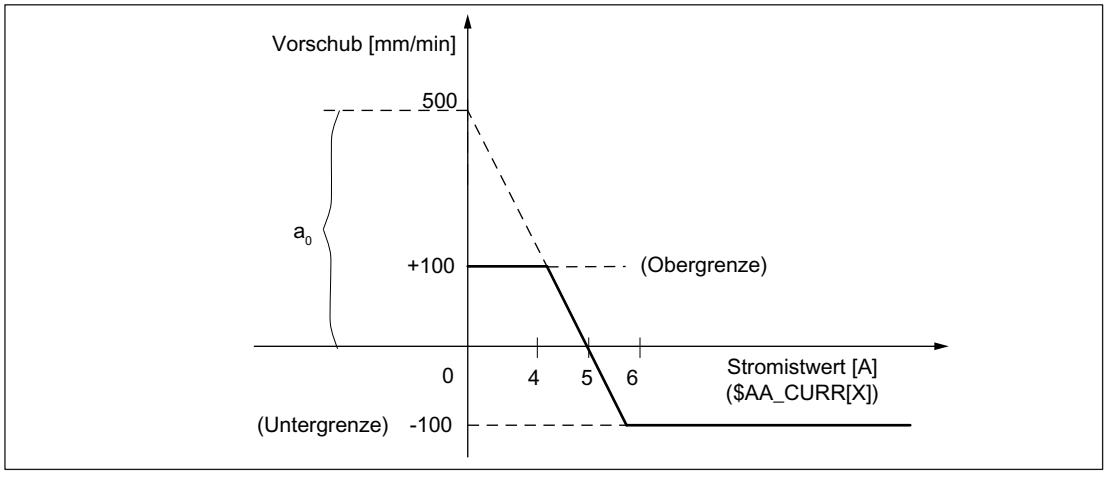


Bild 3-3 Beispiel: Additive Beeinflussung des Bahnvorschubs

Bestimmung der Parameter der Funktion `FCTDEF`:

`FCTDEF(<Poly_Nr>,<Lo_Limit>,<Up_Limit>,`a_0`,`a_1`,`a_2`,`a_3`)`

`<Poly_Nr>`: = 1 (beispielhaft)

`<Lo_Limit>`: = -100

`<Up_Limit>`: = 100

Polynom: $f(x) = a_0 + a_1x + a_2x^2 + a_3x^3$

a_0: 1 / 100 = 5 / $a_0 \Rightarrow a_0 = 500$

a_1 = 100 mm/min / -1 A = -100 [mm/min / A]

a_2 = 0 (kein quadratisches Glied)

a_3 = 0 (kein kubisches Glied)

Berechnung des Überlagerungswertes:

`SYNFCT(<Poly_Nr>,<SysVar_Out>,<SysVar_In>)`

`<Poly_Nr>`: = 1

<SysVar_Out>: $AC_VC (additive Bahnvorschubkorrektur)
<SysVar_In>: $AA_CURR (Antriebs-Stromistwert)

Programmierung:

Programmcode
N100 FCTDEF(1, -100, 100, 500, -100)
N110 ID=1 DO SYNFCT(1, $AC_VC[X], $AA_CURR[X])

Beispiel: Multiplikative Überlagerung des Bahnvorschubs

Der programmierte Vorschub wird mit einem prozentualen Faktor multipliziert (zusätzlicher Override):

$F_{wirksam} = F_{programmiert} * Faktor_{AC}$

<SysVar_Out>	**Bedeutung**
$AC_OVR	Bahnoverride über Synchronaktion vorgebbar

Eingangswert sei die prozentuale Antriebsauslastung $AA_LOAD der X-Achse.

Der Arbeitspunkt wird auf 100% bei 30%-iger Auslastung des Antriebs festgelegt.

Bei 80%-iger Auslastung soll die Achse stehen.

Eine Überhöhung der Geschwindigkeit wird mit +20% der programmierten Geschwindigkeit zugelassen.

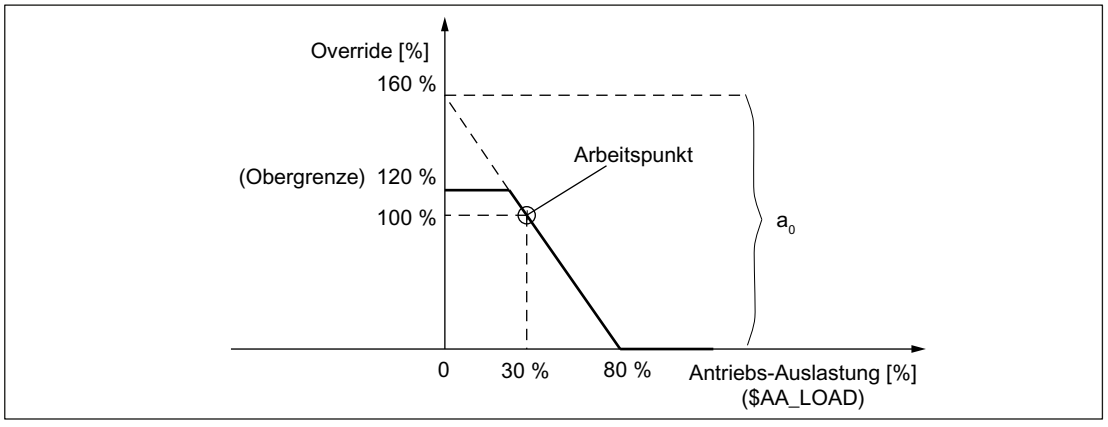


Bild 3-4 Beispiel: Multiplikative Beeinflussung

Bestimmung der Parameter der Funktion FCTDEF:

FCTDEF(<Poly_Nr>,<Lo_Limit>,<Up_Limit>,a_0,a_1,a_2,a_3)
<Poly_Nr>: = 2 (beispielhaft)
<Lo_Limit>: = 0
<Up_Limit>: = 120
Polynomy: $f(x) = a_0 + a_1x + a_2x^2 + a_3x^3$
a_0: $50 / 100 = 80 / a_0 \Rightarrow a_0 = 160$

a_1 = 100 % / -50 % = **-2**
a_2 = **0** (kein quadratisches Glied)
a_3 = **0** (kein kubisches Glied)

Berechnung des Überlagerungswertes:

`SYNFCT(<Poly_Nr>,<SysVar_Out>,<SysVar_In>)`
`<Poly_Nr>`: **= 2**
`<SysVar_Out>`: $AC_OVR（Bahnoverride über Synchronaktion vorgebbar)
`<SysVar_In>`: $AA_LOAD (Antriebsauslastung)

Programmierung:

Programmcode

```
N100 FCTDEF(2, 0, 120, 160, -2)
N110 ID=1 DO SYNFCT(2, $AC_OVR[X], $AA_LOAD[X])
```

Beispiel: Abstandsregelung

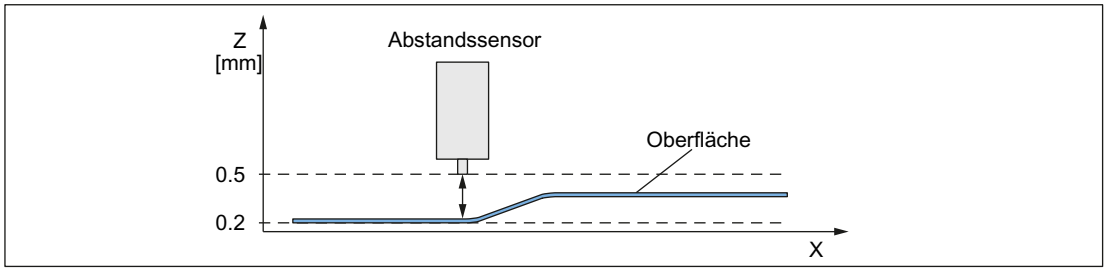


Bild 3-5 Abstandsregelung: Prinzip

Die Abstandsregelung der Zustellachse Z wird über die Funktionen `FCTDEF` und `SYNFCT` sowie den Systemvariablen $AA_OFF und $A_INA durchgeführt.

Randbedingungen:

- Die Analogspannung des Abstandssensors wird über den Analogeingang $A_INA[3] angeschlossen.
- Die Lageabweichungen werden in $AA_OFF aufsummiert (integriert):
  MD36750 $MA_AA_OFF_MODE, Bit 0 = 1
- Bei Überschreitung des oberen Grenzwertes der Z-Achse vom 1 mm wird die X-Achse angehalten:
  SD43350 $SA_AA_OFF_LIMIT[Z] = 1
  Siehe dazu auch Kapitel "Überlagerte Bewegungen ($AA_OFF) (Seite 42)".

Hinweis

$AA_OFF wirkt im Basiskoordinatensystem (BKS)

Die Korrektur wirkt vor der kinematischen Transformation im Basiskoordinatensystem (BKS). Das Beispiel kann daher **nicht** für eine Abstandsregelung in Orientierungsrichtung des Werkzeuges (Werkstückkoordinatensystem WKS) verwendet werden.

Zu Abstandsregelung mit hohen Dynamikanforderungen oder 3D-Abstandsregelung siehe:

Literatur:

Funktionshandbuch Sonderfunktionen; Abstandsregelung (TE1)

Anwenderspezifische Reaktionen

Mit Erreichen des Grenzwerts SD43350 $SA_AA_OFF_LIMIT können anwenderspezifische Reaktionen ausgelöst werden z.B.:

- Kapitel "Override ($A...OVR) (Seite 33)"
- Kapitel "Anwenderspezifische Fehlerreaktionen (SETAL) (Seite 111)"

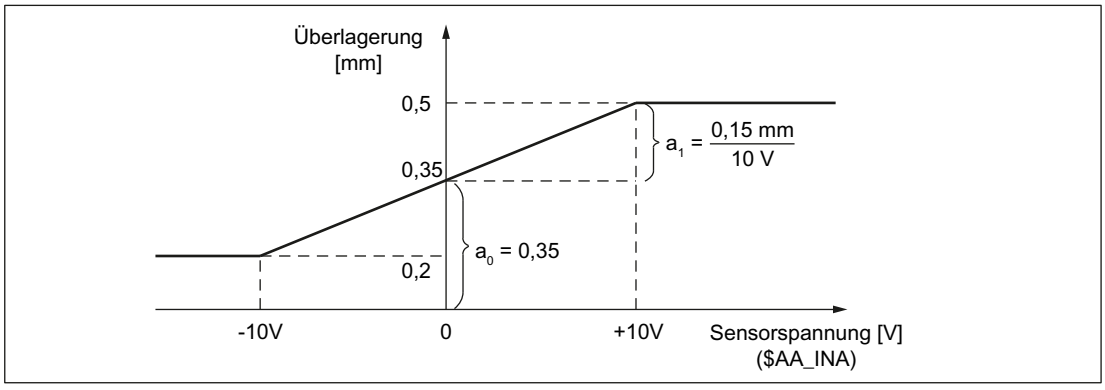


Bild 3-6 Abstandsregelung

Bestimmung der Parameter der Funktion `FCTDEF`:

`FCTDEF(<Poly_Nr>,<Lo_Limit>,<Up_Limit>,`a_0,`$a_1$,`a_2,`$a_3$`)`

`<Poly_Nr>`: = 1 (beispielhaft)

`<Lo_Limit>`: = 0,2

`<Up_Limit>`: = 0,5

Polynomy: $f(x) = a_0 + a_1x + a_2x^2 + a_3x^3$

a_0: 10 / x = 20 / 0,3 ⇒ a_0 = x + 0,2 = 0,15 + 0,2 = 0,35
a_1 = 0,15 mm / 10 V = 1,5 * 10^{-2} mm/V
a_2 = **0** (kein quadratisches Glied)
a_3 = **0** (kein kubisches Glied)

Berechnung des Überlagerungswerts:

`SYNFCT(<Poly_Nr>,<SysVar_Out>,<SysVar_In>)`
`<Poly_Nr>`: = **1**
`<SysVar_Out>`: $AA_OFF（Überlagerte Bewegung einer Achse)
`<SysVar_In>`: $A_INA (Analoger Eingang)

Programmierung:

Programmcode: %_N_AON_SPF	**Kommentar**
`PROC AON`	`; Abstandsregelung "EIN"`
`  FCTDEF(1, 0.2, 0.5, 0.35, 1.5 EX-2)`	`; Polynomdefinition`
`  ID=1 DO SYNFCT(1, $AA_OFF[Z], $A_INA[3])`	`; Abstandsregelung`
`  ID=2 WHENEVER $AA_OFF_LIMIT[Z]<>0 DO $AA_OVR[X] = 0`	`; Grenzwertprüfung`
`  RET`	
`ENDPROC`	

Programmcode: %_N_AOFF_SPF	**Kommentar**
`PROC AOFF`	`; Abstandsregelung "AUS"`
`  CANCEL(1)`	`; Abstandsregelung löschen`
`  CANCEL(2)`	`; Grenzwertprüfung löschen`
`  RET`	
`ENDPROC`	

Programmcode: %_N_MAIN_MPF	**Kommentar**
`N100 $SA_AA_OFF_LIMIT[Z]=1`	
`N110 AON`	`; Abstandsregelung "EIN"`
`...`	
`N200 G1 X100 F1000`	
`N210 AOFF`	`; Abstandsregelung "AUS"`
`M30`	

Siehe auch

Online-Werkzeugkorrektur (FTOC) (Seite 67)

3.7.4 Online-Werkzeugkorrektur (FTOC)

Die Funktion FTOC ermöglicht zur Online-Werkzeugkorrektur die überlagerte Bewegung einer Geometrieachse in Abhängigkeit von einem Bezugswert, z. B. dem Istwert einer beliebigen Achse. Der Korrekturwert wird dabei anhand eines mit `FCTDEF` definierten Polynoms errechnet (siehe Kapitel "Polynomkoeffizienten, -parameter ($AC_FCT...) (Seite 39)"). Der bei der Polynomdefinition angegebene Koeffizient a_0 wird auch von `FTOC` ausgewertet.

Beispiel: Bearbeiten und Abrichten in der Technologie "Schleifen"

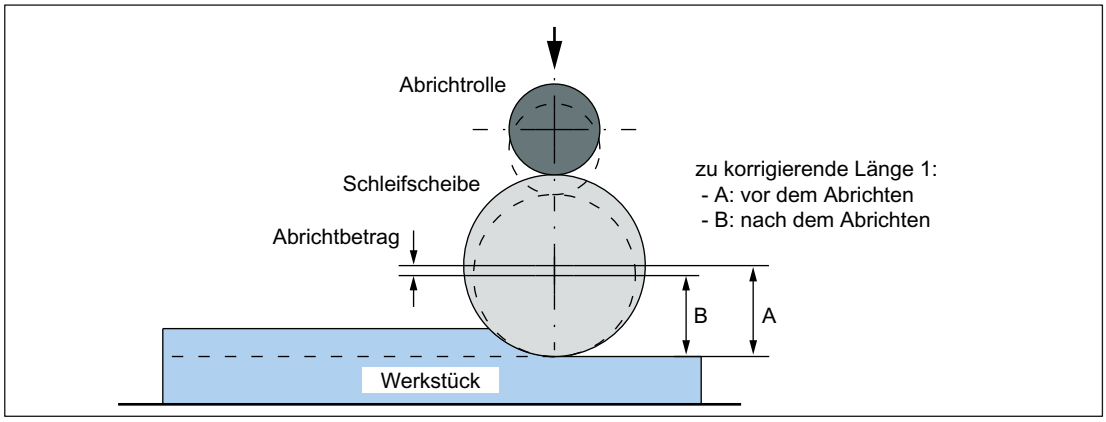


Bild 3-7 Abrichten während der Bearbeitung mit einer Abrichtrolle

Literatur:

Funktionshandbuch Erweiterungsfunktionen; Schleifen (W4)

Syntax

```
FTOC(<Poly_Nr>,<Systemvar>,<Verschleiß>[,<Kanal_Nr>,<Spindel_Nr>])
```

Bedeutung

Parameter	Bedeutung
`<Poly_Nr>`:	Nummer des mit `FCTDEF` definierten Polynoms
`<Systemvar>`:	Beliebige Systemvariable vom Typ REAL, die in Synchronaktionen verwendet werden kann.
`<Verschleiß>`:	Verschleißparameter (Länge 1, 2 oder 3), in dem der Korrekturwert **addiert** wird.

Parameter	Bedeutung
<Kanal_Nr>:	Zielkanal, in dem die Korrektur wirken soll. Damit ist zeitgleiches Abrichten aus einem parallelen Kanal möglich. Im Zielkanal der Korrektur muss die Online-Korrektur mit FTOCON eingeschaltet sein. Wird keine Kanalnummer programmiert, wirkt die Korrektur im aktiven Kanal.
<Spindel_Nr>:	Die Spindelnummer wird programmiert, wenn eine nicht aktive Schleifscheibe abgerichtet werden soll. Voraussetzung: eine der folgenden Funktionen ist aktiv • "konstante Scheibenumfangsgeschwindigkeit" • "Werkzeugüberwachung" Wird keine Spindelnummer programmiert, wird das aktive Werkzeug korrigiert.

Beispiel

Länge einer aktiven Schleifscheibe korrigieren

```
Programmcode                                    Kommentar
FCTDEF(1, -1000, 1000, -$AA_IW[V], 1)
; FTOC:
; Polynnom-Nr: 1
; Systemvariable: $AA_IW[V] (axialer Istwert der V-Achse)
; Verschleißparameter: Länge 3
; Zielkanal: Kanal 1
ID=1 DO FTOC(1, $AA_IW[V], 3, 1)
WAITM (1,1,2)                                   ; Synchronisation mit dem Bearbeitungskanal
G1 V-0.05 F0.01 G91                             ; Verfahrbewegung der V-Achse
...
CANCEL(1)                                       ; Online-Korrektur abwählen


...
```

Hinweis

Da keine Häufigkeit und keine Bedingung in der Synchronaktion angegeben ist, wird der Aktionsteil in jedem Interpolatortakt ausgeführt.

3.7.5 Programmierte Einlesesperre (RDISABLE)

Funktion

Der RDISABLE-Befehl im Aktionsteil bewirkt, dass die weitere Satzbearbeitung angehalten wird, wenn die zugehörige Bedingung erfüllt ist. Es werden nur noch die programmierten Bewegungssynchronaktionen bearbeitet. Wenn die Bedingung für die RDISABLE-Anweisung nicht mehr erfüllt ist, wird die Einlesesperre aufgehoben.

Am Ende des Satzes mit RDISABLE wird Genauhalt ausgelöst, unabhängig davon, ob die Einlesesperre wirksam wird oder nicht. Der Genauhalt wird auch ausgelöst, wenn sich die Steuerung im Bahnsteuerbetrieb befindet (G64, G641 ... G645).

RDISABLE kann satzbezogen oder auch modal (ID=, IDS=) programmiert sein!

Anwendung

Mit RDISABLE kann z. B. abhängig von externen Eingängen das Programm im Interpolatortakt gestartet werden.

Beispiel

Programmcode	Kommentar
WHENEVER $A_INA[2]<7000 DO RDISABLE	; Wenn die Spannung 7V am Eingang 2 unterschreitet, wird die Programmfortsetzung angehalten (Annahme: Wert 1000 entspricht 1V).
...	
N10 G01 X10	; Am Ende von N10 wirkt RDISABLE, wenn während seiner Bearbeitung die Bedingung erfüllt ist.
N20 Y20	
...	

Randbedingungen

Einlesesperre RDISABLE im Zusammenhang mit Achstausch

Wirkt über Synchronaktionen Einlesesperre RDISABLE und Achstausch (z.B. Bahnachse → Positionierachse) gemeinsam in einem Satz, wirkt RDISABLE nicht auf den Aktionssatz, sondern auf den durch den Achstausch implizit erzeugten REPOSA Wiederanfahrsatz:

Programmcode	Kommentar
N100 G0 G60 X300 Y300	
N105 WHEN TRUE DO POS[X]=20 FA[X]=20000	; Synchronaktion → REORG → REPOSA
N110 WHENEVER $AA_IM[X]<>20 DO RDISABLE	; RDISABLE wirkt auf REPOSA
N115 G0 Y20	; 1. X-Achse, 2. Y-Achse
N120 Y-20	
N125 M30	

Durch die Synchronaktion in Satz N105 wird aus der Bahnachse X eine Positionierachse. Dabei wird im Kanal REORG mit REPOSA ausgeführt. RDISABLE in N110 wirkt daher nicht auf

den Satz N115 sondern auf den internen REPOSA-Satz. Dadurch wird zuerst die Positionierachse X und erst danach im Satz N115 die Y-Achse auf ihre programmierte Position verfahren.

Eine explizite Freigabe der Bahnachse X vor dem Verfahren als Positionierachse (Synchronaktion in N105) mit RELEASE(X) vermeidet den REORG-Vorgang und die X- und Y-Achse verfahren gemeinsam in Satz N115.

```
Programmcode                                    Kommentar
N100 G0 G60 X300 Y300
N101 RELEASE(X)                                 ; Explizite Freigabe
N105 WHEN TRUE DO POS[X]=20 FA[X]=20000
...
```

3.7.6 Vorlaufstopp aufheben (STOPREOF)

Mit dem Befehl STOPREOF kann aus Synchronaktion heraus ein bestehender Vorlaufstopp aufgehoben werden.

Hinweis

Der Befehl STOPREOF darf nur in satzweisen Synchronaktionen (ohne Angabe von ID oder IDS) und nur im Zusammenhang mit der Häufigkeitsangabe WHEN programmiert werden.

Beispiel

- N10: Satzweise Synchronaktion
  Wird der Bahnrestweg $AC_DTEB kleiner als 5 mm, wird der bestehende Vorlaufstopp aufgrund des Lesens des analogen Eingangs $A_INA aufgehoben.
- N20: Verfahrsatz, dessen Bahnrestweg über $AC_DTEB ausgewertet wird.
- N30: Verzweigung, die aufgrund des Lesens von $A_INA Vorlaufstopp auslöst.

Aufgrund der Synchronaktion wird nicht erst ab Satzende von N20 der Eingang $A_INA ausgewertet, sondern bereits 5 mm vor dem Satzende. Wird dann am Eingang $A_INA die Spannung größer als 5 V, wird zu " MARKE_1" verzweigt.

```
Programmcode
N10 WHEN $AC_DTEB < 5 DO STOPREOF
N20 G01 X100
N30 IF $A_INA[7] > 5000 GOTOF MARKE_1
```

3.7.7 Restweglöschen (DELDTG)

Mit dem Befehl DELDTG kann in Synchronaktionen der Bahnrestweg und mit der Funktion DELDTG (...) axiale Restwege gelöscht werden.

Nach dem Restweglöschen kann der Wert des gelöschten Restweges über eine Systemvariable gelesen werden:

- Bahnrestweg: $AC_DELT
- Axialer Restweg: $AA_DELT

Syntax

```
DELDTG
DELDTG(<Achse 1>[,<Achse 2>,...])
```

Bedeutung

Parameter	Bedeutung
`DELDTG`	Löschen des Bahnrestweges
`DELDTG(...)`	Löschen der axialen Restwege der angegebenen Kanalachsen
`<Achse n>`:	Kanalachse

Randbedingungen

Bahnspezifisches und axiales Restweglöschen

Bahnspezifisches und axiales Restweglöschen darf nur in einer **satzweise** wirksamen Synchronaktion (ohne Angabe von ID oder IDS) ausgeführt werden.

Bahnspezifisches Restweglöschen

- Das Löschen des Bahnrestweges darf nur in einer satzweise wirksamen Synchronaktion (ohne Angabe von ID oder IDS) ausgeführt werden.
- Das Löschen des Bahnrestweges darf bei aktiver Werkzeugradiuskorrektur **nicht** verwendet werden.

Axiales Restweglöschen

Restweglöschen bei Teilungsachsen:

- **ohne** Hirth-Verzahnung: Die Achse wird sofort abgebremst
- **mit** Hirth-Verzahnung: Die Achse fährt bis zur nächsten Teilungsposition

Beispiele

Bahnrestweg löschen

Wird während des Verfahrsatzes N20 der Eingang $A_IN gesetzt, wird der Bahnrestweg gelöscht.

Programmcode

```
N10 WHEN $A_IN[1]==1 DO DELDTG
N20 G01 X100 Y100 F1000
```

Axiale Restwege löschen

N10: Wird zu einem beliebigen Zeitpunkt innerhalb des Teileprogramms der Eingang 1 gesetzt, wird die die V-Achse als Positionierachse in positiver Verfahrrichtung gestartet.

N100: Satzweise Synchronaktion zum Restweglöschen der V-Achse, abhängig vom digitalen Eingang 2.

N110: Satzweise Synchronaktion zum Restweglöschen der X1-Achse, abhängig vom digitalen Eingang 3.

N120: Die X1-Achse wird satzübergreifend positioniert. Die Y- und Z-Achse als Bahnachsen verfahren. Die satzweisen Synchronaktionen aus N100 und N110 werden zusammen mit N120 ausgeführt. Mit dem Satzende von N120 werden auch die satzweisen Synchronaktionen beendet.

Somit kann der Restweg der X1- und V-Achse nur gelöscht werden, solange N120 aktiv ist.

```
Programmcode
N10 ID=1 WHEN $A_IN[1]==1 DO MOV[V]=1 FA[V]=700
...
N100 WHEN $A_IN[2]==1 DO DELDTG(V)
N110 WHEN $A_IN[3]==1 DO DELDTG(X1)
N120 POSA[X1]=100 FA[X1]=10 G1 Y100 Z100 F1000
```

3.7.8 Verfahren von Achsen, auf Position (POS)

MIt dem Befehl `POS` kann eine Achse über eine Synchronaktion verfahren werden. Die Achse wird dann als Kommandoachse bezeichntet. Ein wechselseitiges Verfahren der Achse über Teileprogramm und Synchronaktion ist möglich. Wird eine über Synchronaktionen verfahrene Kommandoachse anschließend über das Teileprogramm verfahren, wird im Kanal des Teileprogramms ein Vorlaufstopp mit Reorganisieren (`STOPRE`) ausgeführt.

Beispiele:

Beispiel 1: Wechselseitiges Verfahren über Teileprogramm und Synchronaktion

```
Programmcode                                  Kommentar
N10 G01 X100 Y200 F1000                       ; Verfahren über Teileprogramm
...
; Verfahren über statische Synchronaktion wenn Eingang 1 gesetzt wird
N20 ID=1 WHEN $A_IN[1]==1 DO POS[X]=150 FA[X]=200
...
CANCEL(1)                                     ; Synchronaktion abwählen
...
; Erneutes Verfahren über Teileprogramm => impliziter Vorlaufstopp
; mit Reorganisieren, falls die X-Achse zwischenzeitlich über die
; Synchronaktion verfahren wurde
N100 G01 X240 Y200 F1000
```

Beispiel 2: Wechselseitiges Verfahren der X-Achse über zwei Synchronaktionen

Ist die Verfahrbewegung einer Synchronaktion noch aktiv, wenn die Verfahrbewegung der anderen Synchronaktion gestartet wird, löst die zweite Verfahrbewegung die erste ab.

Programmcode

```
; 1. Verfahrbewegung
ID=1 EVERY $A_IN[1]>=1 DO POS[V]=100 FA[V]=560
; 2. Verfahrbewegung
ID=2 EVERY $A_IN[2]>=1 DO POS[V]=$AA_IM[V] FA[V]=790
```

Maßangabe: absolut / inkrementell

In Synchronaktionen können die Befehle `G90` / `G91` zur Festlegung der Maßangabe (absolut / inkrementell) nicht programmiert werden. In einer Synchronaktion wirkt daher defaultmäßig die Maßangabe, die im Teileprogramm zum Zeitpunkt der Ausführung der Synchronaktion wirksam war.

Zur Festlegung der Maßangabe innerhalb einer Synchronaktion, können im Aktionsteil folgende Befehle programmiert werden:

Befehl	Bedeutung
`IC(...)`	inkrementell
`AC(...)`	absolut
`DC(...)`	direkt, d. h. Rundachse auf kürzestem Weg positionieren
`ACN(...)`	Modulo-Rundachse absolut positionieren in negativer Bewegungsrichtung
`ACP(...)`	Modulo-Rundachse absolut programmieren in positiver Bewegungsrichtung
`CAC(...)`	Achse auf codierte Position verfahren, absolut
`CIC(...)`	Achse auf codierte Position verfahren, inkrementell
`CDC(...)`	Rundachse auf kürzestem Weg auf codierte Position verfahren
`CACN(...)`	Modulo-Rundachse in negativer Richtung auf codierte Position verfahren
`CACP(...)`	Modulo-Rundachse in positiver Richtung auf codierte Position verfahren

Beispiele:

Programmcode

```
; inkrementelles Verfahren um 10 mm
ID=1 EVERY G710 $AA_IM[B]>75 DO POS[X]=IC(10)
...
; absolutes Verfahren
ID=1 EVERY G710 $AA_IM[B]>75 DO POS[X]=AC($AA_MW[V]-$AA_IM[W]+13.5)
```

Verhalten bei aktiven axialen Frames

Wird für Synchronaktionen nicht explizit die Einrechnung von programmierbaren und einstellbaren Frames und Werkzeuglängenkorrekturen über das nachfolgende Maschinendatum deaktiviert, wirkt in der parallel zum Teileprogramm ausgeführten Synchronaktion der zum Ausführungszeitpunkt im Teileprogramm aktive Frame und/oder Werkzeuglängenkorrektur:

MD32074 $MA_FRAME_OR_CORRPOS_NOTALLOWED, Bit 9 = 1

Beispiele

Beispiel 1: Verfahren mit **aktiven** Frames / Werkzeuglängenkorrekturen (Bit 9 == 0):

```
Programmcode                                    Kommentar
N100 TRANS X20                                  ; Nullpunktverschiebung in X: 20 mm.
; Synchronaktion: die X-Achse verfährt auf Position 60 mm
IDS=1 EVERY G710 $A_IN==1 DO POS[X]=40
...
; Nullpunktverschiebung in X: -10 mm. =>
; Synchronaktion: die X-Achse verfährt jetzt auf Position 30 mm
N130 TRANS X-10
...
```

Beispiel 2: Verfahren mit **deaktivierten** Frames / Werkzeuglängenkorrekturen (Bit9 == 1):

```
Programmcode                    Kommentar
N100 TRANS X=0,001              ; Nullpunktverschiebung in X: 0,001 Grad
N120 POS[X]=270                 ; X fährt auf Position 270.001 Grad
...
; Mit $A_IN=1, X fährt auf Position 180.000 Grad
IDS=1 EVERY G710 $A_IN==1 DO POS[X]=180
...
; X fährt auf Position 90.001 Grad
N130 POS[X]=90
...
; codierte Position 1 = 100 Grad => X fährt auf 100.001 Grad
N140 POS[X]=CAC(1)
...
; codierte Position 2 = 200 Grad => X fährt auf 200.000 Grad
N150 POS[X]=CIC(1)
```

Hinweis

Wird einen Kommandoachse inkrementell auf Teilungspositionen verfahren, werden die axialen Frames für diese Kommandoachse **nicht** wirksam.

Übernahmen der Kontrolle einer Kommandoachse von PLC

Eine Kommandoachse, die über eine statische Synchronaktion (`IDS`) gestartet wurde, wird unabhängig vom Status des Teileprogramms welches die Synchronaktion enthält, wenn die Kontrolle der Kommandoachse von der PLC übernommen wird:

DB31, ... DBX28.7 == 1 (PLC kontrolliert Achse)

Ausführliche Informationen zur Kontrolle einer Kommandoachse von PLC finden sich in:

Literatur:
/FB2/ Funktionshandbuch Erweiterungsfunktionen; Positionierachsen (P2)

Parametrierbarer Achsstatus

Über das folgende Maschinendatum kann das Verhalten bezüglich des Achsstatus nach Teileprogrammende- und NC-Reset parametriert werden:

MD30450 $MA_IS_CONCURRENT_POS_AX[<Achse>] = <Wert>

<Wert>	Achsstatus vor TP-Ende / NC-RESET 1)	Achsstatus nach TP-Ende / NC-RESET 1)
0	Kanalachse	Kanalachse
0	Kommandoachse	Kanalachse
1	Kanalachse	Kommandoachse
1	Kommandoachse	Kommandoachse
1) TP-Ende: Teileprogrammende		

Siehe auch

Technologiezyklen (Seite 112)

3.7.9 Maßsystem einstellen (G70, G71, G700, G710)

Wird in einer Synchronaktionen mit `G70`, `G71`, `G700`, `G710` nicht explizit ein bestimmtes Maßsystem (inch / metrisch) vorgegeben, wirkt das zum Ausführungszeitpunkt der Synchronaktion im Teileprogramm aktive Maßsystem:

- `G70/G71` im Teileprogramm aktiv:
  - Alle **programmierten** Positionswerte werden im **programmierten** Maßsystem interpretiert.
  - Alle **gelesenen** Positionsdaten werden im **parametrierten Grundsystem** interpretiert.
- `G700/G710` im Teileprogramm aktiv:
  - Alle **programmierten** Positionswerte werden im **programmierten** Maßsystem interpretiert.
  - Alle **gelesenen** Positionsdaten werden im **parametrierten Grundsystem** interpretiert.

Bei der Definition des Maßsystems in der Synchronaktion gelten folgende Regeln:

- Wird im Bedingungsteil ein Maßsystem programmiert, wirkt dieses auch im Aktionsteil wenn in diesem kein eigenes Maßsystem programmiert ist.
- Wird nur im Aktionsteil ein Maßsystem programmiert, wirkt im Bedingungsteil das aktuell im Teileprogramm aktive Maßsystem.
- Im Bedingungs- und Aktionsteil können unterschiedliche Maßsysteme programmiert werden.
- Das in der Synchronaktion programmierte Maßsystem hat keine Auswirkung auf das Teileprogramm

Beispiel

Programmcode	**Kommentar**
N10 ID=1 EVERY $AA_IM[Z]>200 DO POS[Z2]=10	; $AA_IM: #
	; 200: #
	; 10: #
N20 ID=2 EVERY $AA_IM[Z]>200 DO **G70** POS[Z2]=10	; $AA_IM: #
	; 200: #
	; 10: inch
N30 ID=3 EVERY **G71** $AA_IM[Z]>200 DO POS[Z2]=10	; $AA_IM: #
	; 200: mm
	; 10: mm
N40 ID=4 EVERY **G71** $AA_IM[Z]>200 DO **G70** POS[Z2]=10	; $AA_IM: #
	; 200: mm
	; 10: inch
N50 ID=5 EVERY $AA_IM[Z]>200 DO **G700** POS[Z2]=10	; $AA_IM: #
	; 200: #
	; 10: inch
N60 ID=6 EVERY **G710** $AA_IM[Z]>200 DO POS[Z2]=10	; $AA_IM: mm
	; 200: mm
	; 10: mm
N70 ID=7 EVERY **G710** $AA_IM[Z]>200 DO **G700** POS[Z2]=10	; $AA_IM: mm
	; 200: mm
	; 10: inch

#: die Einheit ist abhängig vom parametrierten Grundsystem (MD10240 $MN_SCALING_SYSTEM_IS_METRIC) und dem im Teileprogramm programmierten Maßsystem

Hinweis

Maßsystem und Technologiezyklen

Statt der Vorgabe des Maßsystems im Aktionsteil der Synchronaktion, kann bei Verwendung eines Technologiezyklus das Maßsystem auch im Technologiezyklus programmiert werden.

3.7.10 Position im vorgegebenen Referenzbereich (POSRANGE)

Funktion

Über die Funktion `POSRANGE` kann ermittelt werden, ob sich die aktuelle Position einer Achse innerhalb eines Toleranzbereichs um eine vorgegebene Referenzposition befindet.

Hinweis

Bei Modoluachsen wird die Modulo-Korrektur berücksichtigt.

Syntax

```
<Status> POSRANGE(<Achse>, <RefPos>, <Toleranz>, [<KoordSys>] )
```

Bedeutung

`<Status>`	Rückgabewert der Funktion Typ: BOOL TRUE: Die aktuelle Position der Achse ist innerhalb des Toleranzbereichs. FALSE: Die aktuelle Position der Achse ist nicht innerhalb des Toleranzbereichs.
`<Achse>`	Name der Kanalachse Typ: AXIS
`<RefPos>`	Referenzposition Typ: REAL
`<Toleranz>`	Zulässige Toleranz um die Referenzposition Typ: REAL Die Toleranz wird als Betrag angegeben. Der Toleranzbereich ergibt sich aus: Referenzposition +/- Toleranz
`<KoordSys>`	Optional: Koordinatensystem Typ: INT Wertebereich: 0 = MKS (Maschinenkoordinatensystem) 1 = BKS (Basiskoordinatensystem) 2 = ENS (Einstellbares Nullpunktsystem) 3 = WKS (Werkstückkoordinatensystem)

3.7.11 Verfahren von Achsen, endlos (MOV)

Funktion

Über den Befehl `MOV` kann eine Achse endlos, d.h. ohne Angabe einer Endposition, in eine bestimmte Richtung verfahren werden. Die Achse verfährt so lange in die vorgegebene Richtung, bis die Achse angehalten oder durch `MOV` eine andere Verfahrrichtung vorgegeben wird.

Anwendungsbeispiel: Endlos drehende Rundachse

Syntax

```
MOV[<Achse>] = <Richtung>
```

Bedeutung

`MOV`	Verfahrbefehl für eine Kommandoachse
`<Achse>`	Kanalachsname Typ: AXIS

`<Richtung>`	Verfahrrichtung Typ: INT Wertebereich: > 0: positive Verfahrrichtung (Standard: +1) < 0: negative Verfahrrichtung (Standard: -1) = 0: Anhalten

Hinweis

Teilungsachse

Wird eine Teilungsachse mit `MOV[<Teilungsachse>] = 0` angehalten, hält sie auf der nächster Teilungsposition.

Technologiezyklus

Der Befehl `MOV` darf **nicht** in Technologiezyklen verwendet werden.

Siehe auch

Axialer Vorschub (FA) (Seite 78)

3.7.12 Axialer Vorschub (FA)

Über den Befehl `FA` kann in einer Synchronaktion ein axialer Vorschub vorgegeben werden. Der axiale Vorschub ist modal wirksam.

Beispiele

Konstanter Vorschubwert:

Programmcode

```
ID=1 EVERY $AA_IM[B] > 75 DO POS[U]=100 FA[U]=990
```

Variabler Vorschubwert:

Programmcode

```
ID=1 EVERY $AA_IM[B] > 75 DO POS[U]=100 FA[U]=$AA_VACTM[W]+100
IDS=2 WHENEVER $A_IN[1] == 1 DO POS[X]=100 FA[X]=$R1
```

Anmerkungen

- Der Standardwert für den Vorschub von Positionierachsen wird über das axiale Maschinendatum eingestellt:
  MD32060 $MA_POS_AX_VELO (Löschstellung für Positionierachsgeschwindigkeit)
- Der axiale Vorschub kann als Linear- oder Umdrehungsvorschub vorgegeben werden. Den Vorschubtyp ist einstellbar über das Settingdatum:
  SD43300 $SA_ASSIGN_FEED_PER_REV_SOURCE (Umdrehungsvorschub für Positionierachsen/Spindeln)
- Der Vorschubtyp kann über die Befehle `FPRAON` und `FPRAOF` synchron zum Teileprogramm umgeschaltet werden. Siehe dazu:
  Literatur:
  /FB1/ Funktionshandbuch Grundfunktionen; Vorschübe (V1)

Hinweis

Damit sich parallel ablaufende Technologiezyklen nicht gegenseitig blockieren, wird der axiale Vorschub aus Synchronaktionen nicht als Hilfsfunktion an die NC/PLC-Nahtstelle ausgegeben

Siehe auch

Verfahren von Achsen, endlos (MOV) (Seite 77)

3.7.13 Achstausch (GET, RELEASE, AXTOCHAN)

Über die Befehle `GET`, `RELEASE` und `AXTOCHAN` können Kommandoachsen zwischen Kanälen getauscht werden.

Voraussetzung

Die Kommandoachse die zwischen den Kanälen geauscht wird, muss als Kanalachse im jeweiligen Kanal bekannt bzw. parametriert sein.

Programmierung

Syntax

```
GET(<Achse 1> [{, <Achse n>}])
RELEASE((<Achse 1> [{, <Achse n> }])
AXTOCHAN(<Achse 1>, <Kanalnummer 1> [{, <Achse n>, <Kanalnummer
n> }])
```

Bedeutung

`GET`:	Anforderung zum Tausch einer Achse in den eigenen Kanal
`RELEASE`:	Freigabe einer Achse zum Achstausc
`AXTOCHAN`:	Anforderung einer Achse zum Tausch in den angegebenen Kanal

<Achse n>:	Maschinenachsname	
	Typ:	AXIS
	Wertebereich:	im Kanal definierte Maschinenachsnamen
<Kanalnummer n>:	Kanalnummer	
	Typ:	INTEGER
	Wertebereich:	1 ... maximale Kanalnummer

Achstyp und Achsstatus bezüglich Achstausch

Der zum Aktivierungszeitpunkt der Synchronaktion aktuell gültige Achstyp und Achsstatus kann über die Systemvariablen $AA_AXCHANGE_TYP bzw. $AA_AXCHANGE_STAT abgefragt werden. Abhängig vom Kanal der das Interpolationsrecht dieser Achse besitzt, und vom Zustand für den zulässigen Achstausch, ergibt sich aus der Synchronaktion ein unterschiedlicher Ablauf.

Aus einer Synchronaktion heraus kann eine Achse mit `GET` angefordert werden, wenn

- ein anderer Kanal das Schreibrecht- bzw. Interpolationsrecht für die Achse hat
- die angeforderte Achse bereits dem angeforderten Kanal zugeordnet ist
- die Achse im Zustand neutrale Achse vom PLC kontrolliert ist
- die Achse Kommandoachse, Pendelachse oder konkurrierende PLC-Achse ist
- die Achse bereits dem NC-Programm des Kanals zugeordnet ist

Hinweis

Randbedingung: Eine "ausschließlich von der PLC kontrollierte Achse" oder eine "fest zugeordnete PLC-Achse", kann nicht dem NC-Programm zugeordnet werden.

Aus einer Synchronaktion heraus kann eine Achse mit `RELEASE` freigegeben werden, wenn die Achse:

- bisher dem NC-Programm des Kanals zugeordnet ist.
- bereits im Zustand neutrale Achse ist.
- bereits ein anderer Kanal hat das Interpolationsrecht dieser Achse hat.

Achse aus anderem Kanal anfordern

Hat zum Aktivierungszeitpunkt der Aktion `GET` ein **anderer Kanal** das Interpolationsrecht für die Achse $AA_AXCHANGE_TYP[Achse] == 2, so wird die Achse mittels Achstausch von diesem Kanal angefordert $AA_AXCHANGE_TYP[Achse] == 6 und sobald als möglich dem anfordernden Kanal zugeordnet. Sie nimmt dann den Zustand **neutrale Achse** $AA_AXCHANGE_TYP[Achse] == 3 an.

Der Zustandswechsel nach neutraler Achse **hat kein** Reorganisieren im anfordernden Kanal zur Folge.

Angeforderte Achse wurde bereits als neutrale Achse angefordert:

$AA_AXCHANGE_TYP[<Achse>]==6, so wird die Achse für das NC-Programm angefordert $AA_AXCHANGE_TYP[Achse] == 5 und sobald als möglich dem NC-Programm des Kanals zugeordnet $AA_AXCHANGE_TYP[Achse] == 0.

Hinweis

Diese Zuordnung **hat ein** Reorganisieren zur Folge.

Achse ist bereits dem angeforderten Kanal zugeordnet

Ist die angeforderte Achse zum Aktivierungszeitpunkt **bereits diesem Kanal** zugeordnet, und im Zustand neutrale Achse nicht vom PLC kontrolliert $AA_AXCHANGE_TYP[Achse] == 3, so wird diese Achse dem NC-Programm zugeordnet $AA_AXCHANGE_TYP[Achse] == 0.

Dies hat **einen** Reorganisationsvorgang zur Folge.

Achse im Zustand neutrale Achse ist PLC kontrolliert

Ist die Achse im Zustand neutrale Achse **vom PLC kontrolliert** $AA_AXCHANGE_TYP[Achse] == 4, so wird die Achse als neutrale Achse angefordert $AA_AXCHANGE_TYP[Achse] == 8. Dabei wird die Achse für einen automatischen Achstausch zwischen Kanälen gesperrt (Bit 0 == 0), abhängig vom Bit 0 im Maschinendatum:

MD10722 $MN_AXCHANGE_MASK (Parametrierung des Achstausch-Verhaltens)

Dies entspricht $AA_AXCHANGE_STAT[Achse] == 1.

Achse ist als Kommandoachse aktiv / PLC zugeordnet

Ist die Achse als Kommandoachse bzw. Pendelachse oder konkurrierende Positionierachse (PLC-Achse) aktiv ($AA_AXCHANGE_TYP[<Achse>] == 1), wird die Achse als neutrale Achse angefordert ($AA_AXCHANGE_TYP[<Achse>] == 8). Dabei wird die Achse, abhängig von der Einstellung im nachfolgenden Maschinendatum, für einen automatischen Achstausch zwischen Kanälen gesperrt,:

MD10722 $MN_AXCHANGE_MASK (Parametrierung des Achstausch-Verhaltens)

Dies entspricht $AA_AXCHANGE_STAT[<Achse>] == 1.

Mit einer erneuten Anforderung `GET` wird die Achse dann für das NC-Programm angefordert ⇒ $AA_AXCHANGE_TYP[Achse] == 7.

Achse bereits dem NC-Programm des Kanals zugeordnet

Ist die Achse bereits dem NC-Programm des Kanals zugeordnet ($AA_AXCHANGE_TYP[<Achse>] == 0) oder ist diese Zuordnung angefordert, z. B. Achstausch vom NC-Programm ausgelöst ($AA_AXCHANGE_TYP[<Achse>] == 5 bzw. $AA_AXCHANGE_TYP[<Achse>] == 7), ergibt sich **keine** Zustandsänderung.

Achse zum Achstausch freigeben

Ist die Achse zum Zeitpunkt der Freigabe dem NC-Programm zugeordnet ($AA_AXCHANGE_TYP[<Achse>] == 0), wird sie in den Zustand neutrale Achse überführt ($AA_AXCHANGE_TYP[<Achse>] == 3) und gegebenenfalls zum Achstausch in einen anderen Kanal freigegeben.

Dies hat **einen** Reorganisationsvorgang zur Folge.

Freizugebende Achse ist bereits neutrale Achse:

Ist die Achse bereits im Zustand neutrale Achse ($AA_AXCHANGE_TYP[<Achse>] == 3) oder als Kommando- bzw. Pendelachse aktiv oder als konkurrierende Positionierachse der PLC zugeordnet ($AA_AXCHANGE_TYP[<Achse>] == 1), wird die Achse für einen automatischen Achstausch zwischen Kanälen freigegeben.

$AA_AXCHANGE_STAT[<Achse>] wird, falls kein anderer Grund für eine Bindung der Achse an den Kanal vorliegt, von 1 auf 0 zurückgesetzt. Eine derartige Bindung der Achse liegt z. B. vor bei:

- aktiver Achskopplung
- aktivem Schnellabheben
- aktiver Transformation
- JOG-Anforderung
- rotierendes Frame mit PLC-, Kommando- oder Pendelachsbewegung

Ein anderer Kanal hat bereits das Interpolationsrecht

Hat bereits ein anderer Kanal das Interpolationsrecht ($AA_AXCHANGE_TYP[<Achse>] == 2), ergibt sich keine Zustandsänderung. Das bedeutet auch, dass das Warten auf eine Achse, ausgelöst durch NC-Programm ($AA_AXCHANGE_TYP[<Achse>] == 5) oder eine vorherige Anforderung `GET` aus einer Synchronaktion ($AA_AXCHANGE_TYP[<Achse>] == 6) nicht durch ein Freigabe `RELEASE` aus eine Synchronaktion abgebrochen werden kann.

Randbedingungen

- Werden im Aktionsteil einer Synchronaktion oder in einem Satz eines Technologiezyklus mehrere Anforderungen `GET` und Freigaben `RELEASE` für die gleiche Achse programmiert, heben sich diese Aufträge unter Umständen gegenseitig auf und es wird nur der jeweils letzte Auftrag ausgeführt.
  Beispiel:

  Programmierung: `GET(X,Y) RELEASE(Y,Z) GET(Z)`
  Ausführung: GET(X) RELEASE(Y) GET(Z)

- Sind im Aktionsteil einer Synchronaktion neben `GET` / `RELEASE` noch weitere Befehle programmiert, wird nicht erst auf den Abschluss der `GET` / `RELEASE` Anforderung gewartet bevor diese Befehle ausgeführt werden. Dadurch kann es zu einem Fehler kommen, falls z.B. eine mit `GET` angeforderte Achse für die Positionierbewegung noch nicht verfügbar ist:
  `GET[<Achse>] POS[<Achse>]`

Beispiel 1: GET und RELEASE als Aktion in Synchronaktionen in zwei Kanälen

Voraussetzung: Die Achse Z muss Kanalachse im 1. und 2. Kanal sein

1. Programmablauf im 1. Kanal:

Programmcode	Kommentar
WHEN TRUE DO RELEASE(Z)	; Z-Achse wird neutral
; Einlesesperre solange Z-Achse Programmachse ist	
WHENEVER $AA_TYP[Z] == 1 DO RDISABLE	
N110 G4 F0.1	
...	
; Z-Achse wird wieder NC-Programm-Achse	
WHEN TRUE DO GET(Z)	
; Einlesesperre bis Z-Achse Programmachse ist	
WHENEVER $AA_TYP[Z] <> 1 DO RDISABLE	
N120 G4 F0.1	
...	
WHEN TRUE DO RELEASE(Z)	; Z-Achse wird neutral
; Einlesesperre solange Z-Achse Programmachse ist	
WHENEVER $AA_TYP[Z] == 1 DO RDISABLE	
N130 G4 F0.1	
...	
N140 START(2)	; 2. Kanal starten
N150 ; siehe unten: "3. Fortsetzung: Programmablauf im 1. Kanal"	

2. Programmablauf im 2. Kanal:

Programmcode	Kommentar
WHEN TRUE DO GET(Z)	; Z-Achse in 2. Kanal holen (neutral)
; Einlesesperre solange Z-Achse in anderem Kanal ist	
WHENEVER $AA_TYP[Z] == 0 DO RDISABLE	
N210 G4 F0.1	
...	
WHEN TRUE DO GET(Z)	; Z-Achse wird NC-Programm-Achse
; Einlesesperre bis Z-Achse Programmachse ist	
WHENEVER $AA_TYP[Z] <> 1 DO RDISABLE	
N220 G4 F0.1	
...	
WHEN TRUE DO RELEASE(Z)	; Z-Achse im 2. Kanal neutral
; Einlesesperre solange Z-Achse Programmachse ist	
WHENEVER $AA_TYP[Z] == 1 DO RDISABLE	
N230 G4 F0.1	
...	
N250 WAITM(10,1,2)	; mit Kanal 1 synchronisieren
N999 M30	

3. Fortsetzung: Programmablauf im 1. Kanal:

```
Programmcode                                    Kommentar
N150 WAITM(10,1,2)                              ; mit Kanal 2 synchronisieren
...
WHEN TRUE DO GET(Z)                             ; Z-Achse in diesen Kanal holen
; Einlesesperre solange Z-Achse in anderem Kanal ist
WHENEVER $AA_TYP[Z] == 0 DO RDISABLE
N160 G4 F0.1
...
N199 WAITE(2)                                   ; warte auf Programmende im Kanal 2
N999 M30
```

Achse einem anderen Kanal übergeben (AXTOCHAN)

Aus einer Synchronaktion heraus kann mit dem Befehl AXTOCHAN eine Achse für einen beliebigen Kanal angefordert werden.

Ist die Achse bereits dem NC-Programm des Kanals zugeordnet ($AA_AXCHANGE_TYP[<Achse>] == 0) ergibt sich **keine** Zustandsänderung.

Wird eine Achse aus einer Synchronaktion für den eigenen Kanal angefordert, wird AXTOCHAN auf den Befehl GET abgebildet.

- Mit der **ersten** Anforderung für den eigenen Kanal, wird die Achse zur neutralen Achse.
- Mit der **zweiten** Anforderung wird die Achse dem NC-Programm zugeordnet.

Randbedingung

Eine "PLC kontrollierte Achse" entspricht einer "konkurrierenden Positionierachse" bei der besondere Randbedingungen zu beachten sind. Siehe dazu:

Literatur:

/FB2/ Funktionshandbuch Erweiterungsfunktionen; Positionierachsen (P2)

Hinweis

Eine PLC-Achse kann den Kanal nicht wechseln.

Eine ausschließlich von der PLC kontrollierte Achse kann nicht dem NC-Programm zugeordnet werden.

3.7.14 Verfahren von Spindeln(M, S, SPOS)

Spindeln können über Synchronaktionen gestartet, positioniert und gestoppt werden. Die Programmierung erfolgt im Aktionsteil der Synchronaktion mit der gleichen Syntax wie im Teileprogramm. Ohne numerische Erweiterung gelten die Befehle jeweils für die Masterspindel. Durch Angabe einer numerischen Erweiterung kann jede Spindel programmiert werden:

Programmcode	**Kommentar**
`ID = 1 EVERY $A_IN[1]==1 DO M3 S1000`	`; Master-Spindel`
`ID = 2 EVERY $A_IN[2]==1 DO SPOS=270`	`; Master-Spindel`
`ID = 1 EVERY $A_IN[1]==1 DO M1=3 S1=1000 SPOS[2]=90`	

Werden durch parallel aktive Synchronaktionen für eine Spindel konkurrierende Befehle vorgegeben, entscheidet die zeitliche Reihenfolge der Aktivierung.

Anwenderspezifische Spindelfreigabe

Der Start von Spindelbewegungen zu definierten Zeitpunkten kann über Synchronaktionen durch Blockieren der im Teileprogramm programmierten Bewegung erreicht werden.

Beispiel:

Die Spindel ist innerhalb eines Teileprogramms programmiert und soll nicht am Satzbeginn sondern erst starten, wenn der Eingang 1 gesetzt ist. Per Synchronaktion wird der Spindeloverride bis zur Freigabe über den Eingang 1 auf 0% gehalten. Siehe dazu Kapitel "Override ($A...OVR) (Seite 33)".

Programmcode
`; Solange Eingang 1 nicht gesetzt ist => Spindeloverride = 0%`
`ID=1 WHENEVER $A_IN[1]==0 DO $AA_OVR[S1]=0`
`...`
`; Der Start der Spindel wird ausgelöst`
`; Die Freigabe erfolgt wenn Eingang 1 gesetzt wird`
`G01 X100 F1000 M3 S1=1000`

Übergang zwischen Kommandoachse und Spindel

Da mehrere Synchronaktionen gleichzeitig aktiv sein können, ist es möglich, dass eine Spindelbewegung gestartet wird, während die Spindel bereits aktiv ist. In diesem Fall wird die zuletzt aktivierte Bewegung wirksam. Bei einer Umkehr der Bewegungsrichtung wird die Spindel zunächst abgebremst und dann in die entgegengesetzte Richtung verfahren.

Drehrichtung, Drehzahl und Position können auch während der Bewegung verändert werden.

Beispiele

Programmcode	**Kommentar**
`ID=1 EVERY $AC_TIMER[1] >= 5 DO M3 S300`	`; Drehrichtung und Drehzahl`
`ID=2 EVERY $AC_TIMER[1] >= 7 DO M4 S500`	`; Drehrichtung und Drehzahl`
`ID=3 EVERY $A_IN[1]==1 DO S1000`	`; Drehzahl`

Programmcode	Kommentar
`ID=4 EVERY ($A_IN[4]==1) AND ($A_IN[1]==0) DO SPOS=0`	`; Spindel positionieren`

Übergänge zwischen Achse und Spindel

Im Zustand↓	Nach →	POS	MOV<>0	MOV=0	SPOS	M3/M4	M5	LEADON	TRAIL-ON
wärend des Verfahrens									
	Achse	x	x	x	x	x	x	x	x
	lagegeregelte Spindel	x	x	x	x	x	x	-	-
	drehzahlgeregelte Spindel	-	-	-	x	x	x	-	-
in Bewegung									
	Achse	x	x	x	-	-	-	x	x
	lagegeregelte Spindel	-	-	-	-	-	-	-	-
	drehzahlgeregelte Spindel	-	-	-	x	x	x	-	-
Die mit x gekennzeichneten Übergänge sind zulässig. Die mit - gekennzeichneten Übergänge werden mit Alarm abgewiesen.									

Siehe auch

Kopplungen (CP..., LEAD..., TRAIL..., CTAB...) (Seite 101)

3.7.15 Freigabe für Achscontainer-Drehung zurücknehmen (AXCTSWEC)

Funktion

Mit dem Befehl `AXCTSWEC` kann eine bereits erteilte Freigabe zur Achscontainer-Drehung wieder zurückgenommen werden. Der Befehl löst einen Vorlaufstopp mit Reorganisieren (`STOPRE`) aus.

Damit im Kanal die Freigabe zur Achscontainer-Drehung wieder zurückgenommen wird, müssen folgende Bedingungen erfüllt sein:

- Im Kanal muss die Freigabe zur Achscontainer-Drehung bereits erfolgt sein:
  - `AXCTSWE(<Container>)`
  - `$AC_AXCTSWA[<Container>] == 1`
- Die Achscontainer-Drehung wurde noch nicht begonnen:
  - `$AN_AXCTSWA[<Container>] == 0`

Als Rückmeldung für die erfolgte Zurücknahme der Freigabe wird die folgende kanalspezifische Systemvariable zurückgesetzt:

`$AC_AXCTSWA[<Container>] == 0`

Eine ausführliche Beschreibung der Systemvariablen findet sich in:

Literatur
Listenhandbuch Systemvariablen

Syntax

```
DO AXCTSWEC(<Container>)
```

Bedeutung

AXCTSWEC: Freigabe zur Achscontainer-Drehung für den Kanal zurücknehmen

<Container>: Name des Achscontainers

Mögliche Angaben sind:

- CT<Containernummer>:
  An die Buchstabenkombination CT wird die Nummer des Achscontainers angehängt. Beispiel: CT3
- <Containername>:
  Mit MD12750 $MN_AXCT_NAME_TAB eingestellter individueller Name des Achscontainers. Beispiel: A_CONT3
- <Achsname>:
  Achsname einer Containerachse, die im Kanal bekannt ist.

Beispiel

Programmcode	**Kommentar**
; Initialisierung des globalen Zählers für den Technologiezyklus CTSWEC	
N100 $AC_MARKER[0]=0	
N110 ID=1 DO CTSWEC	; Technologiezyklus CTSWEC siehe unten.
NEXT:	
N200 G0 X30 Z1	
N210 G95 F.5	
N220 M3 S1000	
N230 G0 X25	
N240 G1 Z-10	
N250 G0 X30	
N260 M5	
; Freigabe der Achscontainer-Drehung für Containerspindel S1.	
N270 AXCTSWE(S1)	
N200 GOTO NEXT	

Programmcode	**Kommentar**
PROC CTSWEC(STRING _ex_CT="CT1"	
INT _ex_CTsl_BITmask=1H	
INT _ex_CT_SL_Number=1	
INT _ex_WAIT_number_of_IPOs=1000	
) DISPLOF ICYCOF	
DEFINE _ex_number_of_IPOs AS $AC_MARKER[0]	

```
Programmcode                          Kommentar
  IF ($AC_STOP_COND[0] + $AC_STOP_COND[1] + $AC_STOP_COND[2] + $AC_STOP_COND[3] +
      $AC_STOP_COND[4] + $AC_STOP_COND[5] + $AC_STOP_COND[6] + $AC_STOP_COND[7] +
      $AC_STOP_COND[8] + $AC_STOP_COND[9] + $AC_STOP_COND[10]) > 0)
    ; IPO-Taktzähler inkrementieren
    _ex_number_of_IPOs = _ex_number_of_IPOs + 1
    ; Wenn irgendeine Haltebedingung länger als "_ex_WAIT_number_of_IPOs"
    ; IPO-Takte vorliegt UND die Freigabe des eigenen Slots noch nicht erfolgt ist
    IF ( _ex_number_of_IPOs   >= _ex_WAIT_number_of_IPOs) AND
        ($AN_AXCTSWEC[_ex_CT] == _ex_CTsl_BITmask )
      AXCTSWEC                    ;   Freigabe der Achskontainerdrehung zurücknehmen.
    ENDIF
  ELSE
    ; IPO-Taktzähler zurücksetzen
    _ex_number_of_IPOs = 0
  ENDIF
RET
```

Randbedingung

Ausführzeitpunkt von Synchronaktionen

```
Programmcode
; Freigabe der Achscontainer-Drehung.
N10 AXCTSWE(CT3)
; Verfahren der Containerachse AX_A => Vor dem Verfahren wird auf
; das Ende der Achscontainer-Drehung gewartet: $AN_AXCTSWA[CT3]==0
N20 AX_A = 10
; Rücknahme der Freigabe. Keine Wirkung!
WHEN <Bedingung> DO AXCTSWEC(AX_A)
N30 G4 F1
```

Da nach der Freigabe der Achscontainer-Drehung in `N10`, in `N20` eine Achse des Achscontainers (`AX_A`) verwendet wird und diese Verwendung zu einem Warten auf das Ende der Achscontainer-Drehung führt, kommt die Synchronaktion erst zusammen mit dem Programmsatz `N30` in den Hauptlauf und ist somit wirkungslos.

Abhilfe:

```
Programmcode                                  Kommentar
; Freigabe der Achscontainer-Drehung.
N11 AXCTSWE(CT3)
; Rücknahme der Freigabe.
WHEN <Bedingung> DO AXCTSWEC(AX_A)
N21 ...                                       ;  Ausführbarer NC-Satz
; Verfahren der Containerachse AX_A => Vor dem Verfahren wird auf
; das Ende der Achscontainer-Drehung gewartet: $AN_AXCTSWA[CT3]==0
```

Programmcode	Kommentar
N31 AX_A = 10	

Hinweis

Ohne den ausführbaren Satz N21 gelangt die Synchronaktion erst nach dem Ende der Achscontainer-Drehung mit dem nächsten ausführbaren Programmsatz N31 in den Hauptlauf und wäre damit wie im obigen Beispiel ebenfalls wirkungslos.

3.7.16 Istwertsetzen mit Verlust des Referenzierstatus (PRESETON)

Funktion

Die Prozedur PRESETON() setzt aus Synchronaktionen heraus für **eine** Achsen neue Istwerte im Maschinenkoordinatensystem (MKS). Dies entspricht einer Nullpunktverschiebung des MKS der Achse. Die Achse wird dadurch nicht verfahren.

Aus Synchronaktionen heraus darf PRESETON **nur auf Kommandoachsen**, d.h. auf Achsen die aus einer Synchronaktion heraus gestartet wurden, angewandt werden. Weiter muss dem Kanal die Achse zugeordnet sein, d.h. dieser das Interpolationsrecht für diese Achse haben. Die Achse wird **nicht** per Achstausch von einem anderen Kanal angefordert.

Referenzierstatus

Durch das Setzen eines neuen Istwertes im Maschinenkoordinatensystem wird der Referenzierstatus der Maschinenachse zurückgesetzt:

DB31, ... DBX60.4 / .5 = 0 (Referenziert / Synchronisiert Messsystem 1 / 2)

Es wird empfohlen, PRESETON nur bei Achsen ohne Referenzpunktpflicht zu verwenden.

Zum Wiederherstellen des ursprünglichen Maschinenkoordinatensystems muss das Messsystem der Maschinenachse, z.B. durch Referenzpunktfahren aus dem Teileprogramm (G74), erneut referenziert werden.

⚠ VORSICHT

Verlust des Referenzierstatus

Durch das Setzen eines neuen Istwertes im Maschinenkoordinatensystem mit PRESETON wird der Referenzierstatus der Maschinenachse auf "nicht referenziert / synchronissiert" zurückgesetzt.

Programmierung

Syntax

```
WHEN | EVERY ... DO PRESETON(<Achse>, <Wert>)
```

Bedeutung

WHEN, EVERY:	Als Häufigkeit (Seite 15) darf nur WHEN und EVERY verwendet werden.
PRESETON:	Istwertsetzen mit Verlust des Referenzierstatus
<Achse>:	Maschinenachsname
	Typ: AXIS
	Wertebereich: im Kanal definierte Maschinenachsnamen
<Wert>:	Neuer Istwert der Maschinenachse im Maschinenkoordinatensystem (MKS)
	Die Eingabe erfolgt im aktuell gültigen Maßsystem (inch / metrisch)
	Eine aktive Durchmesserprogrammierung (DIAMON) wird berücksichtigt
	Typ: REAL

Systemvariable

$AC_PRESET

Die achsspezifische Systemvariable $AC_PRESET liefert den Vektor vom Nullpunkt des aktuellen verschobenen MKS' zum Nullpunkt des ursprünglichen MKS_0 nach dem Referenzieren der Maschinenachse.

$AC_PRESET<Achse> = $AC_PRESET<Achse> + "aktuelle Istposition der Achse im MKS" - "PRESETON-Istposition"

Mit Hilfe der Systemvariablen kann die Nullpunktverschiebung wieder rückgängig gemacht werden:

```
PRESETON(<Achse>, $VA_IM + $AC_PRESET[<Achse>]) ; "Aktueller Istwert
der Achse im MKS'" + "Verschiebungen"
```

Beispiel

Programmcode

```
N10 G1 X=10 F5000
; Fahre Achse X als Kommandoachse auf Position 200
N20 WHEN TRUE DO G71 POS[X]=200
; IF Sollposition der Achse X im MKS ($AA_IM[X]) >= 80
; THEN "Istposition der Achse X im MKS" = "Sollposition der Achse X im MKS"
+ "Verschiebung"
;                                    = 80 + 70 = 150
;      "Progr. Endposition der Achse X" = "Progr. Endposition der Achse X"
+ "Verschiebung"
;                                              = 200 + 70 = 270
;      $AC_PRESET = $AC_PRESET - 70
N30 WHEN G71 $AA_IM[X] >= 80 DO PRESETON(X, $AA_IM[X]+70)
N40 G4 F3
```

Randbedingungen

Achsen bei denen `PRESETON` nicht angewandt werden darf

- Fahrende Kommandoachsen im Spindelbetrieb
- Fahrende konkurrierende Positionierachsen (FC18)
- Achsen, die an einer Transformation beteiligt sind
- Fahrende Bahnachsen
- Pendelachsen
- Achsen, bei denen eine oder mehrere der folgenden sicheren Funktionen (Safety Integrated) aktiv ist:
  - Freigabe sichere Endschalter
    MD36901 $MA_SAFE_FUNCTION_ENABLE[<SafeAxis>], Bit 1 = 1
  - Freigabe sichere Nocken, Paar 1 ... 4, Nocke +/-
    MD36901 $MA_SAFE_FUNCTION_ENABLE[<SafeAxis>], Bit 8 ... 15 = 1
- Hirth-Achsen
- Gleichlaufachsen eines Gantry-Verbunds
- Achsen für die Referenzpunktfahren aus dem Teileprogramm (G74) aktiv ist
- Slave-Achse einer Drehzahl-/Drehmomentkopplung (Master-Slave)

Geometrieachsen

- PRESETON kann auf eine stehende Geometrieachse angewandt werden, wenn im Kanal nicht gleichzeitg eine weitere Geometrieachse verfährt.
- PRESETON kann auf eine stehende Geometrieachse angewandt werden, auch wenn im Kanal gleichzeitg eine weitere Geometrieachse verfährt, sich diese aber im Zustand "Neutrale Achse" befindet oder als Kommandoachse verfährt.
  Beispiel: Eine andere Geometrieachse (X) verfährt gleichzeitig im Zustand "**Neutrale Achse**"

Programmcode	Kommentar
`N10 G0 X0 Y0`	`; X, Y: Geometrieachsen`
`N15 RELEASE(Y)` [1)]	`; Neutrale Achse`
`N20 ID=1 WHEN 20.0 < $AA_IM[X] DO PRESETON(Y,20)`	`; $AA_IM: Sollposition im MKS`
`N30 G0 X40`	`; Geometrieachse X verfährt`
`N40 M30`	

1) **Hinweis**

Bei einer Freigabe der Achse im Aktionsteil der Synchronaktion ist nicht gewährleistet, dass die Freigabe rechtzeitig erfolgt.

`N20 ID=1 WHEN 20.0 < $AA_IM[X] DO RELEASE(Y) PRESETON(Y,20) ; NICHT empfohlen !`

Beispiel: Eine andere Geometrieachse (X) verfährt gleichzeitig als **Kommadoachse**

Programmcode	Kommentar
`N10 G0 X0 Y0`	`; X, Y: Geometrieachsen`
`N20 ID=1 WHEN TRUE DO POS[X]=40 FA[X]=1000`	`; Kommandoachse X`
`N30 ID=2 WHEN 20.0 < $AA_IM[X] DO PRESETON(Y,20)`	`; $AA_IM: Sollposition im MKS`
`N40 M30`	

PLC-kontrollierte Achsen

- PRESETON kann auf PLC-kontrollierte Achsen entsprechend ihres aktuellen Typs angewandt werden.

Spindelzustände

In der nachfolgenden Tabelle sind die Reaktionen aufgezeigt, die erfolgen, wenn PRESETON in einer Synchronaktion auf eine Spindel angewandt wird:

PRESETON in Synchronaktion			
Spindelbetriebsart	**Verfahrstatus**	**dem NC-Programm zugeordnet**	**Hauptlaufachse**
Drehzahlsteuerbetrieb	in Bewegung	Alarm 17601	Alarm 17601
	steht	+	+
Positionierbetrieb SPOS	Bewegung	Alarm 17601	Alarm 17601
	steht	+	+

PRESETON in Synchronaktion			
Spindelbetriebsart	**Verfahrstatus**	**dem NC-Programm zugeordnet**	**Hauptlaufachse**
Positionieren über Satzgrenzen SPOSA	in Bewegung	Alarm 17601	-
Achsbetrieb	in Bewegung	Alarm 17601	+
	steht	+	+
+: möglich -: nicht möglich			

PRESETON im NC-Programm			
Spindelbetriebsart	**Verfahrstatus**	**dem NC-Programm zugeordnet**	**Hauptlaufachse**
Drehzahlsteuerbetrieb	in Bewegung	Alarm 22324	Alarm 22324
	steht	+	+
Positionierbetrieb SPOS	Bewegung	-	+
	steht	+	+
Positionieren über Satzgrenzen SPOSA	in Bewegung	Alarm 10610	-
Achsbetrieb	in Bewegung	-	+
	steht	+	+
+: möglich -: nicht möglich			

Achskopplungen

- Leitachsen: Die durch PRESETON verursachte sprungförmige Änderung der Leitachsposition wird in den Folgeachsen nicht herausgefahren. Die Kopplung bleibt unverändert erhalten.
- Folgeachsen: Durch PRESETON wird nur der überlagerte Positionsanteil der Folgeachse beeinflusst.

Gantry-Verbund

- Wird PRESETON auf die Führungsachse eines Gantry-Verbunds angewandt, wird die Nullpunktverschiebung auch in allen Gleichlaufachsen des Gantry-Verbunds durchgeführt.

Teilungsachsen

- PRESETON kann auf Teilungsachsen angewandt werden.

Software-Endschalter, Arbeitsfeldbegrenzung, Schutzbereiche

- Liegt nach einer Nullpunktverschiebung durch PRESETON die Achsposition außerhalb einer der genannten Begrenzungen, wird erst dann ein Alarm angezeigt, wenn versucht wird die Achse zu verfahren.

Satzsuchlauf mit Berechnung

PRESETON-Befehle werden während des Suchlaufs aufgesammelt und mit NC-Start zum Fortsetzen des NC-Programms ausgeführt.

Positionsabhängige NC/PLC-Nahtstellensignale

- Der Status der positionsabhängigen NC/PLC-Nahtstellensignale wird aufgrund der neuen Istposition neu bestimmt.
  Beispiel: Festpunktpositionen
  - Parametrierte Festpunktpositionen: MD30600 $MA_FIX_POINT_POS[0...3] = <Festpunktposition 1...4>
  - NC/PLC-Nahtstellensignale: DB31, ... DBX75.3 ... 5 (JOG Festpunkt anfahren: erreicht)

  Steht die Achse innerhalb der Genauhalttoleranz auf einer Festpunktposition, wird das zugehörige NC/PLC-Nahtstellensignal gesetzt. Mit dem Setzen des Istwertes durch `PRESETON` auf einen anderen Wert ausserhalb der Genauhalttoleranz um die Festpunktposition, wird das Nahtstellensignal zurückgesetzt.

DRF-Verschiebung

- Durch `PRESETON` wird eine DRF-Verschiebung der Achse gelöscht.

Überlagerte Bewegung $AA_OFF

- Eine überlagerte Bewegung ($AA_OFF) (Seite 42) wird durch `PRESETON` nicht beeinflusst.

Online-Werkzeugkorrektur FTOC

- Eine aktive Online-Werkzeugkorrektur (FTOC) (Seite 67) bleibt auch nach `PRESETON` weiter aktiv.

Achsspezifische Kompensationen

Achsspezifische Kompensationen bleiben nach `PRESETON` weiter aktiv.

Betriebsart JOG

- `PRESETON` darf nur auf eine stehende Achsen angewandt werden.

Betriebsart JOG, Maschinenfunktion REF

- `PRESETON` darf **nicht** angewandt werden..

Siehe auch

Fliegendes Trennen (Seite 146)

3.7.17 Istwertsetzen ohne Verlust des Referenzierstatus (PRESETONS)

Funktion

Die Prozedur `PRESETONS()` setzt aus Synchronaktionen heraus für **eine** Achsen neue Istwerte im Maschinenkoordinatensystem (MKS). Dies entspricht einer Nullpunktverschiebung des MKS der Achse. Die Achse wird dadurch nicht verfahren.

Aus Synchronaktionen heraus darf `PRESETONS` **nur auf Kommandoachsen**, d.h. auf Achsen die aus einer Synchronaktion heraus gestartet wurden, angewandt werden. Weiter muss dem

Kanal die Achse zugeordnet sein, d.h. dieser das Interpolationsrecht für diese Achse haben. Die Achse wird **nicht** per Achstausch von einem anderen Kanal angefordert.

Referenzierstatus

Durch das Setzen eines neuen Istwertes im Maschinenkoordinatensystem (MKS) mit `PRESTONS` wird der Referenzierstatus der Maschinenachse **nicht** verändert.

Voraussetzungen

- **Gebertyp**
  `PRESETONS` ist nur bei folgenden Gebertypen des aktiven Messsystems möglich:
  - MD30240 $MA_ENC_TYPE[<Messsystem>] = 0 (Simulierter Geber)
  - MD30240 $MA_ENC_TYPE[<Messsystem>] = 1 (Rohsignalgeber)
- **Referenzier-Mode**
  `PRESETONS` ist nur bei folgenden Referenzier-Modes des aktiven Messsystems möglich:
  - MD34200 $MA_ENC_REFP_MODE[<Messsytem>] = 0 (kein Referenzpunktfahren möglich)
  - MD34200 $MA_ENC_REFP_MODE[<Messsytem>] = 1 (Referenzieren von inkrementellen, rotatorischen oder linearen Messsystemen: Nullimpuls auf der Geberspur)

Inbetriebnahme

Achsspezifische Maschinendaten

Istwertsetzen ohne Verlust des Referenzierstatus (`PRESETONS`) muss achsspezifisch aktiviert werden:

MD30460 $MA_BASE_FUNCTION_MASK, Bit 9 = 1

Hinweis

PRESETON deaktiviert

Mit dem Aktivieren der Funktion "Istwertsetzens ohne Verlust des Referenzierstatus `PRESETONS`" wird die Funktion "Istwertsetzen mit Verlust des Referenzierstatus(`PRESETON`" deaktiviert. Beide Funktionen schließen sich gegenseitig aus.

Programmierung

Syntax

```
<Häufigkeit> ... DO PRESETONS(<Achse>, <Wert>)
```

Bedeutung

`<Häufigkeit>`: Als Häufigkeit (Seite 15) darf nur `WHEN` und `EVERY` verwendet werden.
`PRESETONS`: Istwertsetzen ohne Verlust des Referenzierstatus

<Achse>:	Maschinenachsname	
	Typ:	AXIS
	Wertebereich:	im Kanal definierte Maschinenachsnamen
<Wert>:	Neuer aktueller Istwert der Maschinenachse im Maschinenkoordinatensystem (MKS)	
	Die Eingabe erfolgt im aktiven Maßsystem (inch / metrisch)	
	Eine aktive Durchmesserprogrammierung (DIAMON) wird berücksichtigt	
	Typ:	REAL

Systemvariable

$AC_PRESET

Die achsspezifische Systemvariable $AC_PRESET liefert den Vektor vom Nullpunkt des aktuellen verschobenen MKS' zum Nullpunkt des ursprünglichen MKS_0 nach dem Referenzieren der Maschinenachse.

$AC_PRESET<Achse> = $AC_PRESET<Achse> + "aktuelle Istposition der Achse im MKS" - "PRESETONS-Istposition"

Mit Hilfe der Systemvariablen kann die Nullpunktverschiebung wieder rückgängig gemacht werden:

```
PRESETONS(<Achse>, $VA_IM + $AC_PRESET[<Achse>]) ; "Aktueller
Istwert der Achse im MKS'" + "Verschiebungen"
```

Beispiel

Nullpunktverschiebung des MKS der Achse X um 70 Einheiten.

Die programmierte Endposition der Achse X (Kommandoachse) wird mit PRESETONS in das neue MKS transformiert.

Programmcode

```
N10 G1 X=10 F5000
; Fahre Achse X als Kommandoachse auf Position 200
N20 WHEN TRUE DO G71 POS[X]=200
; IF Sollposition der Achse X im MKS ($AA_IM[X]) >= 80
; THEN "Istposition der Achse X im MKS" = "Sollposition der Achse X im MKS"
+ "Verschiebung"
;                                    = 80 + 70 = 150
;      "Progr. Endposition der Achse X" = "Progr. Endposition der Achse X"
+ "Verschiebung"
;                                              = 200 + 70 = 270
;      $AC_PRESET = $AC_PRESET - 70
N30 WHEN G71 $AA_IM[X] >= 80 DO PRESETONS(X, $AA_IM[X]+70)
N40 G4 F3
```

Randbedingungen

Achsen bei denen `PRESETONS` nicht angewandt werden darf

- Fahrende Kommandoachsen im Spindelbetrieb
- Fahrende konkurrierende Positionierachsen (FC18)
- Achsen, die an einer Transformation beteiligt sind
- Fahrende Bahnachsen
- Pendelachsen
- Achsen, bei denen eine oder mehrere der folgenden sicheren Funktionen (Safety Integrated) aktiv ist:
  - Freigabe sichere Endschalter
    MD36901 $MA_SAFE_FUNCTION_ENABLE[<SafeAxis>], Bit 1 = 1
  - Freigabe sichere Nocken, Paar 1 ... 4, Nocke +/-
    MD36901 $MA_SAFE_FUNCTION_ENABLE[<SafeAxis>], Bit 8 ... 15 = 1
- Hirth-Achsen
- Gleichlaufachsen eines Gantry-Verbunds
- Achsen für die Referenzpunktfahren aus dem Teileprogramm (G74) aktiv ist
- Slave-Achse einer Drehzahl-/Drehmomentkopplung (Master-Slave)

Geometrieachsen

- PRESETONS kann auf eine stehende Geometrieachse angewandt werden, wenn im Kanal nicht gleichzeitg eine weitere Geometrieachse verfährt.
- PRESETONS kann auf eine stehende Geometrieachse angewandt werden, auch wenn im Kanal gleichzeitg eine weitere Geometrieachse verfährt, sich diese aber im Zustand "Neutrale Achse" befindet oder als Kommandoachse verfährt.
  Beispiel: Eine andere Geometrieachse (X) verfährt gleichzeitig im Zustand "**Neutrale Achse**"

Programmcode	Kommentar
`N10 G0 X0 Y0`	`; X, Y: Geometrieachsen`
`N15 RELEASE(Y)` [1)]	`; Neutrale Achse`
`N20 ID=1 WHEN 20.0 < $AA_IM[X] DO PRESETONS(Y,20)`	`; $AA_IM: Sollposition im MKS`
`N30 G0 X40`	`; Geometrieachse X verfährt`
`N40 M30`	

1) **Hinweis**

Bei einer Freigabe der Achse im Aktionsteil der Synchronaktion ist nicht gewährleistet, dass die Freigabe rechtzeitig erfolgt.

`N20 ID=1 WHEN 20.0 < $AA_IM[X] DO RELEASE(Y) PRESETONS(Y,20) ; NICHT empfohlen !`

Beispiel: Eine andere Geometrieachse (X) verfährt gleichzeitig als **Kommadoachse**

Programmcode	Kommentar
`N10 G0 X0 Y0`	`; X, Y: Geometrieachsen`
`N20 ID=1 WHEN TRUE DO POS[X]=40 FA[X]=1000`	`; Kommandoachse X`
`N30 ID=2 WHEN 20.0 < $AA_IM[X] DO PRESETONS(Y,20)`	`; $AA_IM: Sollposition im MKS`
`N40 M30`	

PLC-kontrollierte Achsen

- PRESETONS kann auf PLC-kontrollierte Achsen entsprechend ihres aktuellen Typs angewandt werden.

Spindelzustände

In der nachfolgenden Tabelle sind die Reaktionen aufgezeigt, die erfolgen, wenn PRESETONS in einer Synchronaktion auf eine Spindel angewandt wird:

PRESETONS in Synchronaktion			
Spindelbetriebsart	**Verfahrstatus**	**dem NC-Programm zugeordnet**	**Hauptlaufachse**
Drehzahlsteuerbetrieb	in Bewegung	Alarm 17601	Alarm 17601
	steht	+	+
Positionierbetrieb SPOS	Bewegung	Alarm 17601	Alarm 17601
	steht	+	+

PRESETONS in Synchronaktion			
Spindelbetriebsart	**Verfahrstatus**	**dem NC-Programm zugeordnet**	**Hauptlaufachse**
Positionieren über Satzgrenzen SPOSA	in Bewegung	Alarm 17601	-
Achsbetrieb	in Bewegung	Alarm 17601	+
	steht	+	+
+: möglich -: nicht möglich			

PRESETONS im NC-Programm			
Spindelbetriebsart	**Verfahrstatus**	**dem NC-Programm zugeordnet**	**Hauptlaufachse**
Drehzahlsteuerbetrieb	in Bewegung	Alarm 22324	Alarm 22324
	steht	+	+
Positionierbetrieb SPOS	Bewegung	-	+
	steht	+	+
Positionieren über Satzgrenzen SPOSA	in Bewegung	Alarm 10610	-
Achsbetrieb	in Bewegung	-	+
	steht	+	+
+: möglich -: nicht möglich			

Achskopplungen

- Leitachsen: Die durch PRESETONS verursachte sprungförmige Änderung der Leitachsposition wird in den Folgeachsen nicht herausgefahren. Die Kopplung bleibt unverändert erhalten.
- Folgeachsen: Durch PRESETONS wird nur der überlagerte Positionsanteil der Folgeachse beeinflusst.

Gantry-Verbund

- Wird PRESETONS auf die Führungsachse eines Gantry-Verbunds angewandt, wird die Nullpunktverschiebung auch in allen Gleichlaufachsen des Gantry-Verbunds durchgeführt.

Teilungsachsen

- PRESETONS kann auf Teilungsachsen angewandt werden.

Software-Endschalter, Arbeitsfeldbegrenzung, Schutzbereiche

- Liegt nach einer Nullpunktverschiebung durch PRESETONS die Achsposition außerhalb einer der genannten Begrenzungen, wird erst dann ein Alarm angezeigt, wenn versucht wird die Achse zu verfahren.

Satzsuchlauf mit Berechnung

PRESETONS-Befehle werden während des Suchlaufs aufgesammelt und mit NC-Start zum Fortsetzen des NC-Programms ausgeführt.

Positionsabhängige NC/PLC-Nahtstellensignale

- Der Status der positionsabhängigen NC/PLC-Nahtstellensignale wird aufgrund der neuen Istposition neu bestimmt.
  Beispiel: Festpunktpositionen
  - Parametrierte Festpunktpositionen: MD30600 $MA_FIX_POINT_POS[0...3] = <Festpunktposition 1...4>
  - NC/PLC-Nahtstellensignale: DB31, ... DBX75.3 ... 5 (JOG Festpunkt anfahren: erreicht)

  Steht die Achse innerhalb der Genauhalttoleranz auf einer Festpunktposition, wird das zugehörige NC/PLC-Nahtstellensignal gesetzt. Mit dem Setzen des Istwertes durch `PRESETONS` auf einen anderen Wert ausserhalb der Genauhalttoleranz um die Festpunktposition, wird das Nahtstellensignal zurückgesetzt.

DRF-Verschiebung

- Durch `PRESETONS` wird eine DRF-Verschiebung der Achse gelöscht.

Überlagerte Bewegung $AA_OFF

- Eine überlagerte Bewegung ($AA_OFF) (Seite 42) wird durch `PRESETONS` nicht beeinflusst.

Online-Werkzeugkorrektur FTOC

- Eine aktive Online-Werkzeugkorrektur (FTOC) (Seite 67) bleibt auch nach `PRESETONS` weiter aktiv.

Achsspezifische Kompensationen

Achsspezifische Kompensationen bleiben nach `PRESETONS` weiter aktiv.

Betriebsart JOG

- `PRESETONS` darf nur auf eine stehende Achsen angewandt werden.

Betriebsart JOG, Maschinenfunktion REF

- `PRESETONS` darf **nicht** angewandt werden..

3.7.18 Kopplungen (CP..., LEAD..., TRAIL..., CTAB...)

In Synchronaktionen können für die Funktionen Mitschleppen (`TRAIL...`), Kurventabellen (`CTAB...`), Leitwertkopplung (`LEAD...`) und Generische Kopplung (`CP...`) die in Kapitel "Sprachelemente für Synchronaktionen und Technologiezyklen (Seite 53)" aufgeführten Befehle programmiert werden:

Hinweis

Generische Kopplung

Für Befehle der "Generischen Kopplung" `CP...` ist zu beachten, dass diese in Synchronaktionen immer in der Reihenfolge der Programmierung von links nach rechts abgearbeitet werden. D. h. im Gegensatz zur Programmierung im Teileprogramm ist die Wirkung der verschiedenen Befehle von der Reihenfolge in der Synchronaktion abhängig.

Kurventabellen

Die Befehle `CTAB` und `CTABINV` können in der Bedingung und in der Aktion verwendet werden.

Literatur

Ausführliche Informationen zu den Koppelbefehlen finden sich in:

- Mitschleppen, Kurventabellen, Leitwertkopplung:
  Programmierhandbuch Arbeitsvorbereitung, Kapitel "Achskopplungen"
- Generische Kopplung
  Funktionsbeschreibung Sonderfunktionen, Kapitel "Achskopplungen (M3)" > "Generische Kopplung"

Mitschleppen

Beim Einschalten der Kopplung aus der Synchronaktion kann die Leitachse in Bewegung sein. Die Folgeachse wird in diesem Fall auf die Sollgeschwindigkeit beschleunigt. Die Position der Leitachse zum Synchronisationszeitpunkt der Geschwindigkeiten ist Startposition für das Mitschleppen.

Leitwertkopplung

Syntax

```
... DO LEADON(<FA>, <LA>, <NR>, <OVW>)
```

Bedeutung

`<FA>`:	Name der Folgeachse Typ: AXIS
`<LA>`:	Name der Leitachse Typ: AXIS

<NR>:	Nummer der Kurventabelle Typ: INT
<OVW>:	Status der Überschreiberlaubnis Typ: BOOL 0: Das Überschreiben der Tabelle ist **nicht** erlaubt 1: Das Überschreiben der Tabelle ist erlaubt

- Über Synchronaktionen kann auch während einer aktiven Leitwertkopplung die zugrundeliegende Kurventabelle ohne Neusynchronisation geändert werden.
  Die Folgeachse versucht dabei schnellst möglich dem durch die neue Kurventabelle vorgegebenen Positionswerte zu folgen.
- Um eine zu koppelnde Achse über Synchronaktionen programmieren zu können, muss die Achse zuvor mit RELEASE freigegeben werden.
  Beispiel:

```
Programmcode
...
N60 RELEASE(X)
N50 ID=1 EVERY SR1==1 DO LEADON(C, X, 1)
```

Beispiel: Fliegendes Trennen

Ein Strangmaterial, das sich stetig durch einen Arbeitsbereich einer Trennvorrichtung bewegt, soll in gleichlange Stücke zerteilt werden.

- X-Achse: Achse in der sich das Strangmaterial bewegt (WKS)
- X1-Achse: Maschinenachse des Strangmaterials (MKS)
- Y-Achse: Achse, in der die Trennvorrichtung mit dem Strangmaterial "mitfährt"

Es wird angenommen, dass die Zustellung des Trennwerkzeuges und seine Steuerung über das PLC-Anwenderprogramm kontrolliert werden. Zur Feststellung der Synchronität zwischen Strangmaterial und Trennwerkzeug können die Signale der PLC-Nahtstelle ausgewertet werden.

```
Programmcode                        Kommentar
N100 R3=1500                        ;   Länge eines abzutrennenden Teiles
N200 R2=100000 R13=R2/300
N300 R4=100000
N400 R6=30                          ;   Startposition Y Achse
N500 R1=1                           ;   Startbedingung für Bandachse
N600 LEADOF(Y,X)                    ;   Kopplung löschen
N700 CTABDEF(Y,X,1,0)               ;   Tabellendefinition
N800 X=30 Y=30                      ;   Wertepaare
N900 X=R13 Y=R13
N1000 X=2*R13 Y=30
N1100 CTABEND                       ;   Ende der Tabelledefinition
N1200 PRESETON(X1,0)                ;   PRESET zu Beginn
N1300 Y=R6 G0                       ;   Startposition Y Achse, Achse ist linear
```

Programmcode	Kommentar
; PRESET nach Länge R3, neuer Beginn nach Abtrennen	
N1400 ID=1 WHENEVER $AA_IW[X]>$R3 DO PESETON(X1,0)	
N1500 RELEASE(Y)	
; Y über Tabelle 1 an X ankoppeln bei X <10	
N1800 ID=6 EVERY $AA_IM[X]<10 DO LEADON(Y,X,1)	
; > 30 vor gefahrener Trennlänge abkoppeln	
N1900 ID=10 EVERY $AA_IM[X]>$R3-30 DO LEADOF(Y,X)	
N2000 WAITP(X)	
; Strangachse stetig in Bewegung setzen	
N2100 ID=7 WHEN $R1==1 DO MOV[X]=1 FA[X]=$R4	
N2200 M30	

Generische Kopplung

- Bei der Aktivierung eines Koppelmoduls in einer Synchronaktion **muss** die Folgeachse bereits im Kanal aktiv sein und sich im Zustand "Neutrale Achse oder "Achse bereits dem Teileprogramm des Kanals zugeordnet" befinden. Der entsprechende Achszustand kann, falls erforderlich, in der Synchronaktion durch Programmierung von `GET[<Folgeachse>]` erzeugt werden.
- Die Befehle der generischen Kopplung `CP...` werden in Synchronaktionen direkt durch das Koppelmodul verarbeitet. Damit wird der Befehl sofort wirksam.
- Mit der Programmierung eines Koppelfaktors (`CPLNUM`, `CPLDEN`) oder Tabellennummer (`CPLCTID`) wird ein zuvor aktiviertes nichtlineares Koppelverhältnis, z.B. einer Kurventabelle, deaktiviert.

Generische Kopplung: Kopplungstyp TRAIL, LEAD, EG oder COUP verwenden.

Wird im Rahmen der generischen Kopplung ein Verhalten entsprechend einer der bekannten Kopplungstypen "Mitschleppen", "Leitwertkopplung", "Elektronisches Getriebe" oder "Synchronspindel" gewünscht, ist in Synchronaktionen beim Anlegen oder Definieren des Koppelmoduls zusätzlich der Befehl `CPSETTYPE` möglich:

```
CPSETTYPE[FAx] = <Kopplungstyp>
```

<Kopplungstyp>	Bedeutung
CP	Freie Programmierbarkeit
TRAIL	Kopplungsart "Mitschleppen"
LEAD	Kopplungsart "Leitwertkopplung"
EG	Kopplungsart "Elektronisches Getriebe"
COUP	Kopplungsart "Synchronspindel"

Randbedingungen

Synchronlaufstatus einer Folgeachse

Über die Systemvariable $AA_SYNC[<Achse>] kann im Teileprogramm oder Synchronaktion der Synchronlaufstatus einer Folgeachse gelesen werden.

Achstausch bei kanalübergreifender Kopplung

Beim Achstausch müssen die Folge- und Leitachsen dem aufrufenden Kanal bekannt sein. Der Achstausch der Leitachsen ist unabhängig vom Zustand der Kopplung möglich. Durch eine definierte oder aktive Kopplung entstehen keine weiteren Randbedingungen.

Hinweis

Durch die Aktivierung der Kopplung wird die Folgeachse zur Hauptlaufachse und steht für einen Achstausch nicht zur Verfügung. Damit wird die Folgeachse aus dem Kanal abgemeldet. Eine überlagerte Bewegung ist bei dieser Art von Kopplung somit nicht möglich.

Siehe auch Kapitel "Achstausch (GET, RELEASE, AXTOCHAN) (Seite 79)"

Konfliktvermeidung beim Wechsel von Folgeachse zu Kanalachse

Um eine über Synchronaktionen verfahrene Folgeachse wieder als Kanalachse verfahren zu können, muss sichergestellt werden, dass die Kopplung ausgeschaltet ist, bevor der Kanal die betreffende Achse anfordert.

Das folgende Beispiel zeigt einen **Fehlerfall**:

Programmcode

```
...
N50 WHEN TRUE DO TRAILOF(Y, X)
N60 Y100
```

Die Y-Achse wird in `N50` nicht rechtzeitig freigeben, da `TRAILOF` durch die satzweise Synchronaktion erst mit `N60` aktiv wird.

Korrigiertes Beispiel:

Programmcode **Kommentar**

```
...
N50 WHEN TRUE DO TRAILOF(Y, X)
N55 WAITP(Y)             ; Warten auf Verfahrende der Positionierachse
N60 Y100
```

Beispiele

Kopplung definieren: Y = Folgeachse, X = Leitachse

Programmcode

```
... DO CPLDEF[Y]=X CPLNUM[Y,X]=1.5
```

Kopplung einschalten und Koppelverhältnis definieren.

- N10 mit korrekter Reihenfolge: zuerst `CPLON`, danach `CPLNUM`
- N20 mit **falscher** Reihenfolge: zuerst `CPLNUM`, danach CPLON

Programmcode

```
N10 ... DO CPLON[Y]=X CPLNUM[X,Y]=1.5
```

Programmcode

```
N20 ... DO CPLNUM[X,Y]=2 CPLON[Y]=X              ; Fehler
```

Kopplung einschalten, aus/einschalten mit impliziter Neusynchronisation

Programmcode

```
N10 ... DO CPLON[X]=Y CPLNUM[X,Y]=3
N20 Y100 F100
N30 ... DO CPLOF=X CPLON[X]=Y CPLNUM[X,Y]=3
```

Kopplung einschalten, ausschalten und als Kommandoachse verfahren

Programmcode

```
N10 ... DO CPLON[X]=Y CPLNUM[X,Y]=3
N20 Y100 F100
N30 ... DO CPLOF=X MOV[X]=10
```

3.7.19 Messen (MEAWA, MEAC)

Zum Messen können in Synchronaktionen folgende Befehle verwendet werden:

- `MEAWA` (Messen ohne Restweglöschen)
- `MEAC` (Kontinuierliches Messen ohne Restweglöschen)

Während die Messfunktion im Teileprogramm auf einen Bewegungssatz begrenzt ist, kann die Messfunktion aus Synchronaktionen heraus beliebig ein- und ausgeschaltet werden.

Hinweis

Über statischen Synchronaktionen `IDS...` kann auch in der Betriebsart JOG gemessen werden.

Literatur

Ausführliche Informationen zu den Messbefehlen finden sich in:

- Mitschleppen, Kurventabellen, Leitwertkopplung:
  Programmierhandbuch Arbeitsvorbereitung, Kapitel "Achskopplungen"
- Generische Kopplung
  Funktionsbeschreibung Sonderfunktionen, Kapitel "M3 Achskopplungen" > "Generische Kopplung"

Messaufträge und Zustandsänderungen

Wenn der Messauftrag aus Synchronaktionen erfolgt ist, zeigt die Steuerung folgendes Verhalten:

Zustand	Verhalten
Betriebsartenwechsel	Ein Messauftrag, der durch eine modale Synchronaktion aktiviert wurde, wird durch den Betriebsartenwechsel nicht beeinflusst. Er bleibt über Satzgrenzen hinweg wirksam.
RESET	Der Messauftrag wird abgebrochen
Satzsuchlauf	Die Messaufträge werden gesammelt und erst bei Erfüllung der programmierten Bedingung aktiviert
REPOS	Aktivierte Messaufträge werden nicht beeinflusst.
Programmende	Messaufträge, die aus statischen Synchronaktionen gestartet wurden, bleiben erhalten.

Anmerkungen

Systemvariable

Folgende Systemvariablen können im Zusammenhang mit Synchronaktionen verwendet werden:

- $AA_MEAACT (Axiales Messen aktiv)
- $A_PROBE (Messtasterzustand)
- $AA_MM1 ... 4 (Messtasterposition 1. bis 4. Trigger (MKS))

Folgende Systemvariable kann im Zusammenhang mit Synchronaktionen **nicht** verwendet werden:

- $AC_MEA (Messtaster hat geschaltet)

Messauftrag

Pro Achse darf nur ein Messauftrag aktiv sein.

Priorität bei mehreren Messungen

Ein erneuter Messauftrag für die gleiche Achse bewirkt, dass die Triggerereignisse erneut aktiviert und die Messergebnisse zurückgesetzt werden.

Messaufträge, die aus dem Teileprogramm gestartet wurden, können aus Synchronaktionen nicht beeinflusst werden. Wird aus einer Synchronaktion ein Messauftrag für eine Achse gestartet für die bereits ein Messauftrag aus dem Teileprogramm aktiv ist, wird ein Alarm angezeigt.

Ist ein Messauftrag aus einer Synchronaktion aktiv, kann Messen aus dem Teileprogramm heraus nicht mehr gestartet werden.

Speichern von Messergebnissen

Zum Speichern von Messergebnissen wird in den Systemvariablen $AC_FIFO ein FIFO-Speicher eingerichtet. Siehe dazu Kapitel "FIFO-Variablen ($AC_FIFO) (Seite 29)".

Beispiele

Für die folgenden Beispiele werden über Maschinendaten 2 FIFO-Speicher eingerichtet:

- MD28050 $MC_MM_NUM_R_PARAM = 300
- MD28258 $MC_MM_NUM_AC_TIMER = 1
- MD28260 $MC_NUM_AC_FIFO = 1 (FIFO-Speicher einrichten)
- MD28262 $MC_START_AC_FIFO = 100 (FIFO-Speicher beginnt ab R100)
- MD28264 $MC_LEN_AC_FIFO = 28 (22 Variablen + 6 Verwaltungsdaten)
- MD28266 $MC_MODE_AC_FIFO = 0 (keine Summenbildung)

Beispiel 1

Für die X-Achse sollen zwischen 0 und 100 mm alle steigenden Flanken von Messtaster 1 erfasst werden. Es wird dabei vorausgesetzt, dass nicht mehr als 22 Messflanken auftreten.

```
Programmcode                                    Kommentar
DEF INT ANZAHL                                  ;   Anzahl aktueller Messwerte
DEF INT INDEX_R                                 ;   Schleifenindex
N10 G0 X0                                       ;   Startpunkt der Messung anfahren
; Messen: Modus=1 (gleichzeitig), FIFO-Speicher=1,
; Triggerereignis=1 (steigende Flanke von Messtaster 1)
N20 MEAC[X]=( 1, 1, 1) POS[X]=100
N30 STOPRE                                      ;   Anhalten der Vorverarbeitung
N40 MEAC[X]=(0)                                 ;   Messung abbrechen
N50 ANZAHL=$AC_FIFO1[4]                         ;   Anzahl gespeicherter Messwerte
N60 ANZAHL = ANZAHL - 1
N70 FOR INDEX_R=0 TO ANZAHL
N80   R[INDEX_R]=$AC_FIFO1[0]                   ;   Messwert in R-Par. speichern
N90 ENDFOR
```

Beispiel 2

Für die X-Achse sollen zwischen 0 und 100 mm alle steigenden und fallenden Flanken von Messtaster 1 erfasst werden. Die Anzahl der Messungen ist nicht bekannt. Daher müssen parallel zur Messung die Messwerte abgeholt und ab $R1 aufsteigend abgelegt werden. Die Anzahl der abgelegten Messwerte wird in $R0 eingetragen.

```
Programmcode
$AC_MARKER[1]=1                                 ;   Index für R-Parameter-Index initialisieren
N10 G0 X0                                       ;   Startpunkt der Messung anfahren
; liegt ein Messwerte im FIFO-Speicher vor, wird der ältesten Wert gelesen und im
; aktuelle R-Parameter[$AC_MARKER[1]] ablegt.
; Anschließend wird der R-Parameterindex inkrementiert.
N20 ID=1 WHENEVER $AC_FIFO1[4] >= 1 DO $R[$AC_MARKER[1]] = $AC_FIFO1[0]
                                      $AC_MARKER[1] = $AC_MARKER[1] + 1
```

Programmcode

```
; Kontinuierliches Messen: Modus=1 (gleichzeitig), FIFO-Speicher=1,
; Triggerereignis 1=1 (steigende Flanke von Messtaster 1),
; Triggerereignis 2=-1 (fallende Flanke von Messtaster 1)
N30 MEAC[X]=(1, 1, 1, -1) POS[X]=100
N40 MEAC[X]=(0)                          ;   Messung ausschalten
N50 STOPRE                               ;   Anhalten der Vorverarbeitung
N60 R0 = $AC_MARKER[1]                   ;   Anzahl der erfassten Messwerte
```

Beispiel 3

Für die X-Achse sollen zwischen 0 und 500 mm steigende und fallende Flanken von Messtaster 1 erfasst werden. Die Anzahl der Messungen wird auf 10 beschränkt.

Abschließend wird der Restweg der X-Achse gelöscht.

Programmcode

```
N10 G0 X0                                ;   Startpunkt der Messung anfahren
; Abbruchbedingung: Nach 10 oder mehr Messungen kontinuierliche Messung abwählen
; und "Restweg löschen" durchführen
N10 WHEN $AC_FIFO1[4] >= 10 DO MEAC[X]=(0) DELDTG(X)
; Kontinuierliches Messen: Modus=1 (gleichzeitig), FIFO-Speicher=1,
; Triggerereignis 1=1 (steigende Flanke von Messtaster 1),
; Triggerereignis 2=-1 (fallende Flanke von Messtaster 1)
N20 MEAC[X]=(1, 1, 1, -1) G01 X100 F500
N30 MEAC[X]=(0)                          ;   Messung ausschalten
N40 R0 = $AC_FIFO1[4]                    ;   Anzahl der erfassten Messwerte
```

3.7.20 Fahren auf Festanschlag (FXS, FXST, FXSW, FOCON, FOCOF, FOC)

Funktion

Fahren auf Festanschlag

Über die Befehle `FXS`, `FXST` und `FXSW` kann über Synchronaktionen die Funktion "Fahren auf Festanschlag" gesteuert werden.

Die Aktivierung kann auch ohne Verfahrbewegung der betreffenden Achse erfolgen. Das Moment wird sofort begrenzt. Auf Anschlag wird überwacht, sobald die Achse verfahren wird.

Fahren mit begrenztem Moment/Kraft

Über die Befehle `FOCON`, `FOCOF` und `FOC` kann über Synchronaktionen das Verfahren mit begrenztem Moment/Kraft gesteuert werden.

Syntax

```
FXS[<Achse>]=<Anforderung>
FXST[<Achse>]=<Klemmmoment>
```

```
FXSW[<Achse>]=<Fensterbreite>
FOCON[<Achse>]
FOCOF[<Achse>]
FOC[<Achse>]
```

Bedeutung

Parameter	Bedeutung
FXS:	Fahren auf Festanschlag
<Anforderung>:	Anforderung an die Funktion "Fahren auf Festanschlag": 0 = Ausschalten 1 = Einschalten
FXST:	Klemmmoment einstellen
<Klemmmoment>:	Klemmmoment in % vom maximalen Moment des Antriebs
FXSW:	Überwachungsfenster einstellen
<Fensterbreite>:	Breite des Toleranzfensters um den Festanschlag Einheit: mm, inch oder Grad
FOCON:	Modale Moment/Kraft-Begrenzung einschalten
FOCOF:	Modale Moment/Kraft-Begrenzung ausschalten
FOC:	Satzweise wirksame Moment/Kraft-Begrenzung
<Achse>:	Name der Kanalachse auf die der Befehl angewandt wird

Anmerkungen

Vermeidung von Mehrfachanwahl

Die Funktion "Fahren auf Festanschlag" darf pro Achse nur einmal eingeschaltet werden. Im Fehlerfall wird der Alarm 20092 angezeigt und die entsprechenden Alarmreaktion wirksam.

Zur Vermeidung von Mehrfachanwahlen wird die Verwendung eine Anwahlmerkers in der Synchronaktion empfohlen.

Beispiel:

```
Programmcode                                    Kommentar
N10 R1=0                                        ;   Anwahlmerker initialisieren
...
N20 IDS=1 WHENEVER ($R1==0 AND $AA_IW[AX3] > 7) DO $R1=1 FXS[AX1]=1
```

Einschalten während der Anfahrbewegung

Durch eine satzbezogenen Synchronaktion kann "Fahren auf Festanschlag" auch während der Anfahrbewegung eingeschaltet werden.

Beispiel:

```
Programmcode                                    Kommentar
N10 G0 G90 X0 Y0                                ;   Grundstellung anfahren
...
```

Programmcode	Kommentar
; "Fahren auf Festanschlag" wird für die X-Achse eingeschaltet,	
; sobald die Sollposition im WKS > 20 mm ist	
; Ausführung der satzweisen Synchronaktion: mit N30	
N20 WHEN G71 $AA_IW[X] > 20 DO FXS[X]=1	
N30 G1 F200 X100	; Verfahrsatz der X-Achse

Beispiel: Fahren auf Festanschlag vollständig über Synchronaktionen

```
; WENN Anwahlanforderung $R1==1 UND Zustand der Y-Achse == "nicht auf Anschlag"
; DANN: Für die Y-Achse:
;  - FXS einschalten
;  - Verfahren auf Position 150 mm
;  - Antriebsmoment auf 10 % reduzieren
IDS=1 WHENEVER G71 (($R1==1) AND $AA_FXS[y]==0)) DO $R1=0 FXS[Y]=1 FXST[Y]=10
                                                    FA[Y]=200 POS[Y]=150
...
; WENN Zustand der Y-Achse == "Anschlag wurde erkannt"
; DANN: Antriebsmoment auf 30 % erhöhen
IDS=2 WHENEVER ($AA_FXS[Y]==4) DO FXST[Y]=30
...
; WENN Zustand der Y-Achse == "Anschlag wurde erfolgreich angefahren"
; DANN: Antriebsmoment entsprechend Vorgabe $R0 einstellen
IDS=3 WHENEVER ($AA_FXS[Y]==1) DO FXST[Y]=$R0
...
; Abwahl in Abhängigkeit von R3 und zurückfahren.
IDS=4 WHENEVER (($R3==1) AND $AA_FXS[Y]==1)) DO FXS[Y]=0 FA[Y]=1000 POS[Y]=0
...
N10 R1=0 FXS[Y]=0 G0 G90 Y0          ;   Initialisierung
N30 RELEASE(Y)                       ;   Y-Achse zum Verfahren in Synchronaktionen freigeben
N50 ...
N60 GET(Y)                           ;   Y-Achse wieder in den Bahnverbund aufnehmen
```

3.7.21 Kanalsynchronisation (SETM, CLEARM)

Mit den Befehlen `SETM` und `CLEARM` können Synchronisationsmarken im Kanal in dem die Synchronaktion abläuft, gesetzt und gelöscht werden.

Syntax

```
SETM(<Nr_Marker 1> [,<Nr_Marker 2> {, ... < Nr_Marker n>} ] )
CLEARM(<Nr_Marker 1> [,<Nr_Marker 2> {, ... < Nr_Marker n>} ] )
```

Bedeutung

Eine ausführliche Beschreibung der Befehle `SETM` und `CLEARM` findet sich in:

Literatur

Programmierhandbuch Arbeitsvorbereitung, Kapitel "Flexible NC-Programmierung" > "Programmkoordinierung (INIT, START, WAITM, WAITMC, WAITE, SETM, CLEARM)"

3.7.22 Anwenderspezifische Fehlerreaktionen (SETAL)

Über Synchronaktionen kann auf applikationsspezifische Fehlerzustände anwenderspezifisch reagiert werden. Mögliche Reaktionen sind:

- Achse mit stoppen über Override = 0%
- Anwenderspezifischen Alarm anzeigen
- Digitalen Ausgang setzen

Alarm anzeigen

Syntax

```
SETAL(<Alarm_Nr>[,"Alarmtext"])
```

Bedeutung

Parameter	Bedeutung
`<Alarm_Nr>`:	Alarmnummer aus dem Bereich: 65000 - 69999

Eine vollständige Beschreibung zur Projektierung von Anwender-Alarmen findet sich in:

Literatur

Inbetriebnahmehandbuch Basesoftware und HMI-Advanced,
Kapitel "HMI-Advanced" > "HMI-System konfigurieren" > "Anwender-Alarme projektieren"

Beispiele

```
; Wird der Abstand zwischen Achsen X1 und X2 kleiner 5 mm =>
; Achse X2 anhalten
ID=1 WHENEVER G71 ($AA_IM[X1]-$AA_IM[X2])<5.0 DO $AA_OVR[X2]=0

; Wird der Abstand zwischen Achsen X1 und X2 kleiner 5 mm =>
; Alarm 65000 anzeigen
ID=1 WHENEVER G71 ($AA_IM[X1]-$AA_IM[X2])<5.0 DO SETAL(65000)
```

3.8 Technologiezyklen

3.8.1 Allgemein

Definition

Als Technologiezyklus wird ein Unterprogramm bezeichnet, das im Aktionsteil einer Synchronaktion aufgerufen wird. In einem Technologiezyklus dürfen alle Sprachelemente und Systemvariablen verwendet werden, die auch im Aktionsteil einer Synchronaktion verwendet werden können. Darüber hinaus gibt es folgende zusätzliche Sprachelemente, die nur innerhalb eines Technologiezyklus verwendet werden dürfen:

- Kapitel "Systemvariable für Synchronaktionen (Seite 19)"
- Kapitel "Anwenderdefinierte Variablen für Synchronaktionen (Seite 52)"
- Kapitel "Sprachelemente für Synchronaktionen und Technologiezyklen (Seite 53)"
- Kapitel "Sprachelemente nur für Technologiezyklen (Seite 59)"
- Kapitel "Aktionen in Synchronaktionen (Seite 60)"

Programmende

Als Programmende sind folgende Befehle erlaubt: `M02`, `M17`, `M30`, `RET`

Suchpfad

Beim Aufruf eines Technologiezyklus wird der gleiche Suchpfad wie bei Unterprogrammen und Zyklen verwendet.

Literatur
Programmierhandbuch Arbeitsvorbereitung, Kapitel "Flexible NC-Programmierung" > "Unterprogrammtechnik" > "Allgemeines" > "Suchpfad"

Mehrfachaufrufe

Wird eine Bedingung, während der Technologiezyklus abgearbeitet wird, erneut erfüllt, wird der Technologiezyklus **nicht** erneut gestartet.

Wird ein Technologiezyklus aufgrund einer erfüllten Bedingung `WHENEVER` gestartet und ist die Bedingung nach Abschluss des Technologiezyklus immer noch erfüllt, wird der Technologiezyklus erneut gestartet.

Verhalten bei satzweisen Synchronaktionen

Eine satzweise Synchronaktion ist immer an den nächsten Hauptlaufsatz gebunden. Ist die Abarbeitungszeit des Technologiezyklus länger als die Bearbeitungszeit des zugehörigen Hauptlaufsatzes, wird der Technologiezyklus mit dem Satzwechsel abgebrochen.

Abarbeitungsreihenfolge von Technologiezyklen

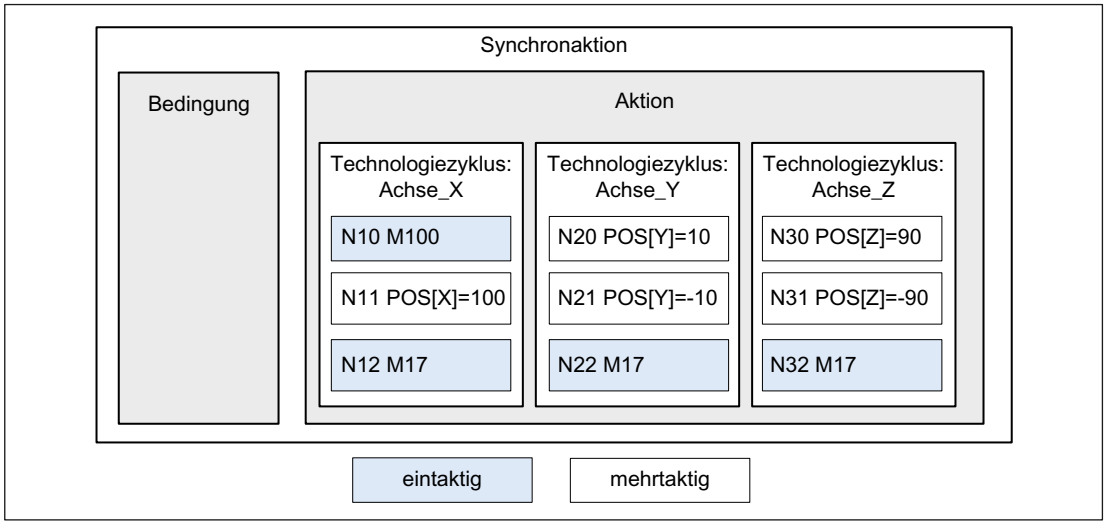
Sind im Aktionsteil einer Synchronaktion mehrere Technologiezyklen programmiert, werden sie in der Reihenfolge von links nach rechts abgearbeitet.

Beispiel:

Aufruf von drei Technologiezyklen im Aktionsteil eines Technologiezyklus

Programmcode

```
ID=1 <Bedingungsteil> DO ACHSE_X ACHSE_Y ACHSE_Z
```



Ausführungsreihenfolge der Sätze der Technologiezyklen: `N10`, `N11`, `N12`, `N20`, `N21`, `N22`, `N30`, `N31`, `N32`

Hinweis

Randbedingungen

- Im Aktionsteil einer Synchronaktion dürfen maximal 8 Technologiezyklen aufgerufen werden.
- Im Aktionsteil einer Synchronaktion in der ein Technologiezyklus aufgerufen wird, darf, außer dem Aufruf weiterer Technologiezyklen, keine andere Aktion programmiert werden.

Siehe auch

Bearbeitungsmode (ICYCON, ICYCOF) (Seite 114)

3.8.2 Bearbeitungsmode (ICYCON, ICYCOF)

Funktion

Über die Befehle `ICYCOF` und `ICYCON` kann die Bearbeitungsmode der Aktionen innerhalb von Technologiezyklen gesteuert werden.

Standardmäßig ist Bearbeitungsmode `ICYCON` aktiv.

Bearbeitungsmode: ICYCON

Im Bearbeitungsmode `ICYCON` wird ein Technologiezyklus satzweise abgearbeitet. Dabei wird die Ausführung aller in einem Satz programmierten Aktionen im gleichen Interpolatortakt angestoßen. Sobald alle angestoßen Aktionen beendet sind, wird im darauf folgenden Interpolatortakt der nächste Satz bearbeitet.

Man unterscheidet ein- und mehrtaktigen Aktionen. Beispiele sind:

- Eintaktige Aktionen: Hilfsfunktionsausgabe, Wertzuweisungen
- Mehrtaktige Aktionen: Verfahrbewegungen von Achsen und Spindeln

Jeder Satz eines Technologiezyklus benötigt mindestens einen Interpolatortakt.

Bearbeitungsmode: ICYCOF

Im Bearbeitungsmode `ICYCOF` werden alle Aktionen aller Sätze eines Technologiezyklus gleichzeitig parallel angestoßen.

Unterprogramm als Teileprogramm

Wird ein Unterprogramm als Teileprogramm abgearbeitet, haben die Befehle `ICYCOF` und `ICYCON` keine Auswirkung.

Syntax

Im Aktionsteil einer Synchronaktion

```
ID=1 <Bedingungsteil> DO [ICYCOF] <Technologiezyklus_1> [ICYCOF |
ICYCON] <Technologiezyklus_2> ...
```

Als Eigenschaft eines Unterprogramms

```
PROC <Name> [ICYCOF | ICYCON]
```

Innerhalb eines Unterprogramms

```
PROC <Name>
  N10 ...
  N20 [ICYCOF | ICYCON]
  N90 ...
  N100 [ICYCOF | ICYCON]
  N110 ...
RET
```

Beispiel

Programmcode	Wirksamer Bearbeitungsmode	Interpolatortakt
PROC TECHNOCYC	ICYCON	
$R1=1	ICYCON	1
POS[X]=100	ICYCON	2 ... 25
ICYCOF	ICYCOF	26
$R1=2	ICYCOF	26
$R2=$R1+1	ICYCOF	26
POS[X]=110	ICYCOF	26
$R3=3	ICYCOF	26
RET	ICYCOF	26

3.8.3 Definitionen (DEF, DEFINE)

Wird ein Unterprogramm als Technologiezyklus verwendet, das Befehle zur Variablen- (DEF) und/oder Makrodefinition (DEFINE) enthält, haben diese bei der Abarbeitung des Technologiezyklus **keine Auswirkung**.

Obwohl Variablen- und Makrodefinitionen innerhalb eines Technologiezyklus keine Auswirkung haben, müssen sie dennoch syntaktisch korrekt sein. Im Fehlerfall wird die Abarbeitung des Technologiezyklus abgebrochen und ein Alarm angezeigt.

Da die Variablen und Makros im Technologiezyklus nicht zur Verfügung stehen, müssen u. U. im Programmcode besondere Maßnahmen getroffen werden. Siehe Kapitel "Kontext-Variable ($P_TECCYCLE) (Seite 116)".

3.8.4 Parameterübergabe

Alle Arten der Paramterübergabe und Parameterdefinition die im Rahmen von Unterprogrammen möglich sind, können auch dann verwendet werden, wenn das Unterprogramm als Technologiezyklus verwendet wird:

- Call-by-Value
- Call-by-Reference
- Default-Parameter

Literatur

Eine Ausführliche Beschreibung der Paramterübergabe und Parameterdefinition bei Unterprogrammen findet sich in:

Programmieranleitung Arbeitsvorbereitung, Kapitel "Fexible NC-Programmierung" > "Unterprogrammtechnik" > "Definition eines Unterprogramms" bzw. "Aufruf eines Unterprogramms"

3.8.5 Kontext-Variable ($P_TECCYCLE)

Funktion

Über die Systemvariable $P_TECCYCLE kann innerhalb eines Unterprogramms ermittelt werden, ob das Unterprogramm aktuell als Teileprogramm oder Technologiezyklus abgearbeitet wird:

- $P_TECCYCLE == TRUE: Abarbeitung als Technologiezyklus
- $P_TECCYCLE == FALSE: Abarbeitung als Teileprogramm

Wird ein Unterprogramm sowohl als Teileprogramm als auch als Technologiezyklus verwendet, kann somit festgelegt werden, welche Programmabschnitte als Teileprogramm und welche als Technologiezyklus abgearbeitet werden.

Anwendung

In Technologiezyklen haben Variablen- (DEF) und Makrodefinitionen (DEFINE) keine Auswirkung. Wird ein Unterprogramm als Technologiezyklus verwendet, das entsprechende Definitionen enthält, ist im Programmcode eine Fallunterscheidung notwendig, da die Variablen und Makros dann nicht zur Verfügung stehen.

Beispiel:
Verfahrparameter über Anwendervariablen im Teileprogramm und R-Parameter im Technologiezyklus

```
Programmcode                                    Kommentar: Verwendung im
PROC UP_1
  DEF REAL POS_X=100.0                              Teileprogramm
  DEF REAL F_X=250.0                                Teileprogramm

  IF $P_TECCYCLE==TRUE
    $R1=100.0                                       Technologiezyklus
    $R2=250.0                                       Technologiezyklus
  ENDIF



  IF $P_TECCYCLE==TRUE
    N100 POS[X]=$R1 FA[X]=$R2                       Technologiezyklus
  ELSE
    N200 POS[X]=POS_X FA[X]=F_X                     Teileprogramm
  ENDIF
RET
```

Siehe auch

Definitionen (DEF, DEFINE) (Seite 115)

3.9 Geschützte Synchronaktionen

Jede Synchronaktion ist über ihre Identifikationsnummer `ID` eindeutig gekennzeichnet.

Über die folgenden Maschinendaten kann NC-global bzw. kanalspezifisch ein Bereich von Identifikationsnummern definiert werden, mit denen eine Synchronaktion gegen Überschreiben, Löschen (`CANCEL(ID)`) und Sperren (`LOCK(ID)`) geschützt werden kann:

- NC-global:
  MD11500 $MN_PREVENT_SYNACT_LOCK (Geschützte Synchronaktionen)
- Kanalspezifisch:
  MD21240 $MC_PREVENT_SYNACT_LOCK_CHAN (Geschützte Synchronaktionen)

Das Verhalten ist in beiden Fällen gleich.

Geschützte Synchronaktionen können über die NC/PLC-Nahtstelle nicht gesperrt werden bzw. werden als nicht sperrbar angezeigt:

- DB21, ... DBB300 ... 307 (Synchronaktionen sperren)
- DB21, ... DBB308 ... 315 (Synchronaktionen sperrbar)

Anwendung

Die vom Maschinenhersteller definierten Synchronaktionen zur Reaktion auf bestimmte Maschinenzustände sollen nach der Inbetriebnahme nicht mehr beeinflusst werden können.

Hinweis

Es wird empfohlen den Schutz der Synchronaktionen während der Inbetriebnahmephase nicht zu aktivieren, da sonst bei jeder Änderung an der Synchronaktion Power On-Reset notwendig ist

Beispiel

In einem System mit 2 Kanälen sollen die Synchronaktionen folgender Identifikationsnummer-Bereiche geschützt werden:

Kanal 1: 20 ... 30

Kanal 2: 25 ... 35

Maschinendatenprojektierung

NC-globaler Schutzbereich:

- MD11500 $MN_PREVENT_SYNACT_LOCK[0] = 25
- MD11500 $MN_PREVENT_SYNACT_LOCK[1] = 35

Kanalspezifischer Schutzbereich für Kanal 1:

- MD21240 $MC_PREVENT_SYNACT_LOCK_CHAN[0] = 20
- MD21240 $MC_PREVENT_SYNACT_LOCK_CHAN1] = 30

Kanalspezifischer Schutzbereich für Kanal 2:

- MD21241 $MC_PREVENT_SYNACT_LOCK_CHAN[0] = -1
- MD21241 $MC_PREVENT_SYNACT_LOCK_CHAN[1] = -1

Im Kanal 2 wurde kein eigener Schutzbereich definiert, daher wirkt der NC-globale Schutzbereich.

3.10 Koordinierung über Teileprogramm und Synchronaktion (LOCK, UNLOCK, RESET, CANCEL)

Jeder modalen und statischen Synchronaktion muss bei der Definition eine eindeutige Identifikationsnummer zugeordnet werden:

Programmcode
ID=<**Nummer**> Bedingungsteil DO Aktionsteil
IDS=<**Nummer**> Bedingungsteil DO Aktionsteil

Über die folgenden Befehle können unter Angabe der Identifikationsnummer Synchronaktionen aus Teileprogrammen und Synchronaktionen koordiniert werden:

Schlüsselwort	Bedeutung	TP[1)]	SA[2)]
LOCK(<Nummer>):	Synchronaktion sperren Ein aktiver Positioniervorgang wird unterbrochen.	-	x
UNLOCK(<Nummer>):	Unterbrochene Synchronaktion fortsetzen Ein unterbrochener Positioniervorgang wird fortgesetzt.	-	x
RESET(<Nummer>):	Synchronaktion abbrechen Ein aktiver Positioniervorgang wird abgebrochen. Wird ein Technologiezyklus neu gestartet, beginnt seine Bearbeitung mit dem 1. Satz im Technologiezyklus. Je nach Art der Bedingung, werden die Aktionen nach erneutem Erfüllen der Bedingung wieder ausgeführt. Bereits ausgeführte Synchronaktionen mit Bedingung WHEN werden nach RESET nicht mehr bearbeitet.	-	x
CANCEL(<Nummer>):	Synchronaktion löschen Ein aktiver Positioniervorgang wird beendet.	x	-
[1)] Programmmierbar im Teileprogramm [2)] Programmmierbar in einer Synchronaktion / Technologiezyklus			

3.11 Koordinierung über PLC

Bezüglich ihrer Ausführung durch die NC können nicht geschützte Synchronaktionen gesperrt werden. Es können entweder alle Synchronaktionen im Kanal gemeinsam gesperrt werden oder einzeln die Synchronaktionen im Bereich ID/IDS 1 - 64.

Gesamt, kanalspezifisch

Alle Synchronaktionen im Kanal sperren:

DB21, … DBX1.2 = 1 (Synchronaktion aus)

Einzeln, kanalspezifisch

Sperrbare Synchronaktionen

Welche Synchronaktionen sperrbar sind, wird angezeigt über:

DB21, … DBX308.0 - 315.7 == 1 (Synchronaktionen ID/IDS sperrbar)

Das Aktualisieren der Anzeige muss über das nachfolgende Triggersignal aktiv vom PLC-Anwenderprogramm ausgelöst werden:

DB21, … DBX281.1 = **1** (Anforderung: sperrbare Synchronaktionen aktualisieren)

Die NC aktualisiert daraufhin die Anzeige der sperrbaren Synchronaktionen und quittiert das Aktualisieren durch Rücksetzen des Triggersignals:

DB21, … DBX281.1 = **0** (Quittung: sperrbare Synchronaktionen aktualisiert)

Synchronaktionen sperren

Für jede Synchronaktion, die im Kanal gesperrt werden soll, ist vom PLC-Anwenderprogramm das entsprechende Sperrbit zu setzen:

DB21, … DBX300.0 - 307.7 = 1 (Synchronaktion ID/IDS 1 - 64 sperren)

Als Anforderung an den Kanal, die aktuellen Sperrbits zu übernehmen, muss vom PLC-Anwenderprogramm das nachfolgende Triggersignal gesetzt werden:

DB21, … DBX280.1 = **1** (Anforderung: zu sperrende Synchronaktionen übernehmen)

Die NC übernimmt daraufhin die zu sperrenden Synchronaktionen in den Kanal und quittiert die Übernahme durch Rücksetzen des Triggersignals:

DB21, … DBX280.1 = **0** (Quittung: zu sperrende Synchronaktionen übernommen)

Siehe auch

Geschützte Synchronaktionen (Seite 117)

3.12 Projektierung

Anzahl Synchronaktionselemente

Die Anzahl an Synchronaktionselemente die pro Kanal zur Verfügung gestellt werden sollen, wird eingestellt über das Maschinendatum:

MD28250 $MC_MM_NUM_SYNC_ELEMENTS (Anzahl Elemente für Ausdrücke in Synchronaktionen)

Pro Synchronaktion werden mindesten 4 Synchronaktionselementen benötigt. Weiter Synchronaktionselementen werden benötigt bei:

Anweisung	Anzahl benötigter Elemente
Operator in der Bedingung	1
Aktion	>= 1
Zuweisung	2
weiteren Operanden in komplexen Ausdrücken	1

Somit hängt die Anzahl programmierbarer Synchronaktionen von der Anzahl der zur Verfügung stehenden Synchronaktionselemente und der Komplexität der Synchronaktionen ab.

Speicherauslastung

Mit der Statusanzeige für Synchronaktionen (siehe Kapitel "Diagnose (nur HMI Advanced) (Seite 125)") lässt sich die Speicherauslastung des Synchronaktionsspeichers verfolgen.

Die Anzahl freier Synchronaktionselement kann auch über die Systemvariable $AC_SYNA_MEM gelesen werden.

Werden im laufenden Betrieb mehr Synchronaktionselemente benötigt als zur Verfügung stehen, wird der Alarm "14751 zu wenig Ressourcen für Bewegungssynchronaktionen" angezeigt.

Anzahl FCTDEF-Elemente

Die Anzahl an `FCTDEF`-Elementen pro Kanal wird eingestellt über das Maschinendatum:

MD28252 $MC_MM_NUM_FCTDEF_ELEMENTS (Anzahl der FCTDEF-Elemente)

Synchronaktionen und Interpolatortakt

Bei einer großen Anzahl gleichzeitig aktiver Synchronaktionen muss unter Umständen der Interpolatortakt erhöht werden:

MD10070 $MN_IPO_SYSCLOCK_TIME_RATIO

Tabelle 3-1 Zeitbedarf einzelnen Operationen

Synchronaktionsbefehle	Zeitbedarf [1]	
	gesamt	fett markierter Anteil
Grundlast für eine Synchronaktion, wenn Bedingung nicht erfüllt ist: **WHENEVER FALSE** DO $AC_MARKER[0]=0	10 µs	**10 µs**
Variable lesen: WHENEVER **$AA_IM[Y]>10** DO $AC_MARKER[0]=1	11 µs	**1 µs**
Variable schreiben: DO **$R2=1**	11-12 µs	**1-2 µs**
Settingdatum lesen/schreiben: DO**$$SN_SW_CAM_MINUS_POS_TAB_1[0]=20**	24 µs	**14 µs**

Synchronaktionsbefehle	Zeitbedarf [1]	
	gesamt	fett markierter Anteil
Grundrechenarten, z. B. Multiplikation: DO $R2=**$R2*2**	22 µs	**12 µs**
Trigonometrische Funktionen (z. B. cos): DO $R2=**COS($R2)**	23 µs	**13 µs**
Positionierachse starten: WHEN TRUE DO **POS[z]=10**	83 µs	**73 µs**
1) gemessen mit SINUMERIK 840D mit NCU 7x0.3 PN		

3.13 Steuerungsverhalten in bestimmten Betriebszuständen

3.13.1 Power On

Während des Hochlauf der NC (Power On) sind keine Synchronaktionen aktiv.

Synchronaktionen, die sofort nach dem Hochlauf der NC (Power On) aktiv sein sollen, müssen als statische Synchronaktionen innerhalb eines ASUP ereignisgesteuert oder über PLC-Anwenderprogramm aktiviert werden.

Literatur

Ausführliche Informationen zum Aktivieren von Synchronaktionen nach dem Hochlauf der NC (Power On) finden sich:

PLC-Anwenderprogramm

Funktionshandbuch Grundfunktionen; PLC-Grundprogramm für SINUERMIK 840D sl
Kapitel "Struktur und Funktionen des Grundprogramms" > "Funktionen des Grundprogramms mit Aufruf vom Anwenderprogramm"

Ereignisgesteuert

Funktionshandbuch Grundfunktionen; BAG, Kanal, Programmbetrieb (K1)
Kapitel "Programmbetrieb" > "Ereignisgesteuerte Programmaufrufe"

3.13.2 NC-Reset

Zustand nach NC-Reset:

Von:	Satzweise und modale Synchronaktion (ID)	Statische Synchronaktion (IDS)
Synchronaktion	abgebrochen bzw. inaktiv	aktiv
Verfahrbewegung	Die Verfahrbewegungen werden abgebrochen	

Von:	Satzweise und modale Synchronaktion (ID)	Statische Synchronaktion (IDS)
Drehzahlgeregelte Spindel	MD35040 $MA_SPIND_ACTIVE_AFTER_RESET = <Wert> TRUE ⇒ Die Spindel bleibt aktiv FALSE ⇒ Die Spindel wird gestoppt	
Leitwertkopplung	MD20110 $MC_RESET_MODE_MASK, Bit13 = <Wert> 1 ⇒ Die Kopplung bleibt aktiv 0 ⇒ Die Kopplung wird gelöst	
Messen	abgebrochen	

3.13.3 NC-Stop

Satzweise und modale Synchronaktionen (ID)

Verfahrbewegungen aus satzweisen und modalen Synchronaktionen werden durch NC-Stop angehalten.

Eine satzweise oder modale Synchronaktion bleibt auch während der Kanal im Zustand "Unterbrochen" ist aktiv:

DB21, ... DBX35.6 == 1 (Kanalzustand unterbrochen)

Wird in dieser Zeit die Bedingung erfüllt, werden, außer Verfahrbewegungen, die Aktionen ausgeführt.

Mit NC-Start werden angehaltene Verfahrbewegungen fortgesetzt.

Statische Synchronaktionen (IDS)

Verfahrbewegungen aus statischen Synchronaktionen werden durch NC-Stop **nicht** angehalten.

3.13.4 Betriebsartenwechsel

Zustand nach Betriebsartenwechsel:

Von:	Satzweise und modale Synchronaktion (ID)	Statische Synchronaktion (IDS)
Synchronaktion	abgebrochen bzw. inaktiv [1)]	aktiv
Verfahrbewegung	abgebrochen [2)]	aktiv
Drehzahlgeregelte Spindel	aktiv	aktiv
Leitwertkopplung	MD20110 $MC_RESET_MODE_MASK, Bit13 = <Wert> 1 ⇒ Die Kopplung bleibt aktiv 0 ⇒ Die Kopplung wird gelöst	aktiv
Messen	abgebrochen	aktiv
1) Nach einem erneuten Wechsel in die Betriebsart AUTMATIK werden die Synchronaktionen wieder aktiv. 2) Programmende `M30` wird verzögert, bis die Achse steht.		

3.13.5 Programmende

Zustand nach Programmende:

Von:	Satzweise und modale Synchronaktion (ID)	Statische Synchronaktion (IDS)
Synchronaktion	abgebrochen bzw. inaktiv	aktiv
Verfahrbewegung	abgebrochen 1)	aktiv
Drehzahlgeregelte Spindel	MD35040 $MA_SPIND_ACTIVE_AFTER_RESET = <Wert> TRUE ⇒ Die Spindel bleibt aktiv FALSE ⇒ Die Spindel wird gestoppt	aktiv
Leitwertkopplung	MD20110 $MC_RESET_MODE_MASK, Bit13 = <Wert> 1 ⇒ Die Kopplung bleibt aktiv 0 ⇒ Die Kopplung wird gelöst	aktiv
Messen	abgebrochen	aktiv
1) Programmende `M30` wird verzögert, bis die Achse steht.		

3.13.6 Satzsuchlauf

Satzweise und modale Synchronaktionen (ID)

Synchronaktionen werden während des Satzsuchlaufs aufgesammelt aber nicht aktiv. D. h. die Bedingungen werden nicht ausgewertet, die Aktionen nicht ausgeführt.

Die Synchronaktionen werden erst mit NC-Start aktiv. D. h. die Bedingungen werden ausgewertet, und die Aktionen gegebenenfalls ausgeführt.

Statische Synchronaktionen (IDS)

Statische Synchronaktionen die bereits aktiv sind, wirken auch während des Satzsuchlaufs.

3.13.7 Programmunterbrechung durch ASUP

Satzweise und modale (ID) Synchronaktionen

Aktive modale Synchronaktionen bleiben auch während des ASUP weiter aktiv.

Aus satzweisen und modalen Synchronaktionen gestartete Verfahrbewegungen werden unterbrochen. Wird mit dem Ende des ASUP auf den Unterbrechungspunkt des Teileprogramms positioniert (`REPOS`), werden die unterbrochenen Verfahrbewegungen fortgesetzt.

Statische Synchronaktionen (IDS)

Statische Synchronaktionen bleiben auch während des ASUP weiter aktiv.

Aus statischen Synchronaktionen gestartete Verfahrbewegungen werden durch das ASUP nicht unterbrochen.

Synchronaktionen des ASUP

Wird das ASUP nicht mit `REPOS` fortgesetzt, wirken die modalen und statischen Synchronaktionen aus dem ASUP im Teileprogramm weiter.

3.13.8 REPOS

Im Restsatz gelten die Synchronaktionen wie im Unterbrechungssatz.

Änderungen an den modalen Synchronaktionen im asynchronen Unterprogramm sind im unterbrochenen Programm nicht wirksam.

Die mit `FCTDEF` programmierten Polynomkoeffizienten werden von `ASUP` und `REPOS` nicht beeinflusst.

Im asynchronen Unterprogramm wirken die Koeffizienten aus dem aufrufenden Programm. Im aufrufenden Programm wirken die Koeffizienten aus dem asynchronen Unterprogramm weiter.

Wurden mit Betriebsartenwechsel oder dem Start des Interruptprogramms Positionierbewegungen aus Synchronaktionen unterbrochen, so werden diese mit `REPOS` fortgesetzt.

3.13.9 Verhalten bei Alarmen

- Löst eine Aktion einer Synchronaktion einen Alarm aus, wird diese Aktion abgebrochen. Andere Aktionen der Synchronaktion werden bearbeitet.
- Löst eine modale Synchronaktion einen Alarm aus, wird sie ab dem Alarmzeitpunkt inaktiv.
- Löst ein Technologiezyklus einen Alarm mit Bewegungsstopp aus, wird er abgebrochen und nicht weiter bearbeitet.
- Bei einem Alarm mit Bewegungsstopp, werden alle von Synchronaktionen gestarteten Achs– bzw. Spindelbewegungen gestoppt. Aktionen ohne Verfahrbewegungen werden weiter ausgefuehrt.
- Bei einem Alarm mit Interpreterstopp, wirkt bei Synchronaktionen der Interpreterstopp erst nach dem vollständigen Abarbeiten der vordekodierten Sätze.

3.14 Diagnose (nur HMI Advanced)

Funktionalität der Diagnose

Für die Diagnose von Synchronaktionen stehen die folgenden speziellen Testmittel zur Verfügung:

- Statusanzeige der Synchronaktionen im Bedienbereich Maschine
- Systemvariablen anzeigen im Bedienbereich Parameter
  Es können aktuelle Werte aller Synchronaktions-Variablen angezeigt werden.
  (Hauptlaufvariablen anzeigen)
- Systemvariable protokollieren im Bedienbereich Parameter
  Es können Variablen-Verläufe im Interpolatortakt-Raster aufgezeichnet werden.
  (Hauptlaufvariablen protokollieren)

Diese Funktionalität ist in der Bedienoberfläche wie folgt strukturiert:

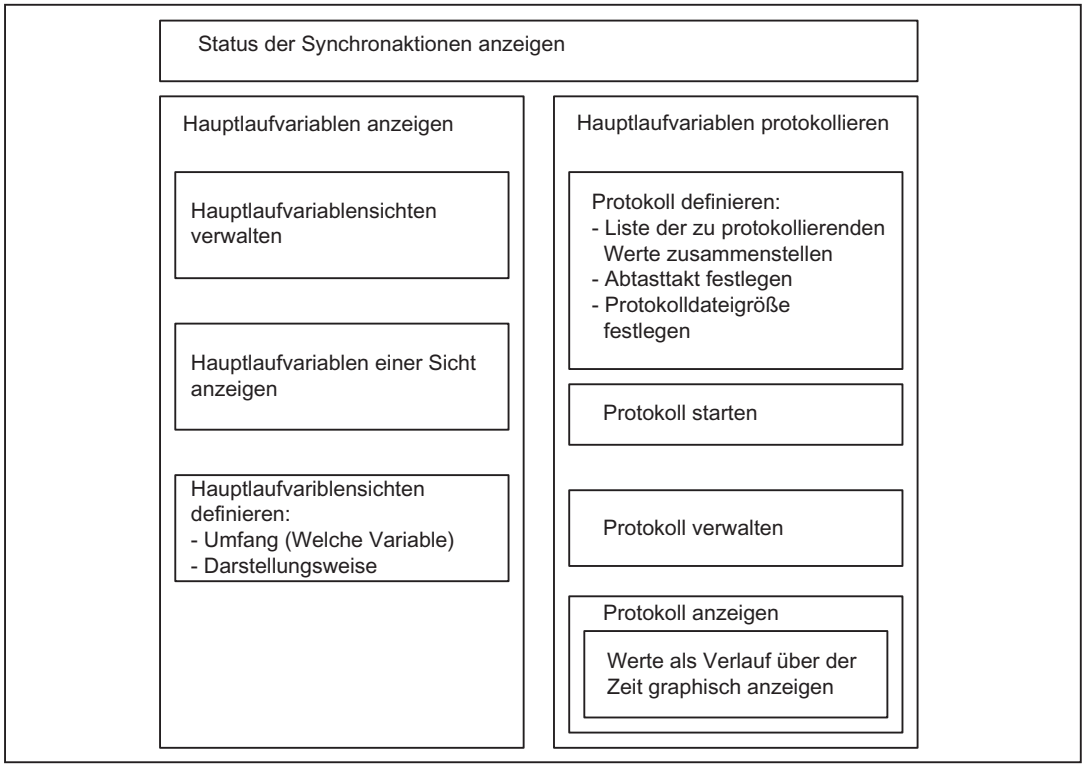


Bild 3-8 Funktionalität der Testmittel für Synchronaktionen

Die Beschreibung der Bedienung dieser Funktionen finden Sie in:

Literatur:

/BAD/ Bedienungshandbuch HMI Advanced.

3.14.1 Status der Synchronaktionen anzeigen

Über das Statusbild der Synchronaktionen werden folgende Informationen angezeigt:

- Übersicht der programmierten Synchronaktionen
- Gültigkeit und Identifikationsnummer (nur bei modalen Synchronaktionen)
  Siehe dazu Kapitel "Gültigkeit, Identifikationsnummern (ID, IDS) (Seite 14)"
- Status der Synchronaktion

Status

Status	Bedeutung
keine Angabe	Die Bedingung wird im Interpolatortakt überprüft.
gesperrt	Die Synchronaktion ist gesperrt. Siehe Kapitel: • Koordinierung über Teileprogramm und Synchronaktion (LOCK, UNLOCK, RESET, CANCEL) (Seite 118) • Koordinierung über PLC (Seite 118)
aktiv	Der Aktionsteil der Synchronaktion wird ausgeführt. Besteht die Aktion aus einem Technologiezyklus, wird die aktuelle Satznummer in diesem angezeigt.

Literatur

Bedienhandbuch HMI-Advanced, Kapitel "Bedienbereich Maschine" > "Allgemeine Funktionen und Anzeigen" > "Status der Synchronaktionen"

3.14.2 Hauptlaufvariablen anzeigen

Bedeutung

Für den Test von Synchronaktionen ist es möglich, die Systemvariablen zu verfolgen. Die zulässigen Variablen werden in einer Vorschlagsliste zur Auswahl angeboten.

Die vollständige Liste der einzelnen Systemvariablen mit Kennzeichnung des Schreibzugriffs W und des Lesezugriffs R für Synchronaktionen finden Sie in:

Literatur:

/PGA1/ Listenhandbuch Systemvariablen

Sichten

In Sichten legt der Anwender fest, welche Werte für eine bestimmte Bearbeitungssituation wichtig sind und wie (nach Zeilen und Spalten, mit welchem Text) diese Werte angezeigt werden sollen. Es können mehrere Sichten zusammengestellt und in benannten Dateien abgespeichert werden.

Sichten verwalten

Eine definierte Sicht kann unter einem anwenderdefinierten Namen abgespeichert und wieder aufgerufen werden. Die in einer Sicht enthaltenen Variablen können verändert werden. (Sicht bearbeiten).

Hauptlaufvariable einer Sicht anzeigen

Die Anzeige der zu einer Sicht gehörenden Werte erfolgt durch Aufruf der entsprechenden Anwendersicht.

3.14.3 Hauptlaufvariablen protokollieren

Ausgangssituation

Die genaue Verfolgung der Abläufe in Synchronaktionen erfordert die Beobachtung der Zustände im Interpolatortakt.

Methode

Die in einer Protokolldefinition festgelegten Werte werden im angegebenen Takt in eine Protokolldatei definierter Größe eingeschrieben. Für die Anzeige der Inhalte der Protokolldateien werden Funktionen angeboten.

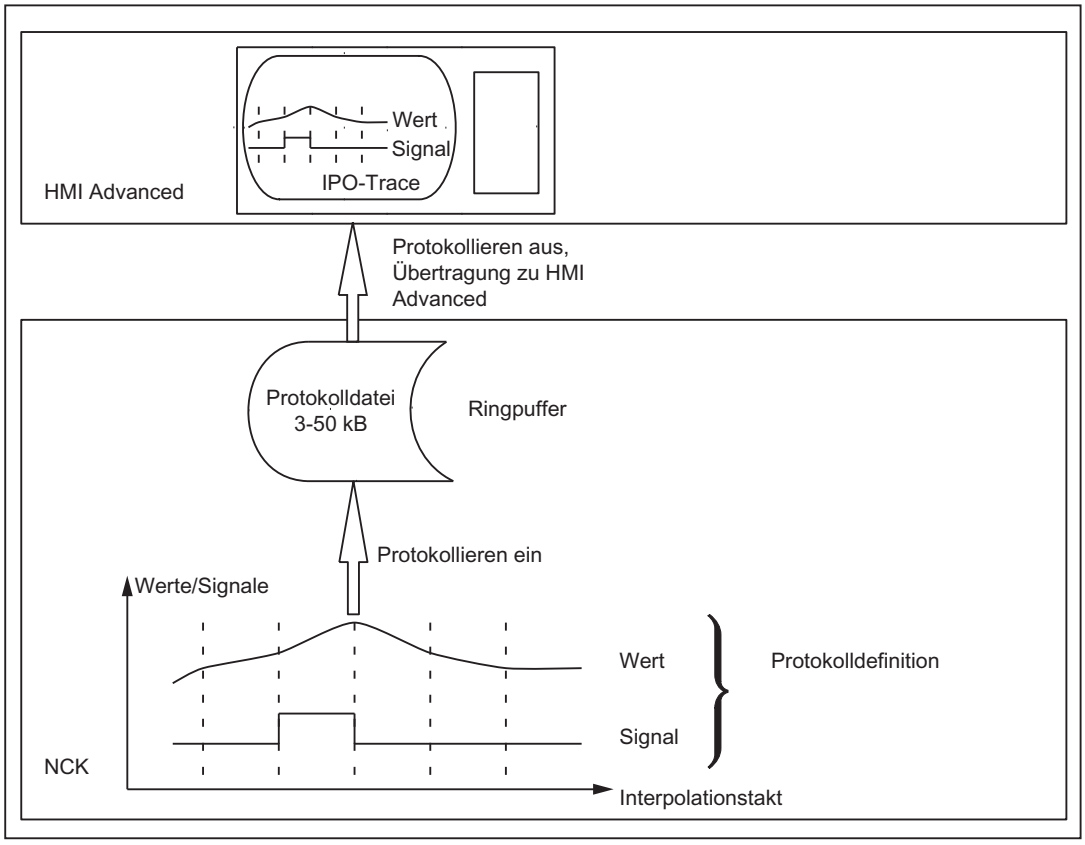


Bild 3-9 Schematischer Ablauf Hauptlaufvariablen protokollieren

Bedienung

Die Hinweise zur Bedienung der Protokollierfunktion finden Sie in:

Literatur:

/BAD/ Bedienungshandbuch HMI Advanced

Protokolldefinition

In der Protokolldefinition können bis zu 6 Variablen angegeben werden, deren Werte im angegebenen Takt in die Protokolldatei eingeschrieben werden sollen. Für die Auswahl der zu protokollierenden Variablen wird eine Liste angeboten. Der Takt ist in Vielfachen des Interpolatortaktes wählbar. Die Dateigröße in kByte kann gewählt werden. Eine Protokolldefinition muss initialisiert werden, damit sie auf NCK aktiviert werden kann zum Erfassen der gewünschten Werte.

Protokolldateigröße

Als Größe der Protokollierdatei können Werte von minimal 3 kByte bis maximal 50 kByte gewählt werden.

Speichermethode

Beim Überschreiten der effektiven Protokolldateigröße werden die ältesten Einträge überschrieben, so dass sich ein Ringpuffer ergibt.

Protokollierung starten

Die Protokollierung gemäß einer der initialisierten Protokolldefinitionen wird gestartet durch:

- Bedienung
- Setzen der Systemvariablen $A_PROTO=1 aus dem Teileprogramm

Der Startzeitpunkt muss so gewählt werden, dass die zu protokollierenden Variablen erst nach der Aktivierung von Abläufen auf der Maschine verändert werden. Der Start bezieht sich auf die zuletzt initialisierte Protokolldefinition.

Protokollierung stoppen

Die Funktion schließt die Protokolldatenerfassung im NCK ab. Die Datei mit den erfassten Werten wird auf HMI zur Abspeicherung und Auswertung (Protokoll graphisch) bereitgestellt. Die Protokollierung kann gestoppt werden durch:

- Bedienung
- Setzen der Systemvariablen $A_PROTO=0 aus dem Teileprogramm

Funktion Protokoll graphisch

Die bis zu 6 Messwerte eines Protokolls werden graphisch über der Abtastzeit dargestellt. Die Variablennamen werden in der Reihenfolge von oben nach unten entsprechend den Werteverläufen genannt. Die Platzverteilung auf dem Bildschirm erfolgt automatisch. Auf einen ausgewählten Teilbereich der Graphik kann eine Spreizung angewendet werden.

Hinweis

Die graphisch dargestellten Protokolle stehen auf HMI Advanced auch als Textdatei zur Verfügung. Mithilfe eines Editors können die exakten Werte eines Abtastzeitpunktes (Werte mit gleichem Zählindex) numerisch gelesen werden.

Verwalten von Protokollen

Es können mehrere Protokolldefinitionen unter anwenderdefinierten Namen gespeichert und für Initialisierung und Start der Aufzeichnung oder für Änderungen und Löschung wieder aufgerufen werden.

Beispiele

4

4.1 Beispiele für Bedingungen in Synchronaktionen

Bedingung	Programmierung
Bahnrestweg ≤ 10 mm (WKS)	`... WHEN $AC_DTEW <= 10 DO ...`
Restweg der X-Achse ≤ 10 mm (WKS)	`... WHEN $AA_DTEW[X] <= 10 DO ...`
Bahnabstand zum Satzanfang ≥ 20 mm (BKS)	`... WHEN $AC_PLTBB >= 20 DO ...`
Istwert der Y-Achse (MKS) > 10 * SIN(R10)	`... WHEN $AA_IM[y] > 10*SIN(R10) DO...`
Eingang 1 wechselt von 0 nach 1	`... EVERY $A_IN[1]==1 DO ...`
Eingang 1 == 1	`... WHENEVER $A_IN[1]==0 DO ...`

4.2 Schreiben und Lesen von SD/MD aus Synchronaktionen

Zustellung und Pendeln beim Schleifen

Settingdaten, deren Werte während der Bearbeitung unverändert bleiben, werden wie im Teileprogramm Namen angesprochen.

Beispiel: Pendeln aus Synchronaktionen

Programmcode

```
N610 ID=1 WHENEVER $AA_IM[Z] > $SA_OSCILL_REVERSE_POS1[Z] DO $AC_MARKER[1]=0
...
; IMMER WENN aktuelle Position der Pendelachse im MKS < Beginn des Umkehrbereichs 2,
; DANN Override der Zustellachse = 0%
N620 ID=2 WHENEVER $AA_IM[Z] < $SA_OSCILL_REVERSE_POS2[Z] - 6 DO
          $AA_OVR[X]=0 $AC_MARKER[0]=0
...
; IMMER WENN die aktuelle Position der Pendelachse im MKS == Umkehrposition 1,
; DANN Override der Pendelachse = 0%, Override der Zustellachse = 100%
; Damit wird die vorhergehende Synchronaktion aufgehoben!
N630 ID=3 WHENEVER $AA_IM[Z] == $SA_OSCILL_REVERSE_POS1[Z] DO
          $AA_OVR[Z]=0 $AA_OVR[X]=100
...
; IMMER WENN Restweg der Teilzustellung == 0,
; DANN Override der Pendelachse = 100%
; Damit wird die vorhergehende Synchronaktion aufgehoben!
N640 ID=4 WHENEVER $AA_DTEPW[X]==0 DO $AA_OVR[Z]=100 $AC_MARKER[0]=1 $AC_MARKER[1]=1
N650 ID=5 WHENEVER $AC_MARKER[0]==1 DO $AA_OVR[X]=0
N660 ID=6 WHENEVER $AC_MARKER[1]==1 DO $AA_OVR[X]=0
```

Programmcode

```
...
; WENN aktuelle Position der Pendelachse im WKS == Umkehrposition 1,
; DANN Override der Pendelachse = 100%, Override der Zustellachse = 0%
; Damit wird die zweite Synchronaktion einmalig aufgehoben!
N670 ID=7 WHEN $AA_IM[Z] == $SA_OSCILL_REVERSE_POS1[Z] DO $AA_OVR[Z]=100 $AA_OVR[X]=0
...
; Settingdaten, deren Wert sich während der Bearbeitung ändert (z. B. per
; Bedienung oder Synchronaktion) müssen mit $$S... programmiert werden:
; Beispiel: Pendeln aus Synchronaktionen mit Änderung der Pendelposition von
; der Bedienoberfläche
N610 ID=1 WHENEVER $AA_IM[Z] > $$SA_OSCILL_REVERSE_POS1[Z] DO $AC_MARKER[1]=0
...
; IMMER WENN aktuelle Position der Pendelachse im MKS < Beginn des Umkehrbereichs 2,
; DANN Override der Zustellachse = 0%
N620 ID=2 WHENEVER $AA_IM[Z] < $$SA_OSCILL_REVERSE_POS2[Z]-6 DO
          $AA_OVR[X]=0 $AC_MARKER[0]=0
...
; IMMER WENN aktuelle Position der Pendelachse im MKS == der Umkehrposition 1,
; DANN Override der Pendelachse = 0%, Override der Zustellachse = 100%
; Damit wird die vorhergehende Synchronaktion aufgehoben!
N630 ID=3 WHENEVER $AA_IM[Z]==$$SA_OSCILL_REVERSE_POS1[Z] DO
          $AA_OVR[Z]=0 $AA_OVR[X]=100
...
; IMMER WENN Restweg der Teilzustellung == 0,
; DANN Override der Pendelachse = 100%
; Damit wird die vorhergehende Synchronaktion aufgehoben!
N640 ID=4 WHENEVER $AA_DTEPW[X]==0 DO $AA_OVR[Z]=100 $AC_MARKER[0]=1 $AC_MARKER[1]=1
N650 ID=5 WHENEVER $AC_MARKER[0]==1 DO $AA_OVR[X]=0
N660 ID=6 WHENEVER $AC_MARKER[1]==1 DO $AA_OVR[X]=0
...
; WENN aktuelle Position der Pendelachse im WKS == Umkehrposition 1,
; DANN Override der Pendelachse = 100%, Override der Zustellachse = 0%
; Damit wird die zweite Synchronaktion einmalig aufgehoben!
N670 ID=7 WHEN $AA_IM[Z]==$$SA_OSCILL_REVERSE_POS1[Z]
DO $AA_OVR[Z]=100 $AA_OVR[X]=0
```

4.3 Beispiele zur AC-Regelung

Allgemeines Vorgehen

Die folgenden Beispiele benutzen die Polynomauswertefunktion `SYNFCT()`.

1. Darstellung des Zusammenhangs zwischen Eingangswert und Ausgangswert (jeweils Hauptlaufvariablen)
2. Definition dieses Zusammenhanges als Polynom mit Begrenzungen
3. bei Positionsoffset: Setzen der MD und SD
   - MD36750 $MA_AA_OFF_MODE (Wirkung der Wertzuweisung für axiale Überlagerung bei Synchronktionen)
   - SD43350 $SA_AA_OFF_LIMIT (optional) (Obergrenze des Korrekturwertes $AA_OFF bei Abstandsregelung)
4. Aktivierung der Regelung in einer Synchronaktion

4.3.1 Abstandsregelung mit variabler Obergrenze

Beispiel für Polynom mit dynamischer Obergrenze

Für eine Abstandsregelung wird die Obergrenze des Ausgangs ($AA_OFF, Überlagerungswert in Achse V) in Abhängigkeit vom Spindeloverride (Analogeingang 1) verändert. Die obere Begrenzung für das Polynom 1 wird dynamisch in Abhängigkeit von Analogeingang 2 verändert.

Es wird das Polynom 1 direkt über die Systemvariablen definiert:

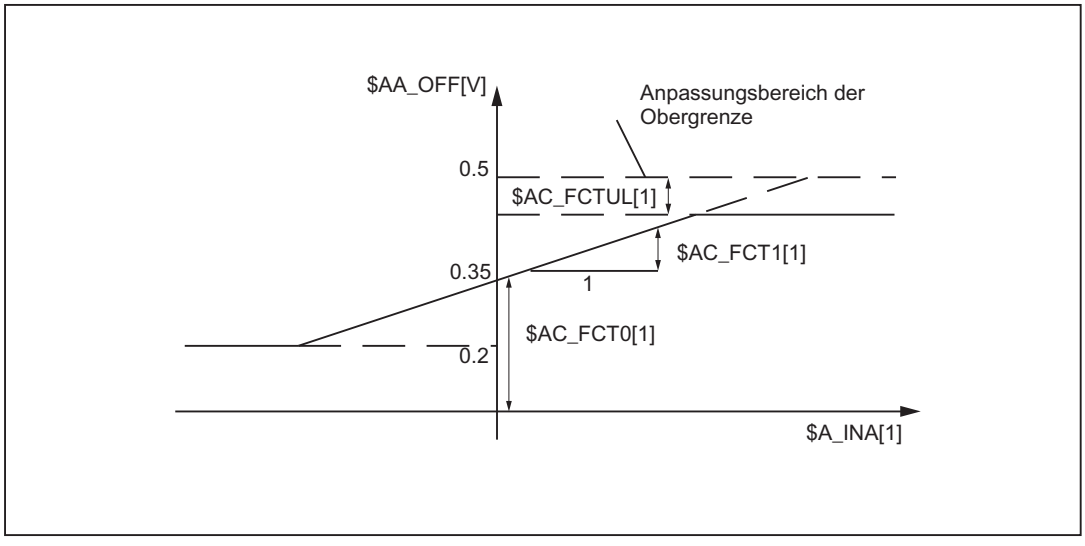


Bild 4-1 Abstandsregelung mit variabler Obergrenze

```
$AC_FCTLL[1]=0.2                              ;  Untere Begrenzung
$AC_FCTUL[1]=0.5                              ;  Anf. Wert obere Begrenzung
$AC_FCT0[1]=0.35                              ;  Nulldurchgang a0
$AC_FCT1[1]=1.5 EX-5                          ;  Steigung a1
STOPRE                                        ;  siehe folgender Hinweis
...
STOPRE                                        ;  siehe folgender Hinweis
ID=1 DO $AC_FCTUL[1]=$A_INA[2]*0.1+0.35       ;  obere Begrenzung dynamisch anpassen
                                                 über Analogeingang 2,
                                              ;  keine Bedingung
                                              ;
ID=2 DO SYNFCT(1, $AA_OFF[V], $A_INA[1])      ;  Abstandsregelung durch Überlagerung
                                                 keine Bedingung
                                              ;
...
```

Hinweis

Bei Verwendung von Systemvariablen im Teileprogramm muss durch Programmierung von `STOPRE` für satzsynchrones Schreiben gesorgt werden. Gleichwertig zur obigen Notation zur Polynomdefinition ist:

```
FCTDEF(1,0.2, 0.5, 0.35, 1.5EX-5).
```

4.3.2 Regelung des Vorschubs

Beispiel für AC-Regelung mit einer analogen Eingangsspannung

Es soll eine Prozessgröße (gemessen über $A_INA[1]) durch Korrektur des Bahn- (oder axialen) Vorschubs additiv beeinflusst auf 2V geregelt werden. Die Vorschubkorrektur soll in den Grenzen ±100 [mm/min] erfolgen.

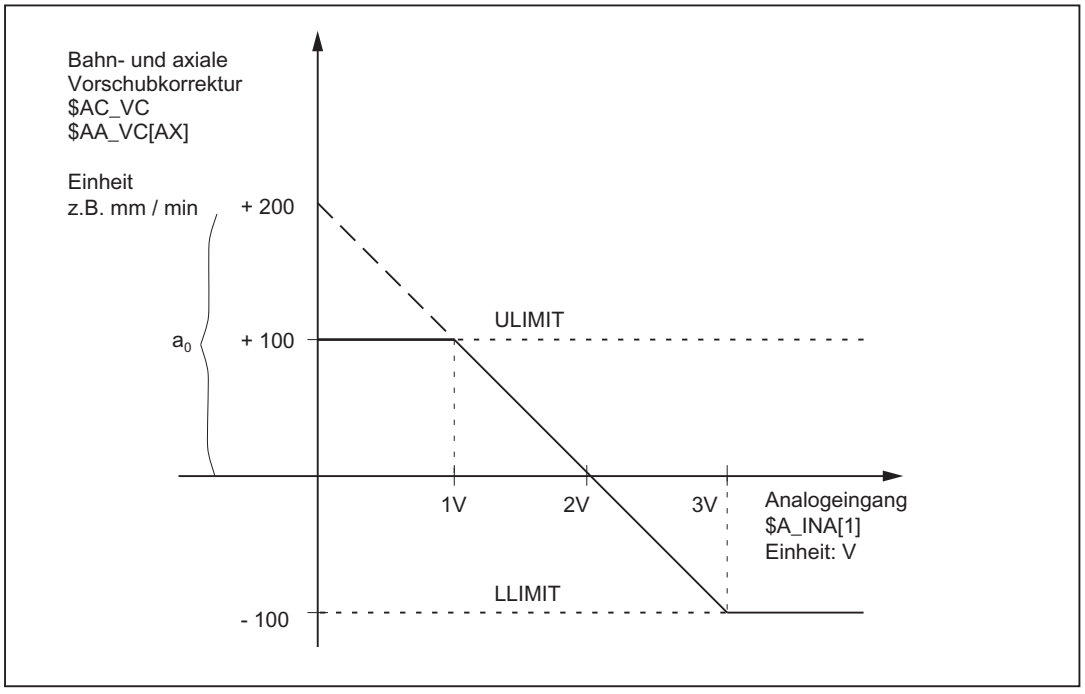


Bild 4-2 Diagramm für AC–Regelung

Bestimmung der Koeffizienten:

$y = f(x) = a_0 + a_1x + a_2x^2 + a_3x^3$

a1 = - 100mm / (1min * 1V)

a1 = - 100% Regelkonstante, Steigung

a0 = - (-100) * 2 = 200

a2 = 0 (kein quadratisches Glied)

a3 = 0 (kein kubisches Glied)

upper limit = 100

lower limit = - 100

```
FCTDEF(          Polynom-Nr,
                 LLIMIT,
                 ULIMIT,
                 a0,                 ;  y für x = 0
                 a1,                 ;  Steigung
```

```
                    a2,                 ;  quadratisches Glied
                    a3 )                ;  kubisches Glied
```

Mit den oben bestimmten Werten lautet die Polynomdefinition:

`FCTDEF`(1, -100, 100, 200, -100, 0, 0)

Mit folgenden Synchronaktionen kann die AC-Regelung eingeschaltet werden:

für den Achsvorschub:

```
ID = 1 DO SYNFCT (1, $AA_VC[X], $A_INA[1])
```

oder für den Bahnvorschub:

```
ID = 2 DO SYNFCT(1, $AC_VC, $A_INA[1])
```

4.3.3 Geschwindigkeit in Abhängigkeit vom normierten Bahnweg regeln

Multiplikative Anpassung

Als Eingangsgröße wird der normierte Bahnweg benutzt: $AC_PATHN.

0: am Satzanfang

1: am Satzende

Die Änderungsgröße $AC_OVR soll in Abhängigkeit von $AC_PATHN nach einem Polynom 3. Ordnung geregelt werden. Der Override soll während der Bewegung von 100 auf 1% reduziert werden.

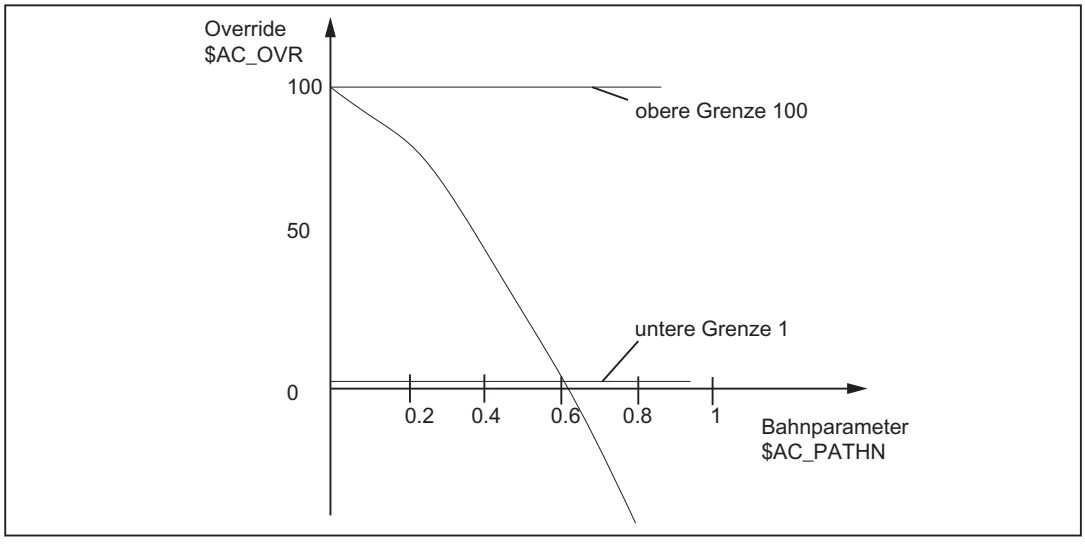


Bild 4-3 Geschwindigkeit kontinuierlich regeln

Polynom 2:

Untere Grenze: 1

Obere Grenze: 100

a_0: 100

a_1: -100

a_2: -100

a_3: nicht benutzt

Mit diesen Werten lautet die Polynomdefinition:

```
FCTDEF(2, 1, 100, 100, -100, -100)
; Aktivierung des Veränderlichen Override abhängig vom Bahnweg:
ID= 1 DO SYNFCT (2, $AC_OVR, $AC_PATHN)
G01 X100 Y100 F1000
```

4.4 Überwachung eines Sicherheitsabstandes zwischen zwei Achsen

Aufgabe

Die Achsen X1 und X2 bedienen zwei unabhängig gesteuerte Transporteinrichtung zum Be- und Entladen von Werkstücken.

Um Kollisionen zu vermeiden, muss zwischen beiden ein Sicherheitsabstand eingehalten werden.

Wird der Sicherheitsabstand unterschritten, so wird die Achse X2 abgebremst. Die Verriegelung gilt so lange, bis Achse X1 den Sicherheitsbereich wieder verlassen hat.

Bewegt sich Achse X1 weiter auf Achse X2 zu und unterschreitet eine engere Sicherheitsschranke, so fährt sie in eine sichere Position.

NC-Sprache	**Kommentar**
ID=1 WHENEVER $AA_IM[X2] - $AA_IM[X1] < 30 DO $AA_OVR[X2]=0	; Sicherheitsschranke
ID=2 EVERY $AA_IM[X2] - $AA_IM[X1] < 15 DO POS[X1]=0	; Sichere Position

4.5 Ausführungszeiten in R-Parameter ablegen

Aufgabe

Lege ab R-Parameter 10 die Ausführungszeit für die Teileprogrammsätze ab.

Programm	**Kommentar**
	; **Ohne** symbolische Programmierung sieht
	; das Beispiel so aus:
IDS=1 EVERY $AC_TIMEC==0 DO $AC_MAR-	; bei Satzwechsel R-Parameter-
KER[0] = $AC_MARKER[0] + 1	; zeiger weiterstellen

Programm	**Kommentar**
IDS=2 DO $R[10+$AC_MARKER[0]] = $AC_TIME	; Schreibe jeweils die aktuelle Zeit ; vom Satzanfang in R-Parameter
	; **Mit** symbolischer Programmierung sieht ; das Beispiel so aus:
DEFINE INDEX AS $AC_MARKER[0]	; Vereinbarungen für symbolische ; Programmierung
IDS=1 EVERY $AC_TIMEC==0 DO INDEX = INDEX + 1	; bei Satzwechsel R-Parameter- ; zeiger weiterstellen
IDS=2 DO $R[10+INDEX] = $AC_TIME	; Schreibe jeweils die aktuelle Zeit ; vom Satzanfang in R-Parameter

4.6 "Einmitten" mit kontinuierlichem Messen

Einführung

Es werden nacheinander die Zahnradlücken vermessen. Aus der Summe der Lücken und der Zähnezahl wird das Lückenmaß ermittelt. Die gesuchte Mittenposition für die Weiterbearbeitung ist die Position des ersten Messpunktes plus 1/2 der durchschnittlichen Lückengröße. Beim Messen wird die Drehzahl so gewählt, dass pro Interpolationstakt ein Messwert sicher erfasst werden kann.

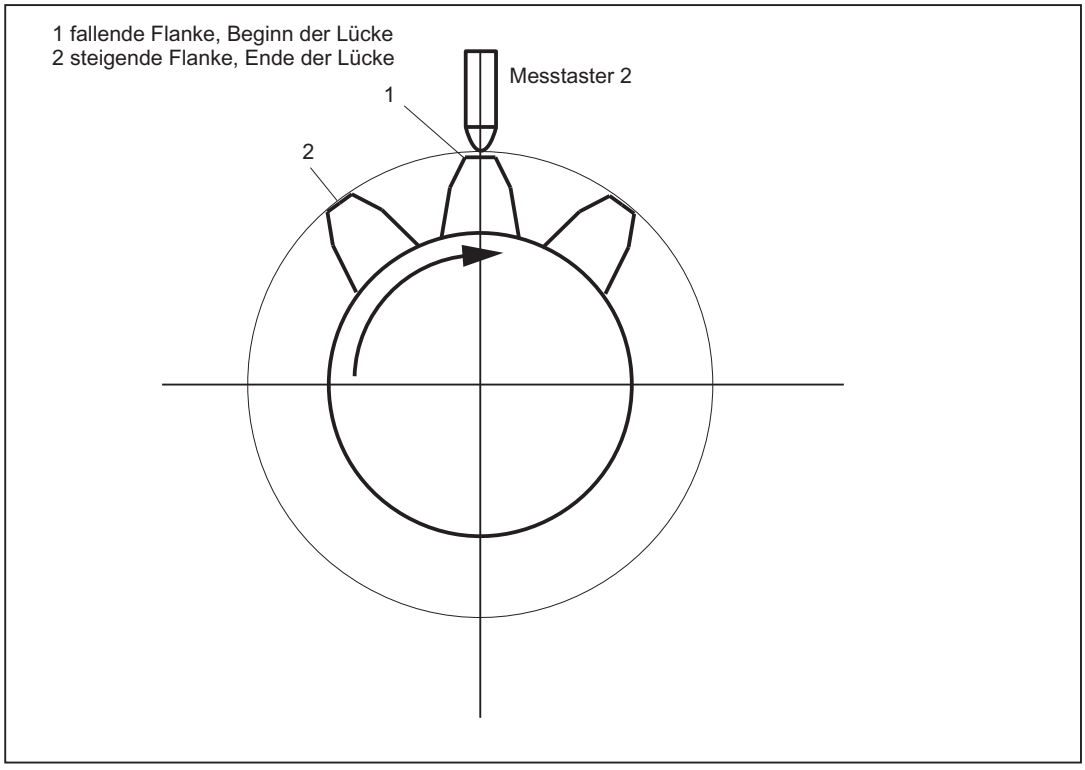


Bild 4-4 Schemabild zum Messen der Zahnradlücken

%_N_MEAC_MITTEN_MPF

;Messen mit der Rundachse B (BACH) mit Anzeige der Differenz
;zwischen den Messwerten

```
;*** Lokale Anwender- Variablen definieren ***
N1 DEF INT ZAEHNEZAHL                ;  Eingabe Anzahl Zahnradzähne
N5 DEF REAL HYS_POS_FLANKE           ;  Hysterrese positive Flanke Taster
N6 DEF REAL HYS_NEG_FLANKE           ;  Hysterrese negative Flanke Taster
;*** Kurznamen für Synchronaktionsmerker definieren ***
define M_ZAEHNE as $AC_MARKER[1]     ;  ID Merker zum Rechnen: neg/pos Flanke je Zahn
define Z_MW as $AC_MARKER[2]         ;  ID Zähler MW FIFO auslesen
define Z_RW as $AC_MARKER[3]         ;  ID Zähler MW Rechnen Zahnlücken
;*** Eingabewerte für ZAHNRADMESSEN ***
N50 ZAEHNEZAHL=26                    ;  Eingabe Anzahl zu messende Zahnradzähne
N70 HYS_POS_FLANKE = 0.160           ;  Hysterrese positive Flanke Taster
N80 HYS_NEG_FLANKE = 0.140           ;  Hysterrese negative Flanke Taster
```

```
Anfang:                              ;  *** Variablen zuweisen ***
R1=0                                 ;  ID2 Rechenergebnis Lückenmaß
R2=0                                 ;  ID2 Rechenergebnis Addition aller Lücken
R3=0                                 ;  Inhalt des zuerst eingelesenen Elements
R4=0                                 ;  R4 Entspricht einem Zahnabstand
R5=0                                 ;  Lückenposition errechnet, Endergebnis
R6=1                                 ;  ID 3 BACH mit MOV einschalten
R7=1                                 ;  ID 5 MEAC einschalten
M_ZAEHNE=ZAEHNEZAHL*2                ;  ID rechnen neg./pos. Flanke je Zahn
Z_MW=0                               ;  ID Zähler MW FIFO auslesen bis Zähnezahl
Z_RW=2                               ;  ID Zähler Rechnen Differenz Zahnlücke
R13=HYS_POS_FLANKE                   ;  Hysterese in Rechenregister
R14=HYS_NEG_FLANKE                   ;  Hysterese in Rechenregister
;*** Achse fahren, messen, rechnen ***
N100 MEAC[BACH]=(0)                  ;  Messauftrag rücksetzen
;Rücksetzen der FIFO1[4] Variablen und Sicherstellen eines definierten Messtrace
N105 $AC_FIFO1[4]=0                  ;  FIFO1 rücksetzen
STOPRE
;  *** FIFO auslesen bis Zähnezahl erreicht ***
;  wenn FIFO1 nicht leer und noch nicht alle Zähne gemessen, Messwert aus FIFO-Vari-
   able in
;  Synchronaktionsparameter umspeichern und Zähler Messwerte erhöhen
```

```
ID=1 WHENEVER ($AC_FIFO1[4]>=1) AND (Z_MW<M_ZAEHNE)
        DO $AC_PARAM[0+Z_MW]=$AC_FIFO1[0] Z_MW=Z_MW+1
;wenn 2 Messwerte vorhanden sind, anfangen zu rechnen, NUR Lückenmaß
;rechnen und Lückensumme, Rechenwertzähler um 2 erhöhen
```

```
ID=2 WHENEVER (Z_MW>=Z_RW) AND (Z_RW<M_ZAEHNE)
        DO $R1=($AC_PARAM[-1+Z_RW]-$R13)-($AC_PARAM[-2+Z_RW]-$R14) Z_RW=Z_RW+2 $R2=
        $R2+$R1
;*** Einschalten der Achse BACH als endlos drehende Rundachse mit MOV ***
WAITP(BACH)
ID=3 EVERY $R6==1 DO MOV[BACH]=1     ;  einschalten
FA[BACH]=1000
ID=4 EVERY $R6==0 und                ;  ausschalten
($AA_STAT[BACH]==1) DO MOV[BACH]=0
; Messen nacheinander, Ablegen in FIFO 1, MT2 neg, MT2 pos Flanke
;gemessen wird der Abstand zwischen 2 Zähnen
;fallende Flanke-...-steigende Flanke, Taster 2
N310 ID=5 WHEN $R7==1 DO MEAC[BACH]=(2, 1, -2, 2)
N320 ID=6 WHEN (Z_MW>=M_ZAEHNE) DO   ;  Messung abbrechen
MEAC[BACH]=(0)
M00
STOPRE
```

```
;*** FIFO Werte holen und abspeichern ***
N400 R3=$AC_PARAM[0]                 ;  Inhalt des zuerst eingelesenen Elements
                                     ;  ;Rücksetzen der FIFO1[4] Variablen
                                     ;  ;und Sicherstellen eines definierten Messt-
                                        race
                                        ;für nächsten Messauftrag
N500 $AC_FIFO1[4]=0
```

```
;*** Differenz zwischen den einzelnen Zähnen rechen ***
N510 R4=R2/(ZAEHNEZAHL)/1000         ;  R4 Entspricht einem durchschnittlichen
                                     ;  Zahnabstand
                                     ;  Division "/1000" entfällt in späteren SW-
                                        Ständen
```

```
;*** Mittenposition berechnen ***
N520 R3=R3/1000                      ;  Erste Messposition auf Grad umgerechnet
N530 R3=R3 MOD 360                   ;  ersten Messpunkt modulo
N540 R5=(R3-R14)+(R4/2)              ;  Lückenposition rechnen
M00
stopre
R6=0                                 ;  Achsdrehung von BACH ausschalten
gotob anfang
M30
```

4.7 Achskopplungen über Synchronaktionen

4.7.1 Einkoppeln auf Leitachse

Aufgabenstellung

Über Polynomsegmente wird eine zyklische Kurventabelle definiert. Gesteuert über Rechenvariablen wird die Bewegung der Leitachse und der Koppelvorgang zwischen Leitachse und abhängiger Achse ein-/ausgeschaltet.

%_N_KOP_SINUS_MPF

```
N5 R1=1                                ;  ID 1, 2 ein-/ausschalten der Kopplung: LEAD-
                                          ON (CACB, BACH)
N6 R2=1                                ;  ID 3, 4 Leitachse bewegen ein-/aus: MOV BACH
N7 R5=36000                            ;  BACH Vorschub/min
N8 STOPRE
```

```
;*** Periodische Tabelle Nr. 4 durch Polynomsegmente definieren ***
N10 CTABDEF (YGEO,XGEO,4,1)
N16 G1 F1200 XGEO=0.000 YGEO=0.000   ;  Grundstellungen anfahren
N17 POLY PO[XGEO]=(79.944,3.420,0.210) PO[YGEO]=(24.634,0.871,-9.670)
N18 PO[XGEO]=(116.059,0.749,-0.656) PO[YGEO]=(22.429,-5.201,0.345)
N19 PO[XGEO]=(243.941,-17.234,11.489) PO[YGEO]=(-22.429,-58.844,39.229)
N20 PO[XGEO]=(280.056,1.220,-0.656) PO[YGEO]=(-24.634,4.165,0.345)
N21 PO[XGEO]=(360.000,-4.050,0.210) PO[YGEO]=(0.000,28.139,-9.670)
N22 CTABEND                            ;  *** Ende der Tabellendefinition***
```

```
; Achse Leitachse und gekoppelte Achse im Eilgang in Grundstellung fahren
N80 G0 BACH=0 CACH=0                   ;  Kanalachsnamen
N50 LEADOF(CACH,BACH)                  ;  ggf. bestehende Kopplung AUS
```

```
N235 ;*** Einschalten der Koppel-Bewegung für die Achse CACH ***
N240 WAITP(CACH)                       ;  Achse auf Kanal synchronisieren
N245 ID=1 EVERY $R1==1 DO LEAD-        ;  Über Tabelle 4 einkoppeln
ON(CACH, BACH, 4)
N250 ID=2 EVERY $R1==0 DO LEAD-        ;  Kopplung ausschalten
OF(CACH, BACH)
```

```
N265 WAITP(BACH)
```

```
N270 ID=3 EVERY $R2==1 DO                ;  Leitachse mit Vorschub in R5 endlos drehen
MOV[BACH]=1 FA[BACH]=R5
N275 ID=4 EVERY $R2==0 DO                ;  Leitachse anhalten
MOV[BACH]=0
N280 M00
N285 STOPRE
N290 R1=0                                ;  Ausschalten Koppelbedingung
N295 R2=0                                ;  Ausschalten Bedingung für Leitachse drehen
N300 R5=180                              ;  Neuer Vorschub für BACH
N305 M30
```

4.7.2 Unrundschleifen über Leitwertkopplung

Aufgabenstellung

Ein unrundes Werkstück, das sich auf der Achse CACH dreht, soll durch Schleifen bearbeitet werden. Der Abstand der Schleifscheibe vom Werkstück wird über die Achse XACH gesteuert. Er hängt von der Drehlage des Werkstückes ab. Der Zusammenhang zwischen den Drehlagen und zugeordneten Bewegungen ist durch Kurventabelle 2 definiert. Das Werkstück soll sich mit Geschwindigkeiten bewegen, die von der Werkstückkontur gemäß Kurventabelle 1 abhängen.

Lösung

CACH wird zu Leitachse einer Leitwertkopplung. Sie wirkt:

- über Tabelle 2 auf die Ausgleichsbewegung der Achse XACH
- über Tabelle 1 auf die "Softwareachse" CASW.

Der Achsoverride der Achse CACH bestimmt sich aus den Istwerten der Achse CASW. Damit ist die geforderte konturabhängige Geschwindigkeit der Achse CACH realisiert.

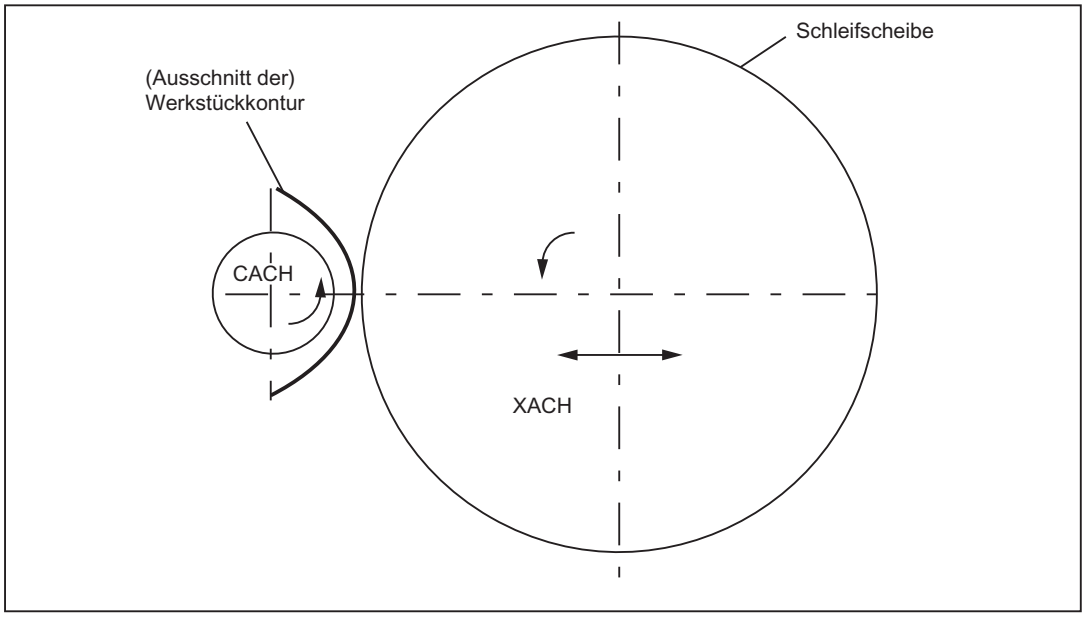


Bild 4-5 Schema Schleifen einer Unrund–Kontur

%_N_CURV_TABS_SPF		
PROC CURV_TABS		
N160 ; *** Tabelle 1 Override definieren ***		
N165 CTABDEF(CASW,CACH,1,1)	;	Tabelle 1 periodisch
N170 CACH=0 CASW=10		
N175 CACH=90 CASW=10		
N180 CACH=180 CASW=100		
N185 CACH=350 CASW=10		
N190 CACH=359.999 CASW=10		
N195 CTABEND		

N160 ; *** Tabelle 2 Lineare Ausgleichbewegung der XACH definieren ***		
CTABDEF(YGEO,XGEO,2,1)	;	Tabelle 2 periodisch
N16 XGEO=0.000 YGEO=0.000		
N16 XGEO=0.001 YGEO=0.000		
N17 POLY PO[XGEO]=(116.000,0.024,0.012) PO[YGEO]=(4.251,0.067,-0.828)		
N18 PO[XGEO]=(244.000,0.072,-0.048) PO[YGEO]=(4.251,-2.937)		
N19 PO[XGEO]=(359.999,-0.060,0.012) PO[YGEO]=(0.000,-2.415,0.828)		
N16 XGEO=360.000 YGEO=0.000		
N20 CTABEND		
M17		

%_N_UNRUND_MPF

; Koppelverbund für eine Unrundbearbeitung

; XACH ist die Zustellachse der Schleifscheibe

; CACH ist die Werkstückachse als Rundachse und Leitwertachse

; Anwendung: Unrunde Kontur schleifen

; Tabelle 1 bildet den Override für Achse CACH als Funktion der Position von CACH ab

; Überlagerung der XGEO Achse mit Handrad Zustellung für Ankratzen

N100 DRFOF	;	Handradüberlagerung abwählen
N200 MSG("DRF anwählen, (Handrad 1 aktiv) und Anwahl INKREMENT.== Handradueberlagerung AKTIV")		
N300 M00		
N500 MSG()	;	Meldung rücksetzen
N600 R2=1	;	LEADON Tabelle 2, Einschalten mit ID=3/4 CACH auf XACH
N700 R3=1	;	LEADON Tabelle 1 , Einschalten mit ID=5/6 CACH auf CASW, Override
N800 R4=1	;	Endlos drehende Rundachse CACH, Start mit ID=7/8
N900 R5=36000	;	FA[CACH] Endlos drehende Rundachse Drehzahl

N1100 STOPRE		
N1200	;	*** Achsen und Leitachse auf FA einstellen ***
	; ;	Achse Leitachse und Folgeachse in Grundstellung fahren
N1300 G0 XGEO=0 CASW=10 CACH=0		
N1400 LEADOF(XACH,CACH)	;	Kopplung AUS XACH Ausgleichsbewegung
N1500 LEADOF(CASW,CACH)	;	Kopplung AUS CASW Overridetabelle
N1600 CURV_TABS	;	Unterprogramm mit der Definition der Tabellen

N1700	;	*** Einschalter der LEADON Ausgleichbewegung XACH ***
N1800 WAITP(XGEO)	;	Achse auf Kanal synchronisieren
N1900 ID=3 EVERY $R2==1 **DO** LEAD-ON(XACH,CACH,2)		
N2000 ID=4 EVERY $R2==0 **DO** LEAD-OF(XACH,CACH)		

N2100	; ;	*** Einschalter der LEADON CASW Overridetabelle ***
N2200 WAITP(CASW)		
N2300 ID=5 EVERY $R3==1 **DO** LEAD-ON(CASW,CACH,1)	;	CTAB Kopplung EIN Leitachse CACH
N2400 ID=6 EVERY $R3==0 **DO** LEAD-OF(CASW,CACH)	;	CTAB Kopplung AUS Leitachse CACH

N2500	; ;	*** Override der CACH von Position CASW mit ID 10 beeinflussen ***
N2700 ID=11 **DO** $$AA_OVR[CACH]= $AA_IM[CASW]	;	"Achsposition" CASW auf OVR CACH zuweisen

N2900 WAITP(CACH)		
N3000 ID=7 EVERY $R4==1 **DO** MOV[CACH]=1 FA[CACH]=R5	;	Als endlos drehende Rundachse starten
N3100 ID=8 EVERY $R4==0 **DO** MOV[CACH]=0	;	Als endlos drehende Rundachse anhalten

N3200 STOPRE		
N3300 R90=$AA_COUP_ACT[CASW]	;	Zustand der Kopplung für CASW zum Prüfen
N3400 MSG("Overridetabelle CASW eingeschalten mit LEADON "<<R90<<", weiter ENDE mit NC-START")		

N3500 M00	;	*** NC HALT ***
N3600 MSG()		
N3700 STOPRE	;	Vorlaufstopp
N3800 R1=0	; ;	Stopp mit ID=2 CASW Achse als endlos drehende Rundachse
N3900 R2=0	; ;	LEADOF mit ID=6 FA XACH und Leitachse CACH
N4000 R3=0	; ;	LEADOF TAB1 CASW mit ID=7/8 CACH auf CASW Overridetabelle
N4100 R4=0	; ;	Achse als endlos drehende Rundachse anhalten, ID=4 CACH
N4200 M30		

Ausbaumöglichkeiten

Das obige Beispiel lässt sich in folgenden Punkten ausbauen:

- Einführung einer Z-Achse, um Schleifscheibe oder Werkstück von einem Unrund zum nächsten auf der gleichen Welle zu bewegen (Nockenwelle).
- Tabellenumschaltungen, wenn die Nocken z. B. für Einlass und Auslass verschiedene Konturen haben.
  `ID = ... <Bedingung> DO LEADOF(XACH, CACH) LEADON(XACH, CACH, <neue Tabellennummer>)`
- Abrichten der Schleifscheibe über online Werkzeugkorrektur gem. Kap. "Online-Werkzeugkorrektur FTOC".

4.7.3 Fliegendes Trennen

Aufgabenstellung

Ein Strangmaterial, das sich stetig durch einen Arbeitsbereich einer Trennvorrichtung bewegt, soll in gleichlange Stücke zerteilt werden.

X-Achse: Achse in der sich das Strangmaterial bewegt, WKS

X1-Achse: Maschinenachse des Strangmaterials, MKS

Y-Achse: Achse, in der die Trennvorrichtung mit dem Strangmaterial "mitfährt"

Es wird angenommen, dass die Zustellung des Trennwerkzeuges und seine Steuerung durch PLC kontrolliert werden. Zur Feststellung der Synchronität zwischen Strangmaterial und Trennwerkzeug können die Signale der PLC-Nahtstelle ausgewertet werden.

Aktionen

Kopplung einschalten, `LEADON`

Kopplung ausschalten, `LEADOF`

Istwertsetzen, `PRESETON`

Programmcode	**Kommentar**
`%_N_SCHERE1_MPF`	
`;$PATH=/_N_WKS_DIR/_N_DEMOFBE_WPD`	
`N100 R3=1500`	`; Länge eines abzutrennenden Teils`
`N200 R2=100000 R13=R2/300`	
`N300 R4=100000`	
`N400 R6=30`	`; Startposition Y Achse`
`N500 R1=1`	`; Startbedingung für Bandachse`
`N600 LEADOF(Y,X)`	`; Löschen einer evtl. bestehenden Kopplung`
`N700 CTABDEF(Y,X,1,0)`	`; Tabellendefinition`
`N800 X=30 Y=30`	`; Wertepaare`
`N900 X=R13 Y=R13`	
`N1000 X=2*R13 Y=30`	
`N1100 CTABEND`	`; Ende der Tabelledefinition`
`N1200 PRESETON(X1,0)`	`; Preset-Verschiebung zu Beginn`
`N1300 Y=R6 G0`	`; Startposition Y Achse`
	`; Achse ist Linearachse`
`N1400 ID=1 EVERY $AA_IW[X]>$R3 DO PRESETON(X1,0)`	`; Preset-Verschiebung nach Länge R3, PRESETON darf nur mit WHEN und EVERY erfolgen`
	`; neuer Beginn nach Abtrennen`
`N1500 WAITP(Y)`	
`N1800 ID=6 EVERY $AA_IM[X] < 10 DO LEADON(Y,X,1)`	`; Y über Tabelle 1 an X ankoppeln bei X < 10`

Programmcode	**Kommentar**
N1900 ID=10 EVERY $AA_IM[X] > $R3-30 DO LEADOF(Y,X)	; > 30 vor gefahrener Trennlänge abkop- peln
N2000 WAITP(X)	
N2100 ID=7 WHEN $R1==1 DO MOV[X]=1 FA[X]=$R4	; Strangachse stetig in Bewegung setzen
N2200 M30	

4.8 Technologiezyklen Spindel Positionieren

Anwendung

Im Zusammenwirken mit dem PLC-Programm soll die Spindel, die einen Werkzeugwechsel antreibt:

- in eine Ausgangsstellung positioniert werden,
- auf einen bestimmten Wert positioniert werden, auf dem sich das einzuwechselnde Werkzeug befindet.

Vergl. Kap. "Starten von Kommandoachsen" und Kap. "Beeinflussung von PLC".

Koordinierung

Die Koordinierung zwischen PLC und NCK erfolgt über die ab SW-Stand 4 verfügbaren gemeinsamen Daten (siehe Kap. "Liste der für Synchronaktionen bedeutsamen Systemvariablen")

- $A_DBB[0]: 1 Grundposition einnehmen,
- $A_DBB[1]: 1 Zielposition einnehmen,
- $A_DBW[1]: zu positionierender Wert +/- , PLC berechnet den kürzesten Weg.

Synchronaktionen

%_N_MAIN_MPF

```
...
IDS=1 EVERY $A_DBB[0]==1 DO NULL_POS  ;  ; wenn $A_DBB[0] von PLC gesetzt,
                                      ;  Grundposition einnehmen
IDS=2 EVERY $A_DBB[1]==1 DO ZIEL_POS  ;  wenn $A_DBB[1] von PLC gesetzt,
                                      ;  Spindel auf den in
                                      ;  $A_DBW[1] hinterlegten Wert positionieren
...
```

Technologiezyklus NULL_POS

%_N_NULL_POS_SPF

```
PROC NULL_POS
```

```
SPOS=0                          ;  Antrieb für den Werkzeugwechsel
                                ;  in Grundposition bringen
$A_DBB[0]=0                     ;  Grundposition in NCK ausgeführt
```

Technologiezyklus ZIEL_POS

%_N_ZIEL_POS_SPF

```
PROC ZIEL_POS
SPOS=IC($A_DBW[1])              ;  Spindel auf den Wert positionieren,
                                ;  der in $A_DBW[1]
                                ;  von PLC hinterlegt wurde, Kettenmaß
$A_DBB[1]=0                     ;  Zielpositionieren in NCK ausgeführt
```

4.9 Synchronaktionen im Bereich WZW/BAZ

Einführung

Das folgende Bild zeigt den schematischen Ablauf Werkzeugwechselzyklus.

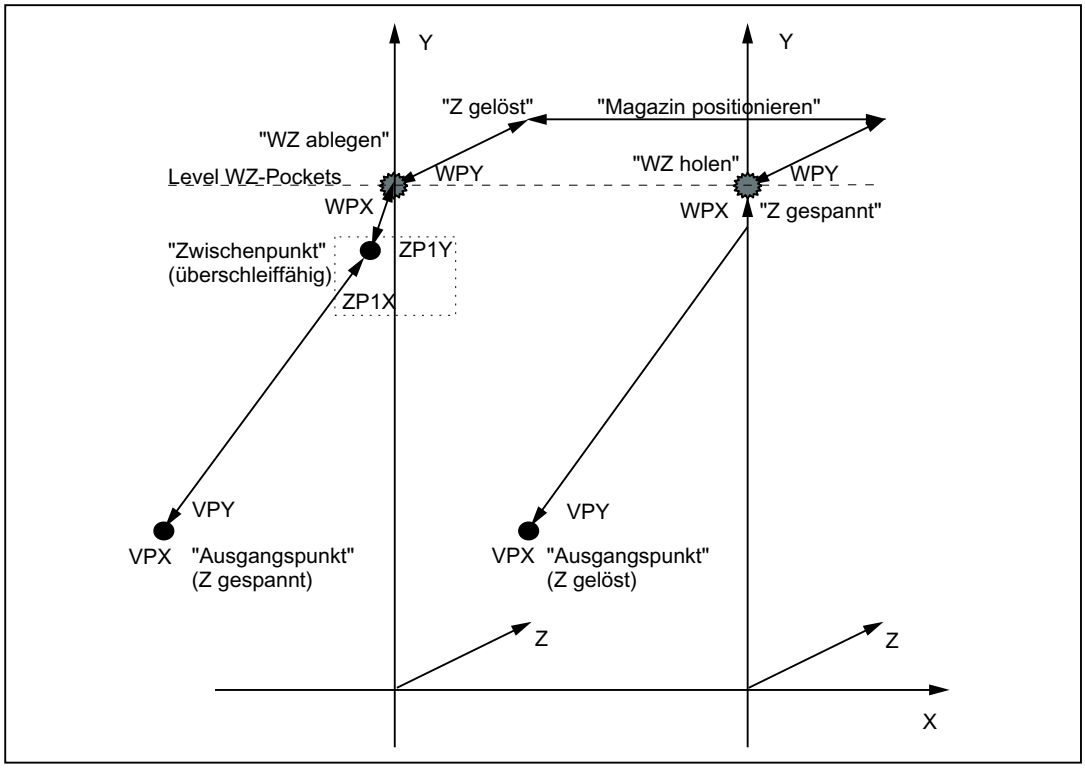


Bild 4-6 Schematischer Ablauf Werkzeugwechselzyklus

Ablaufdiagramm

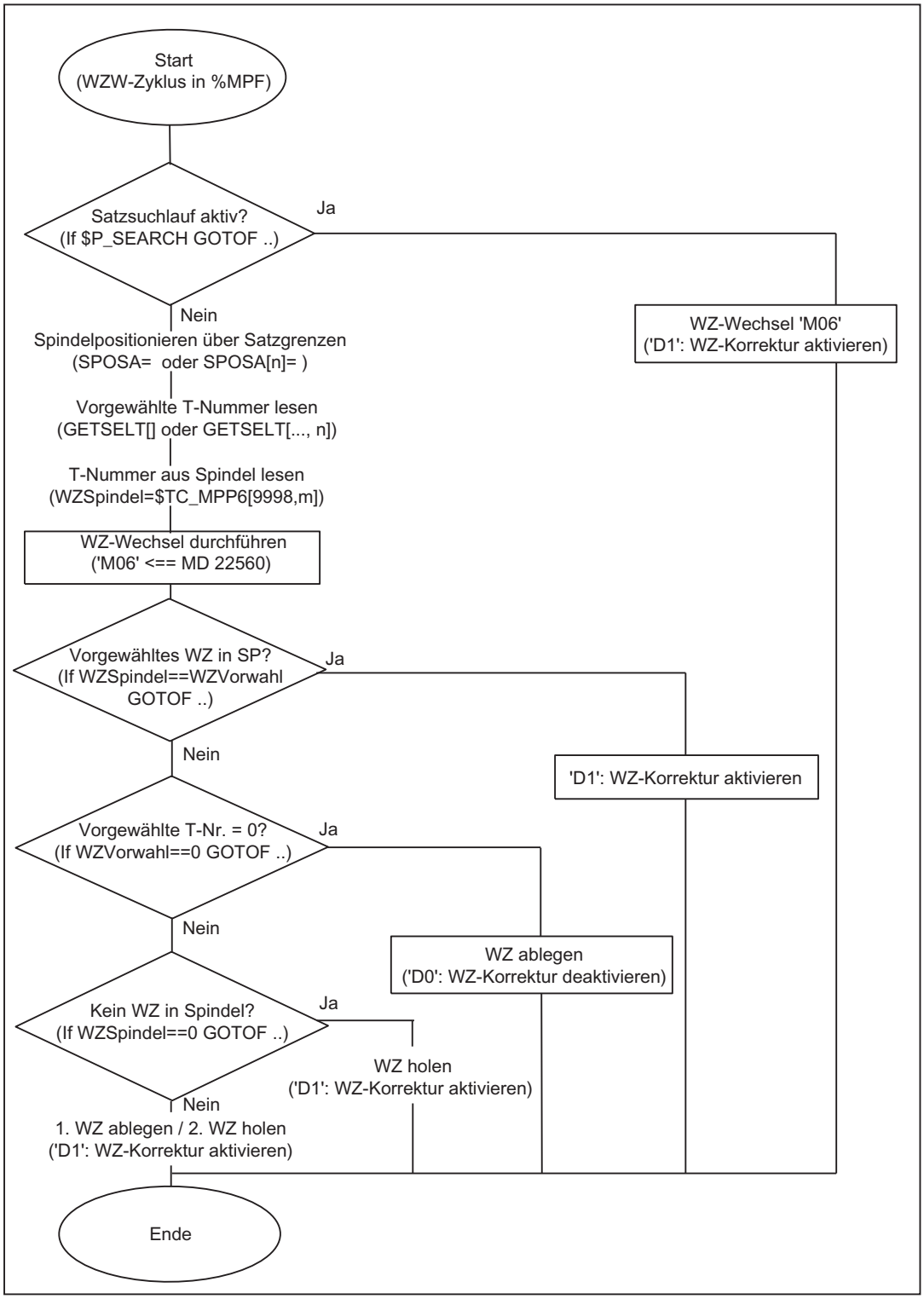


Bild 4-7 Ablaufdiagramm Werkzeugwechselzyklus

NC-Programm	**Kommentar**
%_N_WZW_SPF	
;$PATH=/_N_SPF_DIR	
N10 DEF INT WZVorwahl,WZSpindel	; Marker auf = 1 wenn MagAchse gefahren
N15 WHEN $AC_PATHN<10 DO $AC_MARKER[0]=0 $AC_MARKER[1]=0 $AC_MARKER[2]=0	
N20 ID=3 WHENEVER $A_IN[9]==TRUE DO $AC_MARKER[1]=1	
N25 ID=4 WHENEVER $A_IN[10]==TRUE DO $AC_MARKER[2]=1	; Marker auf = 1 wenn MagAchse gefahren
N30 IF $P_SEARCH GOTOF wzw_vorlauf	; Satzvorlauf aktiv ? ->
N35 SPOSA=0 D0	
N40 GETSELT(WZVorwahl)	; vorgewählte T-Nr. lesen
N45 WZSpindel=$TC_MPP6[9998,1]	; WZ in Spindel lesen
N50 M06	
N55 IF WZSpindel==WZVorwahl GOTOF wz_in_spindel IF WZVorwahl==0 GOTOF ablegen1 IF WZSpindel==0 GOTOF holen1	
;*Werkzeug holen und ablegen*****	
ablegen1holen1:	
N65 WHENEVER $AA_VACTM[C2]<>0 DO $AC_MARKER[1]=1	; wenn MagAchse fährt Marker = 1
N70 G01 G40 G53 G64 G90 X=Magazin1VPX Y=Magazin1VPY Z=Magazin1ZGespannt F70000 M=QU(120) M=QU(123) M=QU(9)	
N75 WHENEVER $AA_STAT[S1]<>4 DO $AC_OVR=0	; Spindel in Position
N80 WHENEVER $AA_VACTM[C2]<>0 DO $AC_MARKER[1]=1	; MagAchse fährt abfragen
N85 WHENEVER $AC_MARKER[1]==0 DO $AC_OVR=0	; Override=0 wenn Achse nicht gefahren
N90 WHENEVER $AA_STAT[C2]<>4 DO $AC_OVR=0	; Override=0 wenn MagAchse ; nicht in Pos fein
N95 WHENEVER $AA_DTEB[C2]>0 DO $AC_OVR=0	; Override=0 wenn Restweg MagAchse > 0
N100 G53 G64 X=Magazin1ZP1X Y=Magazin1ZP1Y F60000	
N105 G53 G64 X=Magazin1WPX Y=Magazin1WPY F60000	
N110 M20	; WZ lösen
N115 G53 G64 Z=MR_Magazin1ZGeloest F40000	
N120 WHENEVER $AA_VACTM[C2]<>0 DO $AC_MARKER[2]=1;	
N125 WHENEVER $AC_MARKER[2]==0 DO $AC_OVR=0	
N130 WHENEVER $AA_STAT[C2]<>4 DO $AC_OVR=0	
N135 WHENEVER $AA_DTEB[C2]>0 DO $AC_OVR=0	
N140 G53 G64 Z=Magazin1ZGespannt F40000	
N145 M18	; Werkzeug spannen
N150 WHEN $AC_PATHN<10 DO M=QU(150) M=QU(121)	; Bedingung immer erfüllt
N155 G53 G64 X=Magazin1VPX Y=Magazin1VPY F60000 D1 M17	

NC-Programm	Kommentar
;*Werkzeug ablegen*****	
ablegen1:	
N160 WHENEVER $AA_VACTM[C2]<>0 DO $AC_MARKER[1]=1	
N165 G01 G40 G53 G64 G90 X=Magazin1VPX Y=Magazin1VPY Z=Magazin1ZGespannt F70000 M=QU(120) M=QU(123) M=QU(9)	
N170 WHENEVER $AA_STAT[S1]<>4 DO $AC_OVR=0	
N175 WHENEVER $AA_VACTM[C2]<>0 DO $AC_MARKER[1]=1	
N180 WHENEVER $AC_MARKER[1]==0 DO $AC_OVR=0	
N185 WHENEVER $AA_STAT[C2]<>4 DO $AC_OVR=0	
N190 WHENEVER $AA_DTEB[C2]>0 DO $AC_OVR=0	
N195 G53 G64 X=Magazin1ZP1X Y=Magazin1ZP1Y F60000	
N200 G53 G64 X=Magazin1WPX Y=Magazin1WPY F60000	
N205 M20	; Werkzeug lösen
N210 G53 G64 Z=Magazin1ZGeloest F40000	
N215 G53 G64 X=Magazin1VPX Y=Magazin1VPY F60000 M=QU(150) M=QU(121) D0 M17	
;*Werkzeug holen*****	
holen1:	
N220 WHENEVER $AA_VACTM[C2]<>0 DO $AC_MARKER[2]=1	
N225 G01 G40 G53 G64 G90 X=Magazin1VPX Y=Magazin1VPY Z=Magazin1ZGeloest F70000 M=QU(120) M=QU(123) M=QU(9)	
N230 G53 G64 X=Magazin1WPX Y=Magazin1WPY F60000	
N235 WHENEVER $AA_STAT[S1]<>4 DO $AC_OVR=0	
N240 WHENEVER $AA_VACTM[C2]<>0 DO $AC_MARKER[2]=1	
N245 WHENEVER $AC_MARKER[2]==0 DO $AC_OVR=0	
N250 WHENEVER $AA_STAT[C2]<>4 DO $AC_OVR=0	
N255 WHENEVER $AA_DTEB[C2]>0 DO $AC_OVR=0	
N260 G53 G64 Z=Magazin1ZGespannt F40000	
N265 M18	; Werkzeug spannen
N270 G53 G64 X=Magazin1VPX Y=Magazin1VPY F60000 M=QU(150) M=QU(121) D1 M17	
;*Werkzeug in Spindel *****	
wz_in_spindel:	
N275 M=QU(121) D1 M17	
;*Satzvorlauf*****	
wzw_vorlauf:	
N280 STOPRE	
N285 D0	
N290 M06	
N295 D1 M17	

Datenlisten

5

5.1 Maschinendaten

5.1.1 Allgemeine Maschinendaten

Nummer	Bezeichner: $MN_	Beschreibung
11110	AUXFU_GROUP_SPEC	Hilfsfunktionsgruppenspezifikation
11500	PREVENT_SYNACT_LOCK	Geschützte Synchronaktionen
18860	MM_MAINTENANCE_MON	Aktivierung der Aufzeichnung von Wartungsdaten

5.1.2 Kanalspezifische Maschinendaten

Nummer	Bezeichner: $MC_	Beschreibung
21240	PREVENT_SYNACT_LOCK_CHAN	Geschützte Synchronaktionen des Kanals
28250	MM_NUM_SYNC_ELEMENTS	Anzahl Elemente für Ausdrücke der Synchronaktionen
28252	MM_NUM_FCTDEF_ELEMENTS	Anzahl der FCTDEF-Elemente
28254	MM_NUM_AC_PARAM	Parameteranzahl $AC_PARAM
28255	MM_BUFFERED_AC_PARAM	Speicherort für $AC_PARAM
28256	MM_NUM_AC_MARKER	Merkeranzahl $AC_MARKER
28257	MM_BUFFERED_AC_MARKER	Speicherort für $AC_MARKER
28258	MM_NUM_AC_TIMER	Anzahl Zeitvariablen $AC_TIMER
28260	NUM_AC_FIFO	Anzahl Variablen $AC_FIFO1, $AC_FIFO2, ...
28262	START_AC_FIFO	FIFO-Variablen speichern ab R-Parameter
28264	LEN_AC_FIFO	Länge der FIFO-Variablen $AC_FIFO ...
28266	MODE_AC_FIFO	Modus der FIFO-Bearbeitung

5.1.3 Achsspezifische Maschinendaten

Nummer	Bezeichner: $MA_	Beschreibung
30450	IS_CONCURRENT_POS_AX	Konkurrierende Positionierachse
32060	POS_AX_VELO	Löschstellung für Positionierachsgeschwindigkeit
32070	CORR_VELO	Achsgeschwindigkeit für Handrad, ext. NPV, cont. Dressing, Abstandsregelung

Nummer	Bezeichner: $MA_	Beschreibung
32074	FRAME_OR_CORRPOS_NOTALLOWED	Wirksamkeit der Frames und Werkzeuglängenkorrektur
32920	AC_FILTER_TIME	Filter-Glättungszeitkonstante für Adaptive Control
33060	MAINTENANCE_DATA	Konfiguration der Aufzeichnung von Wartungsdaten
36750	AA_OFF_MODE	Wirkung der Wertzuweisung für axiale Überlagerung bei Synchronaktionen
37200	COUPLE_POS_TOL_COARSE	Schwellwert für "Synchronlauf grob"
37210	COUPLE_POS_TOL_FINE	Schwellwert für "Synchronlauf fein"

5.2 Settingdaten

5.2.1 Achs-/Spindel-spezifische Settingdaten

Nummer	Bezeichner: $SA_	Beschreibung
43300	ASSIGN_FEED_PER_REV_SOURCE	Umdrehungsvorschub für Positionierachsen/Spindeln
43350	AA_OFF_LIMIT	Obergrenze des Korrekturwertes für $AA_OFF Abstandsregelung
43400	WORKAREA_PLUS_ENABLE	Arbeitsfeldbegrenzung in pos. Richtung

5.3 Signale

5.3.1 Signale an Kanal

Signalname	SINUMERIK 840D sl	SINUMERIK 828D
Alle Synchronaktionen sperren	DB21, ...DBX21.2	-
Synchronaktionen ID/IDS 1 - 64 sperren (allgemeine Anforderung)	DB21, ...DBX280.1	-
Synchronaktion ID/IDS 1 - 64 sperren	DB21, ...DBX300.0 - 307.7	DB460x.DBX0.0 - 7.7

5.3.2 Signale von Kanal

Signalname	SINUMERIK 840D sl	SINUMERIK 828D
Synchronaktionen ID/IDS 1 - 64 gesperrt (allgemeine Rückmeldung)	DB21, ...DBX281.1	-
Synchronaktion ID/IDS 1 - 64 sperrbar	DB21, ...DBX308.0 - 315.7	DB470x.DBX0.0 - 7.7

Anhang

A

A.1 Dokumentationsübersicht

Allgemeine Dokumentation

Werbeschrift
– SINUMERIK 840D sl
– SINUMERIK 828D
– SINUMERIK 828D BASIC T

Katalog NC 62
SINUMERIK 840D sl
Typ 1B

Katalog NC 82
SINUMERIK 828D
BASIC T/BASIC M
SINAMICS S120 Combi
Motoren 1FK7 und 1PH8

Katalog PM 21
SIMOTION,
SINAMICS S120
und Motoren für
Produktionsmaschinen

Projektierungsanleitung
EMV-Aufbaurichtlinie

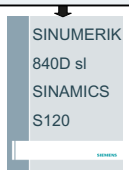
Systemhandbuch
Ctrl-Energy

Anwender-Dokumentation

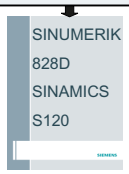
Bedienhandbuch
– Universal
– Drehen
– Fräsen

Programmierhandbuch
– Grundlagen
– Arbeitsvorbereitung
– Messzyklen

Programmierhandbuch
– ISO Drehen
– ISO Fräsen



Diagnosehandbuch



Diagnosehandbuch

Hersteller- / Service-Dokumentation

Gerätehandbuch
– NCU
– Bedienkomponenten und Vernetzung

Gerätehandbuch
Inbetriebnahmehandbuch
Servicehandbuch

Inbetriebnahmehandbuch
– CNC: NCK, PLC, Antrieb
– Basesoftware und Bedien-Software

Listenhandbuch
– Maschinendaten
– Nahtstellensignale
– Variablen

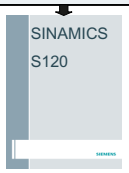
Listenhandbuch
– Maschinendaten
– Nahtstellensignale
– Parameter
– Variablen

Systemhandbuch
Leitfaden für die Maschinen-projektierung

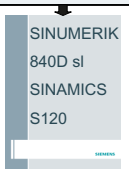
Hersteller- / Service-Dokumentation

Funktionshandbuch
– Grundfunktionen
– Erweiterungsfunktionen
– Sonderfunktionen
– Synchronaktionen
– ISO-Dialekte

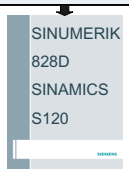
Funktionshandbuch
Werkzeugverwaltung



Funktionshandbuch
Antriebsfunktionen



Funktionshandbuch
Safety Integrated



Funktionshandbuch
Safety Integrated

Info / Training

Trainingsunterlage
– Einfacher Fräsen mit ShopMill
– Einfacher Drehen mit ShopTurn

Handbuch
Werkzeug- und Formenbau

Elektronische Dokumentation

DOConCD

My Documentation Manager

Industry Mall

Index

$

A

B

C

D

E

F

G

H

I

L

M

N

P

R

S

T

W