SIEMENS

SINUMERIK

SINUMERIK 840D sl Synchronaktionen

Funktionshandbuch

Gültig für

Steuerung
SINUMERIK 840D sl / 840DE sl

Software Version
CNC-Software 4.4

09/2011
6FC5397-5BP40-2AA0

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

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

VORSICHT

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

ACHTUNG

bedeutet, dass ein unerwünschtes Ergebnis oder Zustand eintreten kann, wenn der entsprechende Hinweis nicht beachtet wird.

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
Industry Sector
Postfach 48 48
90026 NÜRNBERG
DEUTSCHLAND

Dokumentbestellnummer: 6FC5397-5BP40-0AA0
℗ 01.2011

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 zusammenstellen:

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

Kurzbeschreibung

1

Synchronaktionen

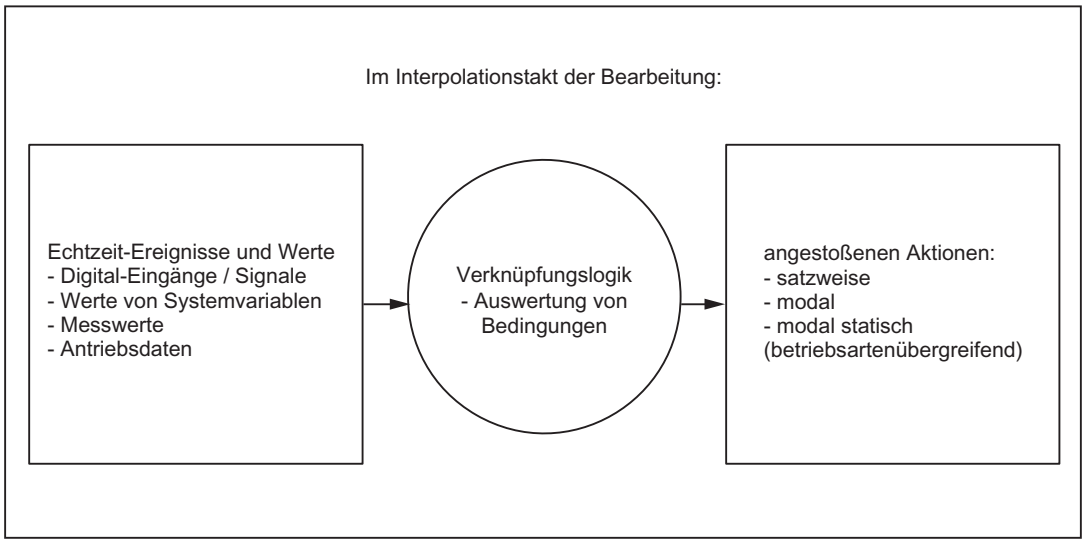
Bewegungssynchronaktionen (kurz Synchronaktionen) sind vom Anwender programmierte Anweisungen, die synchron zur Bearbeitung des Teileprogramms im Interpolationstakt vom NCK ausgewertet werden. Ist die in der Synchronaktion enthaltene Bedingung erfüllt oder keine angegeben, so werden zugeordnete Aktionen synchron zur weiteren Bearbeitung aktiviert.

Anwendungen

Mögliche Aktionen in Synchronaktionen sind z. B.:

- Ausgabe von Hilfsfunktionen an die PLC
- Schreiben und Lesen von Hauptlaufvariablen
- Positionierung von Achsen/Spindeln
- Aktivierung von Synchronprozeduren wie z. B.:
  - Einlesesperre
  - Restweglöschen
  - Vorlaufstopp beenden
- Aktivierung von Technologiezyklen
- Online-Berechnung von Funktionswerten
- Online-Werkzeugkorrekturen
- Kopplungen/Mitschleppen aktivieren/deaktivieren
- Messungen ausführen
- Sperren/Freigeben von Synchronaktionen

Synchronaktionen schematisch



Dokumentation

Die folgenden Kapitel beschreiben die funktionalen Zusammenhänge für Synchronaktionen.

Details zur Programmierung von Synchronaktionen finden Sie in:
Literatur:
Programmierhandbuch Arbeitsvorbereitung

Ausführliche Beschreibung

2

2.1 Definition von Synchronaktionen

Die Definition von Synchronaktionen erfolgt im Teileprogramm.

Statische Synchronaktionen können in einem asynchronen Unterprogramm (ASUP) definiert werden, das durch die PLC aktiviert wird.

2.2 Komponenten von Synchronaktionen

Komponenten (Übersicht)

Die Definition einer Synchronaktion besteht aus folgenden Komponenten:

	Komponenten						
	Gültigkeit, Iden-tifi-ka-tio ns-nummer	**Häufigkeit**	**G-Code für Bedingung und Aktion**	**Bedingung**	**Aktions-ke nn-wort (fest)**	**G-Code für Aktion**	**Aktion oder Tech-no-lo-gi e-zy-klus**
Beispiel:	IDS=1	EVERY	G70	$AAA_IM[B] > 15	DO	G71	POS[X]= 100

2.2.1 Gültigkeit, Identifikationsnummern

Gültigkeit

Für die Festlegung des Gültigkeitsbereichs einer Synchronaktion bieten sich folgende Möglichkeiten:

Gültigkeitsangabe	Wirkung
Keine Angabe	Synchronaktionen ohne Gültigkeitsangabe wirken: • satzweise, d. h. nur für den darauf folgenden Satz • nur im AUTOMATIK-Betrieb • modal über alle Vorlaufstopp-Sätze (auch implizit erzeugte) und über implizit erzeugte Zwischensätze
ID	Synchronaktionen mit Gültigkeitskennung ID wirken: • modal in darauf folgend programmierten Sätzen • nur im AUTOMATIK-Betrieb Wirkungsbegrenzung: • bis eine andere Synchronaktion mit gleicher Identifikationsnummer programmiert wird • bis zum Löschen mit `CANCEL(i)` Siehe "Koordinierungen zwischen Synchronaktionen, Technologiezyklen, Teileprogramm (und PLC) [Seite 92]".
IDS (Option)	Statisch wirksame Synchronaktionen, die mit der Gültigkeitskennung IDS programmiert werden, sind in allen Betriebsarten aktiv. Sie werden auch als statische Synchronaktionen bezeichnet. Anwendungen: • AC-Schleifen auch in Betriebsart JOG aktiv • Verknüpfungslogik für Safety Integrated • Überwachungsfunktionen, Reaktion auf Maschinenzustände in allen Betriebsarten • Optimierung des Werkzeugwechsels • Zyklische Maschinen

Hinweis

Das Löschen von Synchronaktionen mit ID oder IDS erfolgt aus dem Teileprogramm.

Identifikationsnummern

Für modale Synchronaktionen (ID, IDS) werden Identifikationsnummern zwischen 1 und 255 vergeben. Sie sind für die Funktionen der gegenseitigen Koordinierung von Synchronaktionen von Bedeutung (siehe "Koordinierungen zwischen Synchronaktionen, Technologiezyklen, Teileprogramm (und PLC) [Seite 92]").

Modale/statische Synchronaktionen mit Identifikationsnummern von 1 bis 64 können von der PLC aus verriegelt und freigegeben werden (siehe "Beeinflussung durch die PLC [Seite 93]").

Die Identifikationsnummern müssen im Kanal eindeutig vergeben werden.

Beispiele:

Programmcode	**Kommentar**
`IDS=1 EVERY $A_IN[1]==1 DO POS[X]=100`	`alle Betriebsarten`
`ID=2 EVERY $A_IN[1]==0 DO POS[X]=0`	`AUTOMATIK`

Hinweis

Folgende Aktionen sind nur in der Betriebsart AUTOMATIK bei aktivem Programm wirksam:

- `STOPREOF`
- `DELDTG`

2.2.2 Häufigkeit

Schlüsselwort zur Abfrage-Häufigkeit

Durch ein Schlüsselwort (siehe Tabelle) wird angegeben, wie oft die darauf folgende Bedingung abgefragt und die zugehörende Aktion bei erfüllter Bedingung ausgeführt werden soll. Das angegebene Schlüsselwort ist Bestandteil der Synchronaktionsbedingung.

Schlüsselwort	Abfrage-Häufigkeit
Keine Angabe	Ist keine Häufigkeitsangabe programmiert, so wird die Aktion zyklisch in jedem Interpolationstakt ausgeführt.
WHENEVER	Die zugehörigen Aktion/Technologiezyklus wird zyklisch in jedem Interpolationstakt ausgeführt, solange die Bedingung erfüllt ist.
FROM	Wenn die Bedingung einmal erfüllt ist, wird die Aktion/Technologiezyklus zyklisch in jedem Interpolationstakt ausgeführt, solange die Synchronaktion aktiv ist.
WHEN	Wenn die Bedingung erfüllt ist, wird die Aktion/Technologiezyklus ein einziges Mal ausgeführt. Ist die Aktion einmal ausgeführt, so wird die Bedingung nicht mehr überprüft.
EVERY	Die Aktion/Technologiezyklus wird einmal angestoßen, wenn die Bedingung erfüllt ist. Die Aktion/ Technologiezyklus wird wieder ausgeführt, wenn die Bedingung vom Zustand falsch in den Zustand wahr übergeht. Im Gegensatz zum Schlüsselwort WHEN bleibt die Überprüfung der Bedingung nach Ausführung der Aktion solange aktiv, bis die Synchronaktion gelöscht oder inaktiv geschaltet wird.

Hinweis

Informationen zu Technologiezyklen siehe Kapitel "Technologiezyklen [Seite 89]".

2.2.3 G-Code für Bedingung und Aktion

Definierter Ausgangszustand

Durch die Möglichkeit der Programmierung eines G-Codes für die Bedingung und Aktion einer Synchronaktion wird erreicht, dass unabhängig vom gerade aktiven Teileprogrammzustand für die Auswertung der Bedingung und die auszuführende Aktion/ Technologiezyklus definierte Einstellungen bestehen.

Die Abkopplung der Synchronaktionen vom Programmumfeld ist erforderlich, weil Synchronaktionen zu beliebigen Zeitpunkten aufgrund erfüllter Auslösebedingungen ihre Aktionen in definiertem Ausgangszustand ausführen sollen.

Hinweis

Pro Bedingungsanteil darf nur ein G-Code der G-Code-Gruppe programmiert werden.

Ein angegebener G-Code bei der Bedingung gilt für die Auswertung der Bedingung **und** für die Aktion, wenn bei der Aktion kein eigener G-Code angegeben ist.

Anwendungsfälle

- Festlegung der Maßsysteme für Bedingungsauswertung und Aktion durch die G-Codes `G70,G71,G700,G710`.

2.2.4 Bedingung

Bedingungen

Die Ausführung der Aktionen/Technologiezyklen kann von einer Bedingung (**logischer Ausdruck**) abhängig gemacht werden.

Die Bedingung wird im Interpolationstakt überprüft. Ist keine Bedingung angegeben, so wird die Aktion zyklisch in jedem Interpolationstakt ausgeführt.

Mögliche Bedingungen sind z. B.:

- Vergleich von Hauptlaufvariablen wie digitale oder analoge Ein–/Ausgänge
- Berechnung von Echtzeit-Ausdrücken

  (siehe "Auswertungen und Berechnungen in Echtzeit [Seite 10]")
- Zeit / Entfernung vom Satzanfang
- Messwerte, Messergebnisse
- Servo-Werte
- Geschwindigkeiten, Achsstatus

Beispiele:

Programmcode
`WHENEVER $AA_IM[X] > 10.5*SIN(45) DO ...`

oder

Programmcode
`WHENEVER $AA_IM[X] > $$AA_IM[X1] DO ...`

Weitere Beispiele zu Bedingungen siehe "Beispiele für Bedingungen in Synchronaktionen [Seite 1]".

Boole’sche Verknüpfungen

Vergleiche können über Boole’sche Verknüpfungen miteinander verbunden werden.

Zulässig sind die Boole’schen Operatoren der NC-Sprache:

`NOT, AND, OR, XOR, B_OR, B_AND, B_XOR, B_NOT`

Beispiel:

Programmcode	**Kommentar**
`WHENEVER ($A_IN[1]==1) OR ($A_IN[3]==0) DO ...`	`; solange Eingang 1 ansteht oder Eingang 3 nicht ansteht ...`

Vergleich von Echtzeit-Ausdrücken

Innerhalb einer Bedingung können zwei oder mehr Echtzeit-Ausdrücke miteinander verglichen werden. Vergleiche sind jeweils möglich zwischen **typgleichen** Variablen oder Teilausdrücken.

Beispiel:

Programmierung	**Kommentar**
`WHEN $AA_IM[X2] <= $AA_IM[X1] +.5 DO $AA_OVR[X1]=0`	`; Anhalten, wenn Sicherheitsabstand überschritten ist.`

Systemvariablen

Bei den Bedingungen können alle Systemvariablen angesprochen werden, deren Daten von der NC über Synchronaktionen gelesen bzw. geschrieben werden. Darüber hinaus alle Maschinendaten und Settingdaten mit Lesen des Werts im Hauptlauf:

- Maschinendaten: z. B. $$MN_..., $$MC_..., $$MA_...
- Settingdaten: z. B. $$SN_..., $$SC_..., $$SA_...

Hinweis

R-Parameter werden mit $R... adressiert.

Settingdaten und Maschinendaten, deren Wert sich während der Bearbeitung ändern kann, müssen mit **$$**S._.../**$$**M._... programmiert werden.

2.2.5 G-Code für Aktion

Dieser G-Code gibt für alle Aktionen im Satz und Technologiezyklen ggf. einen anderen G-Code vor als den bei der Bedingung gesetzten. Sind Technologiezyklen im Aktionsteil, so gilt der G-Code auch nach Abschluss des Technologiezyklus für alle darauf folgenden Aktionen bis zum nächsten G-Code modal weiter.

Hinweis

Pro Aktionsteil darf nur ein G-Code der G-Code-Gruppe programmiert werden.

2.2.6 Aktion oder Technologiezyklus

Aktionen

In jeder Synchronaktion werden eine oder mehrere Aktionen programmiert. Diese werden ausgeführt, wenn die Bedingung erfüllt ist. Sind mehrere Aktionen in einer Synchronaktion programmiert, so werden diese im selben Interpolationstakt ausgeführt.

Beispiel:

Programmcode	**Kommentar**
`WHEN $AA_IM[Y] >= 35.7 DO M135 $A_OUT[1]=1`	`; Wenn der Istwert der Y-Achse größer oder gleich 35.7 ist, dann wird M135 an PLC ausgegeben und gleichzeitig der Ausgang 1 gesetzt.`

Programm/Technologiezyklus

Als Aktion kann auch der Name eines Programms angegeben werden. In diesem Programm sind alle Aktionen zulässig, die einzeln in Synchronaktionen programmiert werden können. Diese Programme werden im Folgenden auch als Technologiezyklen bezeichnet.

Die Folge von Aktionen in einem Technologiezyklus wird sequentiell im Interpolationstakt abgearbeitet (siehe "Eigenschaften von Technologiezyklen [Seite 89]").

Anwendungsbeispiele:

- Einzelachsprogramme
- Zyklische Maschinen

Bearbeitungsvorgang

Die Sätze eines Teileprogramms werden in der Programmaufbereitung vorbereitet, abgespeichert und dann sequentiell in der Interpolationsebene (Hauptlauf) abgearbeitet. Der Zugriff auf Variablen erfolgt während der Aufbereitung. Bei Verwendung von Hauptlaufvariablen (z. B. Istwerten) wird die Satzaufbereitung unterbrochen, damit die aktuellen Echtzeitwerte bis zum Vorgängersatz bereitgestellt werden können.

Synchronaktionen werden in aufbereiteter Form mit dem aufbereiteten Satz in den Interpolator transportiert. Die verwendeten Hauptlaufvariablen werden im Interpolationstakt ausgewertet. Die Satzaufbereitung wird nicht unterbrochen.

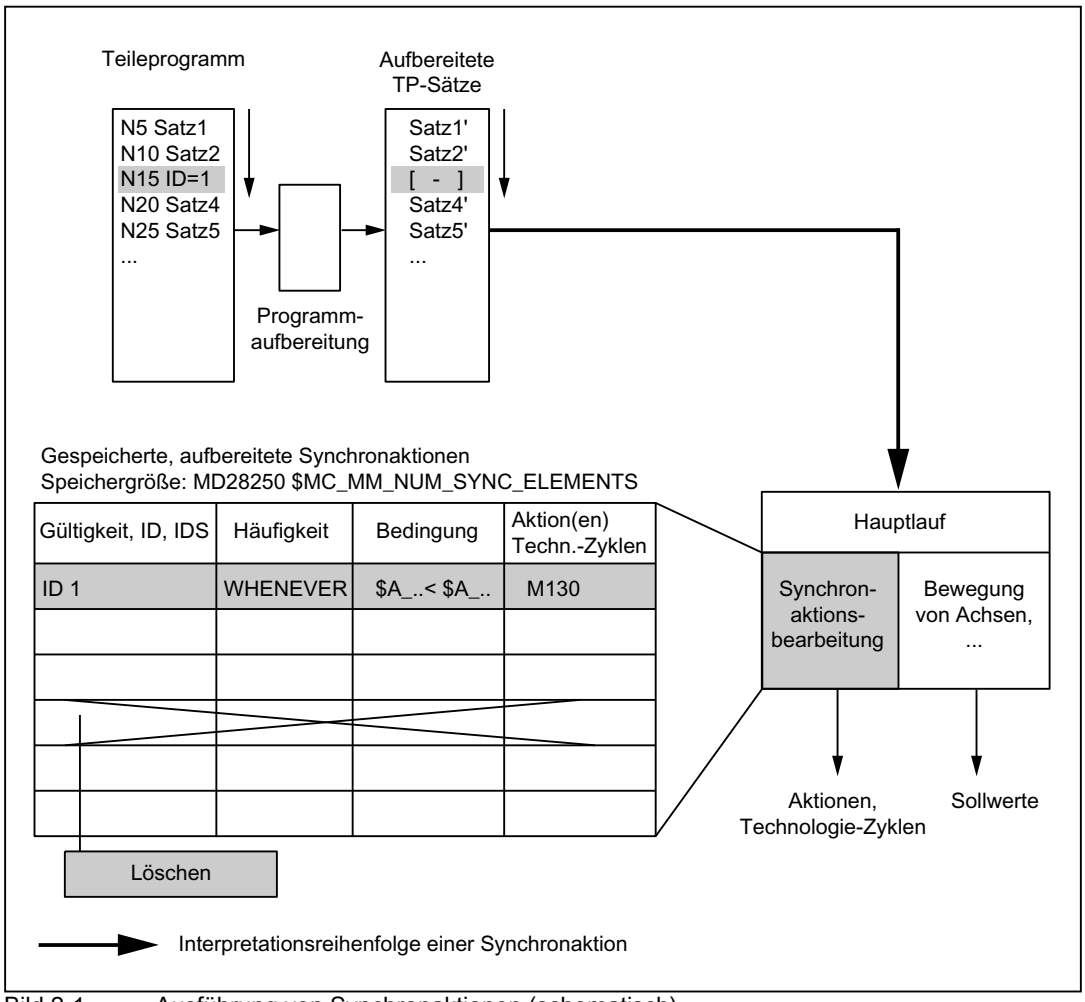


Bild 2-1 Ausführung von Synchronaktionen (schematisch)

Verhalten bei Nullsätzen

Teileprogrammsätze ohne Verfahrbewegung (Nullsätze) werden normalerweise vom Interpreter eliminiert. Wenn aber Synchronaktionen aktiv sind, wird dieser Nullsatz eingekettet und ausgeführt. Hierbei wird ein Genauhalt entsprechend aktiver Funktion (z. B. `G601`) ausgelöst. Damit soll die Synchronaktion die Möglichkeit bekommen, gegebenenfalls zu schalten.

Beispiele für Nullsätze:

Programmcode

```
N1000 G91 X0 Y0 Z0
...
```

Programmcode

```
N10 G90 G64 X100 Y100 Z100
N15 Z100
...
```

Hinweis

Sätze ohne Verfahrbewegung können auch durch Programmsprünge erzeugt werden.

Ausführungsbedingungen

Die Aktionen in Bewegungssynchronaktionen werden ausgeführt, wenn:

- die Synchronaktion existiert und nicht mit `CANCEL(ID)` abgewählt wurde.
- die Synchronaktion nicht gesperrt ist: kein `LOCK(ID)`

  (siehe Kapitel "Koordinierungen zwischen Synchronaktionen, Technologiezyklen, Teileprogramm und PLC").
- aufgrund des Häufigkeitsschlüsselworts eine Auswertung fällig ist.
- die Bedingung erfüllt ist.

Zeitintervall

Die Überprüfung, ob Aktionen in Synchronaktionen zu aktivieren sind, erfolgt im Interpolationstakt.

Aktion(en) werden synchron zur Bahnführung ausgeführt, wenn die links der Aktion(en) stehenden Voraussetzungen erfüllt sind.

Abarbeitungsreihenfolge

Innerhalb eines Interpolationstakts werden modal wirksame Synchronaktionsanweisungen in der Reihenfolge ihrer ID-Nummer bearbeitet (Satz mit ID-Nummer 1 vor Satz mit ID-Nummer 2 ...).

Nach den modal wirksamen Synchronaktionsanweisungen werden die satzweise wirksamen Synchronaktionsanweisungen in der Reihenfolge ihrer Programmierung bearbeitet.

2.3 Liste möglicher Aktionen

Mögliche Aktionen in Synchronaktionen sind:

- Ausgabe von M-, S- und H-Hilfsfunktionen an die PLC
- Durch Setzen (Schreiben) von Hauptlaufvariablen ist möglich:
  - Überlagerte Bewegung: `$AA_OFF` (Option)
  - Aufgeschaltete Werkzeuglängenkorrekturen: `$AA_TOFF` (Option)
  - Vorschubbeeinflussung: `$AC_OVR`, `$AA_OVR`

    Sperren einer programmierten Achsbewegung
  - Servo-Daten-Werte zuweisen: `$V...=`
  - Rechenvariable lesen oder schreiben: `$R[n]=`
  - Schreiben des SD-Wertes im Hauptlauf: `$$SD`
- Lesen des MD-Werts zum Interpretationszeitpunkt: `$MD...=`
- Verändern von SW-Nockenpositionen und -zeiten (Settingdaten)
- Veränderung von Koeffizienten und Grenzen aus `FCTDEF`
- Polynomauswertung: `SYNFCT`
- Online-Werkzeugkorrektur: `FTOC`
- Einlesesperre: `RDISABLE`
- Aufheben Vorlaufstopp: `STOPREOF`
- Restweglöschen: `DELDTG`
- Ermittlung von Kurventabellenwerten
- Axialer Vorschub aus Synchronaktionen
- Axiale Frames
- Achsen / Spindeln aus Synchronaktionen bewegen / positionieren
- Achstausch aus Synchronaktionen
- Spindelbewegungen aus Synchronaktionen
- Istwertsetzen aus Synchronaktionen (Preset)
- Kopplungen und Mitschleppen aktivieren / deaktivieren
- Koppelmodule der Generischen Kopplung aktivieren / deaktivieren
- Messen aus Synchronaktionen
- Setzen und Löschen von Wartemarken der Kanalsynchronisation
- Alarm setzen / Fehlerreaktionen
- Fahren auf Festanschlag: `FXS` (`FXST`, `FXSW`)
- Fahren mit begrenztem Moment: `FOC` (`FOCON` / `FOCOF`)

- Erweitertes Stillsetzen und Rückziehen

  Literatur:
  Funktionshandbuch Sonderfunktionen; Erweitertes Stillsetzen und Rückziehen (R3)
- Lesen und, wenn entsprechend gekennzeichnet, Schreiben von Systemvariablen

  (siehe Handbuch der Systemvariablen)

Hinweis

Unter dem Thema "Aktionen in Synchronaktionen" werden die Aktionen im Detail beschrieben.

2.4 Auswertungen und Berechnungen in Echtzeit

Echtzeitberechnungen

Die Berechnungen, die in Echtzeit durchgeführt werden, sind eine Untermenge der in der NC-Sprache möglichen Berechnungen. Sie sind beschränkt auf die Datentypen REAL, INT, CHAR und BOOL.

In Synchronaktionen sind implizite Typwandlungen zwischen REAL, INT und BOOL in beiden Richtungen möglich. Bei Wertzuweisungen und Parameterübergaben können Variablen unterschiedlicher Datentypen zugewiesen oder übergeben werden.

Echtzeit-Ausdruck

Der Begriff Echtzeit-Ausdruck bezeichnet im Folgenden alle im Interpolationstakt möglichen Berechnungen. Der Echtzeit-Ausdruck wird verwendet in der Bedingung und in der Zuweisung an NC-Adressen und Variablen.

Hauptlaufvariablen

Alle Hauptlaufvariablen werden im Interpolationstakt ausgewertet (gelesen) und können als Bestandteil einer Aktion geschrieben werden.

Die in Synchronaktionen verfügbaren Systemvariablen im Hauptlauf sind in den Systemvariablen-Tabellen (siehe Handbuch der Systemvariablen) an den markierten Feldern unter "HL-Sync" zu erkennen.

Kennzeichnung von Hauptlaufvariablen

Hauptlaufvariablen sind alle Variablen, die mit folgender Kennzeichnung beginnen:

$A...	aktuelle Hauptlaufvariable
$V...	Servo-Werte
$R...	R-Parameter

NC-Maschinen- und Settingdaten werden im Hauptlauf mit folgender Kennzeichnung in Synchronaktionen interpretiert:

- $$M...
- $$S...

Hinweis

MD- / SD-Werte zum Hauptlaufzeitpunkt werden für einen Online-Zugriff mit $$S... oder $$M... adressiert und zum Hauptlaufzeitpunkt ausgewertet, während MD- / SD-Werte mit einem $-Zeichen zum Interpretationszeitpunkt der Synchronaktion gelesen werden.

Settingdaten und Maschinendaten aus der Synchronaktion werden mit den $-Zeichen adressiert und zum Vorlaufzeitpunkt ausgewertet.

Datentyp-Konvertierung

Durch eine interne Datentyp-Konvertierung können bei Wertzuweisungen und Parameterübergaben Variablen unterschiedlicher Datentypen zugewiesen oder übergeben werden.

Möglich sind folgende Datentyp-Konvertierungen:

- INT nach REAL
- REAL nach INT
- REAL nach BOOL
- INT nach BOOL
- BOOL nach REAL oder INT

Beispiele:

1. Konvertierung von INT nach REAL

Programmcode

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

2. Konvertierung von REAL nach INT

Programmcode

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

3. Konvertierung von INT nach BOOL

Programmcode

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

4. Konvertierung von REAL nach BOOL

Programmcode

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

5. Konvertierung von BOOL nach INT

Programmcode

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

6. Konvertierung von BOOL nach REAL

Programmcode
R401=100.542 WHEN $A_IN[3]==TRUE DO $R10=$A_OUT[3]

Hinweis

Datentyp-Konvertierung von REAL nach INT

Bei Konvertierungen von REAL nach INT wird bei gebrochenem Wert ≧ 0.5 aufgerundet, ansonsten wird abgerundet (vgl. Funktion ROUND). Liegen Werte außerhalb des Wertebereichs für INT-Variablen, dann wird ein Alarm ausgegeben und die Konvertierung wird nicht durchgeführt.

Verknüpfung von Hauptlaufvariablen

Hauptlaufvariablen vom Typ REAL oder INT können durch die folgenden Grundrechenarten miteinander verknüpft werden:

- Addition
- Subtraktion
- Multiplikation
- Division
- Integer-Division
- Modulo-Division

Hinweis

Es können nur Variablen gleichen Typs verknüpft werden.

Ausdrücke aus Grundrechenarten

Ausdrücke aus Grundrechenarten können geklammert und geschachtelt werden (siehe Thema "Priorität von Operatoren").

Vergleichsoperatoren

Möglich sind die Vergleichsoperatoren:

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

Boolesche Operatoren

Möglich sind die booleschen Operatoren:

NOT	NICHT
AND	UND
OR	ODER
XOR	exklusives ODER

Bitweise logische Operatoren

Möglich sind die bitweise logischen Operatoren:

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

Priorität von Operatoren

Um bei mehrgliedrigen Ausdrücken das gewünschte Verknüpfungsergebnis zu erhalten, sind die Prioritäten der Operatoren bei Berechnungen und Bedingungen wie folgt zu berücksichtigen:

1.	Verneinung, bitweise Verneinung	NOT, B_NOT
2.	Multiplikation, Division	*, /, DIV, MOD
3.	Addition, Subtraktion	+, -
4.	bitweise UND	B_AND
5.	bitweise exklusives ODER	B_XOR
6.	bitweise ODER	B_OR
7.	UND	AND
8.	exklusives ODER	XOR
9.	ODER	OR
10.	nicht vergeben	
11.	Vergleichsoperatoren	
	gleich	==
	ungleich	<>
	größer	<
	kleiner	>
	größer oder gleich	>=
	kleiner oder gleich	<=

Ggf. sind runde Klammern zu verwenden. Das Verknüpfungsergebnis von Bedingungen muss vom Typ BOOL sein.

Beispiel einer mehrgliedrigen Bedingung:

Programmcode

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

Funktionen

Von einer Hauptlaufvariablen des Typs REAL kann auch der Funktionswert sin, cos etc. gebildet werden.

Möglich sind folgende Funktionen:

SIN, COS, ABS, ASIN, ACOS, TAN, ATAN2, TRUNC, ROUND, LN, EXP, ATAN, POT, SQRT, CTAB, CTABINV

Beispiel:

Programmcode

```
... DO $AC_PARAM[3]=COS($AA_IM[X])
```

Die Bedeutungen der angegebenen Funktionen sind erklärt in:
Literatur:
Programmierhandbuch Grundlagen
Programmierhandbuch Arbeitsvorbereitung

Indizierung

Der Index einer Hauptlauffeldvariablen kann wiederum eine Hauptlaufvariable sein.

Beispiel:

Programmcode

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

Der Index $AC_MARKER[1] wird jeweils im Interpolationstakt aktuell ausgewertet.

Einschränkungen:

- Die Schachtelung der Indizierung mit Hauptlaufvariablen ist nicht erlaubt.
- Von einer Variablen, die nicht in Echtzeit gebildet wird, kann kein Echtzeit-Index gebildet werden. Folgende Programmierung liefert Fehler:
  $AC_PARAM[1]=$P_EP[$AC_MARKER[0]

2.5 Spezielle Hauptlaufvariablen für Synchronaktionen

Liste aller Systemvariablen

Eine vollständige Liste aller ansprechbaren Systemvariablen befindet sich in:
Literatur:
Handbuch der Systemvariablen

Systemvariablen in Synchronaktionen

Die in Synchronaktionen ansprechbaren Systemvariablen sind in den Systemvariablen-Tabellen an der Markierung im Feld "SA" zu erkennen.

Wenn in der Systemvariablen-Tabelle zusätzlich das Feld "HL-Sync" angekreuzt ist, dann wird eine im Hauptlauf aktualisierte Systemvariable auch im Hauptlauf synchronisiert.

Im Folgenden werden die Eigenschaften einiger spezieller Hauptlaufvariablen vorgestellt:

- Merker- / Zähler-Variablen
  - Kanalspezifische Merker
- Zeiten (Timer)
- Synchronaktionsparameter
- R-Parameter
- Maschinen- und Setting-Daten
- FIFO-Variablen (Durchlaufspeicher)

2.5.1 Merker-/Zähler-Variablen

Kanalspezifischer Marker $AC_MARKER[n]

Die Feld-Variable $AC_MARKER[n] dient als Merker oder Zähler und kann in Synchronaktionen gelesen und beschrieben werden.

Datentyp:	INT
n:	Feldindex des Markers (0 - n)

Anzahl

Die Anzahl der Marker $AC_MARKER[n] pro Kanal wird eingestellt über das Maschinendatum:

MD28256 $MC_MM_NUM_AC_MARKER

Höchstwert: 20000

Die Marker sind unter gleichem Namen einmal pro Kanal vorhanden.

Speicherort

Der Speicherort für $AC_MARKER[n] kann festgelegt werden mit dem Maschinendatum:

MD28257 $MC_MM_BUFFERED_AC_MARKER

Wert	Speicherort
0	Dynamischer NC-Speicher (Standardeinstellung)
1	Statischer NC-Speicher

Ein Element benötigt 4 Bytes Speicherplatz. Es muss darauf geachtet werden, dass im gewählten Speicher der erforderliche Speicherplatz verfügbar ist.

Hinweis

Im statischen NC-Speicher abgelegte Marker können in die Datensicherung einbezogen werden.

Verhalten bei POWER ON / NC-Reset / TP-Ende

Bei POWER ON, NC-Reset und Teileprogrammende werden die Marker auf "0" gesetzt. Damit sind für jeden Programmdurchlauf die gleichen Startbedingungen gegeben.

2.5.2 Zeiten (Timer)

$AC_TIMER[n]

Die Systemvariable $AC_TIMER[n] ermöglicht das Starten von Aktionen nach definierten Wartezeiten.

Datentyp:	REAL
n:	Nummer der Timer-Variable
Einheit:	Sekunde

Anzahl

Die Anzahl der verfügbaren Timer-Variablen wird eingestellt über das Maschinendatum:

MD28258 $MC_MM_NUM_AC_TIMER

Timer setzen

Das Hochzählen einer Timer-Variablen wird gestartet durch Wertzuweisung:
`$AC_TIMER[<n>]=<Wert>`

mit:	<n>	Nummer der Zeitvariablen
	<Wert>	Startwert (i. d. R. "0")

Timer anhalten

Das Hochzählen einer Timer-Variablen wird gestoppt durch Zuweisung eines negativen Werts, z. B.:
`$AC_TIMER[n]=-1`

Timer lesen

Der aktuelle Zeitwert kann bei laufender oder gestoppter Timer-Variablen gelesen werden. Nach dem Stoppen der Timer-Variablen durch Zuweisung von "-1" bleibt der zuletzt aktuelle Zeitwert stehen und kann weiterhin gelesen werden.

Beispiel

Ausgabe eines Istwerts über Analogausgang 500 ms nach Erkennen eines digitalen Eingangs:

Programmcode	**Kommentar**
`WHEN $A_IN[1]==1 DO $AC_TIMER[1]=0`	`; Timer rücksetzen und starten.`
`WHEN $AC_TIMER[1]>=0.5 DO $A_OUTA[3]=$AA_IM[X] $AC_TIMER[1]=-1`	

2.5.3 Synchronaktionsparameter

$AC_PARAM[n]

Die Variablen $AC_PARAM[n] dienen für Berechnungen und als Zwischenspeicher und können in Synchronaktionen gelesen und beschrieben werden.

Datentyp:	REAL
n:	Nummer des Parameters 0 - n

Anzahl

Die Anzahl der verfügbaren AC-Parameter-Variablen pro Kanal wird eingestellt über das Maschinendatum:

MD28254 $MC_MM_NUM_AC_PARAM

Höchstwert: 20000

Die Parameter sind unter gleichem Namen einmal pro Kanal vorhanden.

Speicherort

Der Speicherort für $AC_PARAM[n] kann festgelegt werden mit dem Maschinendatum:

MD28255 $MC_MM_BUFFERED_AC_PARAM

Wert	Speicherort
0	Dynamischer NC-Speicher (Standardeinstellung)
1	Statischer NC-Speicher

Ein Element benötigt 8 Bytes Speicherplatz. Es muss darauf geachtet werden, dass im gewählten Speicher der erforderliche Speicherplatz verfügbar ist.

Hinweis

Im statischen NC-Speicher abgelegte Synchronaktionsparameter können in die Datensicherung einbezogen werden.

Verhalten bei POWER ON / NC-Reset / TP-Ende

Bei POWER ON, NC-Reset und Teileprogrammende werden die Parameter auf "0" gesetzt. Damit sind für jeden Programmdurchlauf die gleichen Startbedingungen gegeben.

2.5.4 R-Parameter

Verwendung in Synchronaktionen

Mit dem Einleitungszeichen $ vor dem R-Parameter können Rechenparameter auch in Synchronaktionen verwendet werden:

- R[n] oder Rn: Berechnungen im Teileprogramm
- $R[n] oder $Rn: Berechnungen in Synchronaktionen

Datentyp:	REAL
n:	Nummer der Feld-Variablen

Speicherort

R-Parameter werden im statischen NC-Speicher persistent gespeichert und behalten deshalb über POWER ON, NC-Reset und Teileprogrammende hinweg ihre Werte.

Beispiele

Beispiel 1: Messwert im Rechenparameter übernehmen

Programmcode	**Kommentar**
`WHEN $AC_MEA==1 DO $R10=$AA_MM[Y]`	`; Wenn gültige Messung vorliegt, Messwert in R-Parameter übernehmen.`

Beispiel 2: Auswertung des R-Parameters

Programmcode	Kommentar
WHEN $A_IN[1]==1 DO $R10=$AA_IM[Y]	
G1 X100 F150	
STOPRE	
IF R10 > 50 ...	; Auswertung des R-Parameters

2.5.5 Maschinen- und Settingdaten

MD und SD lesen und schreiben

Das Lesen und Schreiben von Maschinen- und Settingdaten ist auch aus Synchronaktionen möglich. Der Zugriff muss unterschieden werden nach:

- MD und SD, die während der Bearbeitung unverändert bleiben
- MD und SD, die sich während der Bearbeitung verändern

Zum Vorlaufzeitpunkt lesen

Unveränderliche Maschinendaten und Settingdaten werden aus der Synchronaktion adressiert wie in normalen Teileprogramm-Befehlen. Sie werden mit einem $–Zeichen eingeleitet.

Beispiel:

Programmcode
ID=2 WHENEVER $AA_IM[z] < $SA_OSCILL_REVERSE_POS2[Z]-6 DO $AA_OVR[X]=0

Hier wird der während der Bearbeitung als unveränderlich angenommene Umkehrbereich 2 für Pendeln angesprochen.

Zum Hauptlaufzeitpunkt lesen

Während der Bearbeitung sich ändernde Maschinendaten und Settingdaten werden aus der Synchronaktion mit $$-Zeichen eingeleitet.

Beispiel:

Programmcode
ID=1 WHENEVER $AA_IM[z]< $$SA_OSCILL_REVERSE_POS2[Z]-6 DO $AA_OVR[X]=0

In diesem Zusammenhang wird davon ausgegangen, dass die Umkehrposition durch Bedienung jederzeit verändert werden könnte.

Zum Hauptlaufzeitpunkt schreiben

Voraussetzung:

Das aktuell eingestellte Zugriffsrecht muss den Schreibzugriff zulassen. Es ist nur sinnvoll, MD und SD aus der Synchronaktion zu schreiben, wenn dies sofort wirksam wird. Die

Wirksamkeit für alle Maschinendaten und Settingdaten wird angegeben in:
Literatur:
Listen (Buch 1)

Adressierung:

Zu schreibende Maschinendaten und Settingdaten werden mit $$ eingeleitet.

Beispiel:

Programmcode	**Kommentar**
`ID=1 WHEN $AA_IW[X]>10 DO $$SN_SW_CAM_PLUS_POS_TAB_1[0]=20 $$SN_SW_CAM_MINUS_POS_TAB_1[0]=30`	`; Veränderung der Schaltpositionen von SW-Nocken von 20 nach 30.`

2.5.6 FIFO-Variablen (Durchlaufspeicher)

Anwendung

Zur Abspeicherung zusammengehöriger Datenfolgen stehen bis zu 10 `FIFO`-Variablen zur Verfügung: **$AC_FIFO1[n]** bis **$AC_FIFO10[n]**.

Struktur

Die Speicherstruktur einer `FIFO`-Variablen zeigt das Bild: Beispiel `FIFO`-Variable.

Anzahl

Die Anzahl der verfügbaren AC–FIFO–Variablen wird festgelegt über das Maschinendatum

MD28260 $MC_NUM_AC_FIFO

Größe

Die Anzahl der in eine FIFO–Variable ablegbaren Werte wird definiert durch das Maschinendatum

MD28264 $MC_LEN_AC_FIFO

Alle FIFO–Variablen haben gleiche Länge.

Datentyp

Werte in der `FIFO`-Variablen haben den Datentyp `REAL`.

Bedeutung Index

Index n:

Die Indizes 0 bis 5 haben Sonderbedeutungen:

n=0: Beim Schreiben mit Index 0 wird ein neuer Wert in den FIFO abgelegt. Beim Lesen mit Index 0 wird das älteste Element gelesen und aus dem FIFO entfernt.

n=1: Zugriff auf das älteste gespeicherte Element

n=2: Zugriff auf das jüngste gespeicherte Element

n=3: Summe aller FIFO–Elemente

Das MD28266 $MC_MODE_AC_FIFO bestimmt den Modus

der Summenbildung:

Bit 0 = 1 Summe bei jedem Einschreiben

aktualisieren

Bit 0 = 0 keine Summenbildung möglich.n=4: Anzahl der im FIFO verfügbaren Elemente. Auf jedes Element des FIFO kann lesend und schreibend zugegriffen werden.

Das Rücksetzen der FIFO-Variablen erfolgt durch Rücksetzen der Element-Anzahl z. B. für die erste FIFO-Variable: **$AC_FIFO1[4]=0**

n=5: aktueller Schreibindex relativ zum FIFO-Anfang

n= 6 bis 6+nmax: Zugriff auf n-tes FIFO-Element:

Hinweis

Der FIFO-Zugriff ist eine spezielle Form des R-Parameter-Zugriffs: (s. unten)

Die FIFO-Werte werden im R-Parameterbereich hinterlegt.

Die FIFO-Werte liegen im statischen Speicher. Sie bleiben über Programmende/Reset und Power On hinweg erhalten.

Bei der Archivierung von R-Parametern werden die FIFO-Werte mit gesichert.

Das Maschinendatum

MD28262 $MC_START_AC_FIFO

legt die Nummer des R–Parameters fest, ab der FIFO–Variablen im R–Parameter–Bereich liegen.

Die aktuelle Anzahl für R–Parameter eines Kanals wird durch das Maschinendatum

MD28050 $MC_MM_NUM_R_PARAM

definiert. Die zwei folgenden Bilder zeigen Teilelängen von Teilen auf einem Band, die in einer FIFO-Variablen abgelegt wurden schematisch.

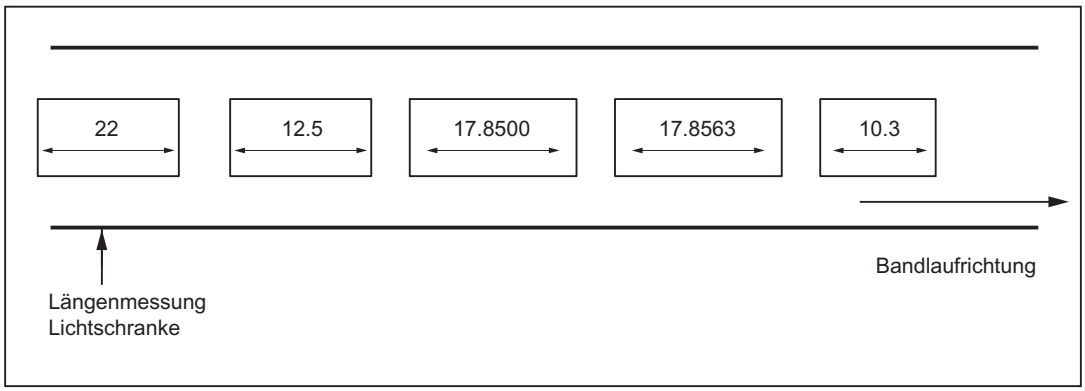


Bild 2-2 Produktlängen einer Teilefolge auf Band

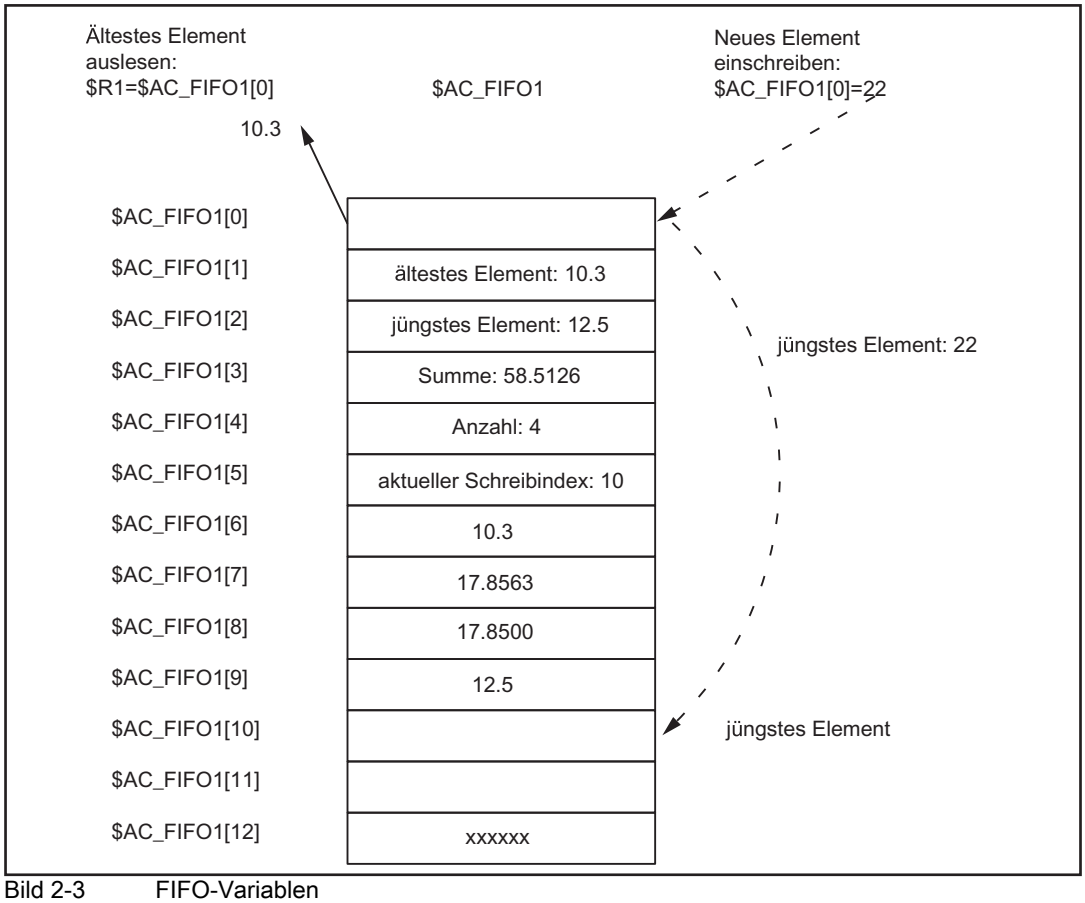


Bild 2-3 FIFO-Variablen

2.5.7 SRAM gespeicherte Systemvariablen

RESET-Verhalten

Im SRAM gespeicherte Systemvariablen $AC_MARKER und $AC_PARAM behalten über `RESET` und Power On hinweg ihre aktuellen Werte.

Hinweis

In Teileprogrammen und Synchronaktionen, die mit SRAM-gespeicherten Systemvariablen arbeiten, muss beachtet werden, dass nach RESET keine Initialisierung der Variablen mit 0 stattfindet. Das erfordert ggf. Anpassungen, wenn zuvor mit DRAM gespeicherten Systemvariablen gearbeitet worden ist.

Datensicherung

Im SRAM gespeicherte Systemvariablen $AC_MARKER und $AC_PARAM können in die Datensicherung einbezogen werden.

Pro Kanal existieren die Sicherungsbausteine:

_N_CHi_ACM	für $AC_MARKER Werte und
_N_CHi_ACP	für $AC_PARAM Werte.

i steht für die jeweilige Kanalnummer.

Reihenfolge

Die gesicherten Bausteine werden im File der Gesamtsicherung _N_INITIAL_INI nach R-Parametern eingetragen.

Literatur:
/IAD/ Inbetriebnahmehandbuch; NCU 840D

2.5.8 Bestimmung des Bahntangentenwinkels in Synchronaktionen

$AC_TANEB

Die in Synchronaktionen lesbare Systemvariable $AC_TANEB (**T**angent **AN**gle at **E**nd of **B**lock) ermittelt den Winkel zwischen der Bahntangente im Endpunkt des aktuellen Satzes und der Bahntangente im Startpunkt des programmierten Folgesatzes.

Der Tangentenwinkel wird stets positiv im Bereich 0.0° bis 180.0° ausgegeben. Existiert kein Nachfolgesatz im Hautlauf, so wird der Winkel -180.0° ausgegeben.

Die Systemvariable $AC_TANEB sollte nicht für Sätze, die vom System erzeugt werden (Zwischensätze) gelesen werden. Zur Unterscheidung, ob es sich um einen programmierten Satz (Hauptsatz) handelt, dient die Systemvariable $AC_BLOCKTYPE.

Programmierbeispiel:

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

2.5.9 Bestimmung des aktuellen Override

Aktueller Override

Der aktuelle Override (NC-Anteil) kann mit den folgenden Systemvariablen in Synchronaktionen gelesen und geschrieben werden:

$AA_OVR	Axial-Override
$AC_OVR	Bahnoverride

PLC-Override

Der von der PLC vorgegebene Override wird für Synchronaktionen in den folgenden Systemvariablen zum Lesen bereitgestellt:

$AA_PLC_OVR	Axial-Override
$AC_PLC_OVR	Bahnoverride

Resultierender Override

Der resultierende Override wird für Synchronaktionen in den folgende Systemvariablen zum Lesen bereitgestellt:

$AA_TOTAL_OVR	Axial Override
$AC_TOTAL_OVR	Bahnoverride

Resultierender Override errechnet sich als:

$AA_OVR	*	$AA_PLC_OVR	bzw.
$AC_OVR	*	$AC_PLC_OVR	

2.5.10 Auslastungsauswertung über Zeitbedarf bei Synchronaktionen

Bedeutung

In einem Interpolationstakt müssen sowohl Synchronaktionen interpretiert als auch Bewegungen usw. von der NC berechnet werden. Mit den im Folgenden vorgestellten Systemvariablen können sich Synchronaktionen über die aktuellen Zeitanteile der Synchronaktionen am Interpolationstakt und über die Rechenzeit der Lageregler informieren.

Die Variablen haben nur gültige Werte, wenn:

MD11510 $MN_IPO_MAX_LOAD > 0

Andernfalls geben die Variablen immer die Nettorechenzeit an, bei der die durch HMI erzeugten Unterbrechungen nicht mehr berücksichtigt werden.

Die Nettorechenzeit ergibt sich aus:

- Synchronaktionszeit
- Lageregelungszeit
- und der restlichen IPO–Rechenzeit ohne HMI bedingte Unterbrechungen

Die Variablen enthalten immer die Werte des vorhergehenden IPO-Takts.

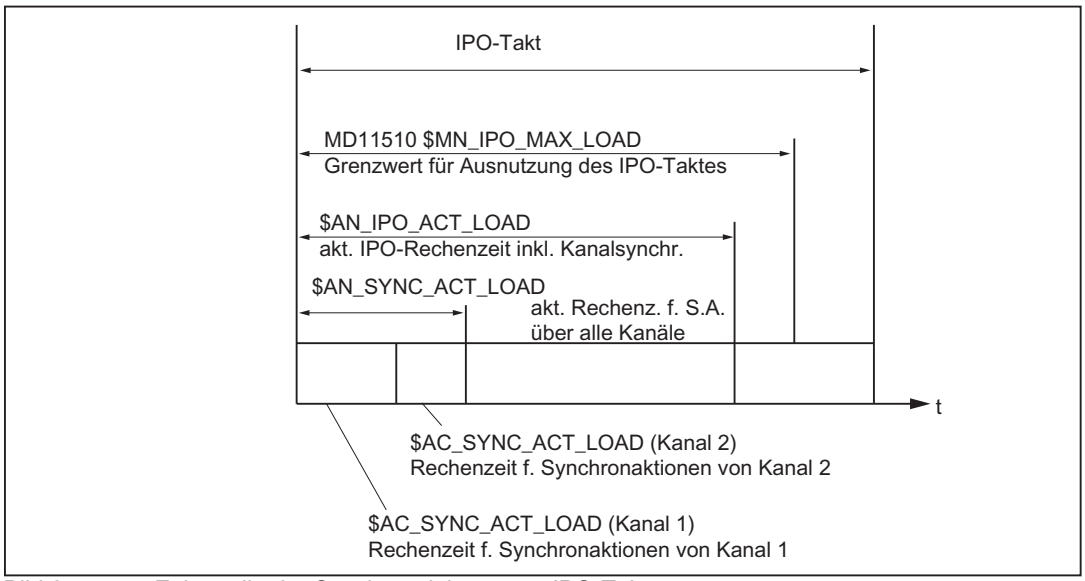


Bild 2-4 Zeitanteile der Synchronaktionen am IPO-Takt

Systemvariable	Bedeutung
$AN_IPO_ACT_LOAD	aktuelle IPO-Rechenzeit (inkl. Synchronaktionen aller Kanäle)
$AN_IPO_MAX_LOAD	längste IPO-Rechenzeit (inkl. Synchronaktionen aller Kanäle)
$AN_IPO_MIN_LOAD	kürzeste IPO-Rechenzeit (inkl. Synchronaktionen aller Kanäle)
$AN_IPO_LOAD_PERCENT	aktuelle IPO-Rechenzeit im Verhältnis zum IPO-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 ges. Synchronaktionen an der gesamten IPO-Rechenzeit (über alle Kanäle)
$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
$AN_SERVO_ACT_LOAD	aktuelle Rechenzeit des Lagereglers
$AN_SERVO_MAX_LOAD	längste Rechenzeit des Lagereglers
$AN_SERVO_MIN_LOAD	kürzeste Rechenzeit des Lagereglers

Überlastmitteilung

Über das MD11510 $MN_IPO_MAX_LOAD wird eingestellt, ab welcher IPO-Netto-Rechenzeit (in % vom IPO-Takt) die Systemvariable $AN_IPO_LOAD_LIMIT auf TRUE gesetzt werden soll.

Unterschreitet die aktuelle Last diese Grenze wieder, so wird die Variable wieder auf FALSE gesetzt.

Ist das MD = 0, so ist die gesamte Diagnosefunktion deaktiviert. Durch die Auswertung von $AN_IPO_LOAD_LIMIT kann der Anwender eine eigene Strategie festlegen, um Ebenenüberlauf zu vermeiden.

Beschreibbare Systemvariablen

Diese Systemvariablen von den oben angegebenen sind aus Synchronaktionen heraus **beschreibbar**:

Systemvariable	Bedeutung
$AN_SERVO_MAX_LOAD	längste Rechenzeit des Lagereglers
$AN_SERVO_MIN_LOAD	kürzeste Rechenzeit des Lagereglers
$AN_IPO_MAX_LOAD	längste IPO-Rechenzeit (inkl. Synchronaktionen aller Kanäle)
$AN_IPO_MIN_LOAD	kürzeste IPO-Rechenzeit (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

Diese Variablen werden bei jedem Schreibzugriff auf die aktuelle Last zurückgesetzt, unabhängig vom geschriebenen Wert.

Programmierbeispiel

```
Programmcode                          Kommentar
$MN_IPO_MAX_LOAD=80                   ; MD: Ausnutzungsgrenze für IPO-Takt
                                      ; Sobald $AN_IPO_LOAD_PERCENT > 80%, wird
                                      ; $AN_IPO_LOAD_LIMIT auf TRUE gesetzt.

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

N10 IDS=1 WHENEVER $AN_IPO_LOAD_LIMIT==TRUE DO M4711 SETAL(63111)
N20 IDS=2 WHENEVER $AN_SYNC_TO_IPO > 30 DO SETAL(63222)
N30 G0 X0 Y0 Z0
...
N999 M30
```

Die erste Synchronaktion erzeugt eine Hilfsfunktionsausgabe und einen Alarm, wenn die gesamte Ausnutzungsgrenze überschritten wird.

Die zweite Synchronaktion erzeugt einen Alarm, wenn der Anteil der Synchronaktionen an der IPO- Rechenzeit (über alle Kanäle) 30% überschreitet.

2.5.11 Liste der für Synchronaktionen bedeutsamen Systemvariablen

Hinweis

Die an dieser Stelle bisher aufgelisteten Systemvariablen, welche von Synchronaktitionen

angesprochen werden können, sind ab den SW–Stand 7.1 zu finden in der eigenständigen Drückschrift:

/PGA1/ Listenhandbuch Systemvariablen.

Es sind alle Systemvariablen mit der entsprechenden Kennzeichnung X von Synchronaktionen (SA) nutzbar (read/write). Weitere Erläuterungen zu den Eigenschaften der Systemvariablen im Hauptlauf siehe Kapitel 2.3 "Spezielle Hauplaufvariablen für Synchronaktionen".

2.6 Aktionen in Synchronaktionen

Aktionen

Jede Synchronaktion enthält nach dem Aktionskennwort **DO ...** eine oder mehrere (max. 16) Aktionen oder einen Technologiezyklus, die bei erfüllter Bedingung ausgeführt werden. (Als Oberbegriff wird im weiteren **Aktionen** verwendet.).

Mehrere Aktionen

Mehrere Aktionen einer Synchronaktion werden bei erfüllter Bedingung im gleichen Interpolationstakt aktiviert.

Liste möglicher Aktionen

Im Aktionsteil von Synchronaktionen sind die folgenden Aktionen möglich:

Tabelle 2-1 Aktionen in Synchronaktionen

... DO ...	Bedeutung	Verweis
Mxx Sxx Hxx	Hilfsfunktionsausgabe an PLC	2.4.1
SETAL(nr)	Alarm setzen, Reaktion auf Fehler	
$V... = ...	Variable zuweisen (Servo–Werte)	2.4.2
$A... = ...	Variable zuweisen (Hauptlaufvariable)	
$AA_OFF =	– Überlagerte Bewegung	
$AC_OVR =	– Geschwindigkeitsbeeinflussung:	
$AA_OVR =	Bahngeschwindigkeit	
$AC_VC =	Achsgeschwindigkeit	
$AA_VC =	add. Bahnvorschubkorrektur	2.4.3
$$SN_SW_CAM_ ...	add. Korrekturwert der Achse	
$AC_FCT...	Verändern von SW–Nockenpositionen	2.4.4
$AA_TOFF =	(Settingdaten) und alle anderen SD	
	Überschreiben von FCTDEF–Parametern	2.4.8
	– aufgeschaltete WZL–Korrekturen	
	Synchronaktionsprozeduren:	
RDISABLE	Einlesesperre aktivieren	2.4.9
STOPREOF	Vorlaufstop beenden	2.4.10
DELDTG	Restweg löschen	2.4.11
FTOC	Online-Werkzeugkorrektur	2.4.7
SYNFCT	Polynomauswertung	2.4.5
ZYKL_T1 (z. B.)	Aufruf von Technologiezyklen	2.5

Tabelle 2-1 Aktionen in Synchronaktionen

... DO ...	Bedeutung	Verweis
	Steuerung von Positionierachsen:	
$AA_OVR[x]= 0	Sperren einer Achsbewegung	2.4.12
ACHSE_X (z. B.)	Aufruf eines Achsprogramms	2.4.13
POS[u]= ...	Positionieren	2.4.13
FA[u]= ...	Achsvorschub festlegen	2.4.14
	Kommando-Achsen kontinuierlich bewegen:	2.4.15
MOV[u]= >0	- vorwärts	"
MOV[u] = <0	- rückwärts	"
MOV[u] = =0	- anhalten	"
AXTOCHAN	Achstausch aus einer Synchronaktion	2.4.16
GET(Achse)	Achse für Achstausch anfordern	"
RELEASE(Achse)	Achse zum Achstausch freigeben	"
	Spindeln:	
SPOS	Positionieren	
M3, M4, M5, S =	Drehrichtung, Halt, Drehzahl	2.4.18
$AA_OVR[S1]= 0	Sperren der Spindelbewegung	
PRESETON(,)	Istwertsetzen	2.4.19
	Mitschleppen und Kopplungen	2.4.20
	Kopplungen aktivieren/deaktivieren:	
LEADON	Folgeachse an Leitachse ankoppeln	
LEADOF	Kopplung aufheben	2.4.20
TRAILON	Asynchr. Mitschleppen ein	
TRAILOF	Asynchr. Mitschleppen aus	
MEAWA	Messen ohne Restweglöschen	2.4.21
MEAC	zyklisches Messen	
	Kanalsynchronisation:	2.4.22
SETM	Setzen einer Wartemarke	
CLEARM	Löschen einer Wartemarke	
	Koordinierung zw. Synchronaktionen:	2.5.1
LOCK	- Synchronaktion/Technologiezyklus sperren	
UNLOCK	- Synchronaktion/Technologiezyklus freigeben	
RESET	- Technologiezyklus rücksetzen	

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

Literatur

Details zur Hilfsfunktionsausgabe im Allgemeinen finden Sie in:

Funktionshandbuch Grundfunktionen; Hilfsfunktionsausgabe an PLC (H2)

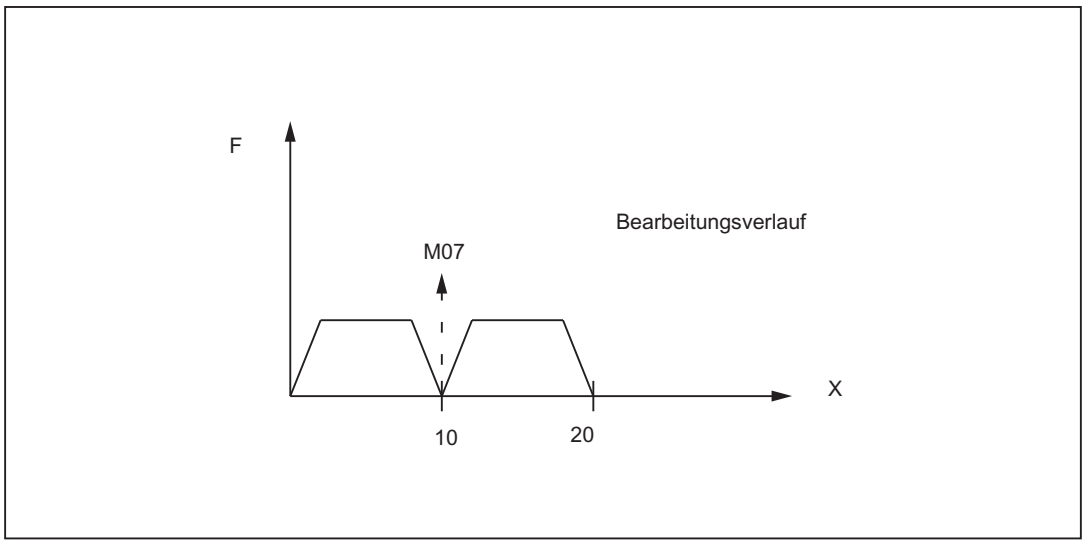
Hilfsfunktionsausgabe aus Synchronaktionen

Der Vorteil der Hilfsfunktionsausgabe aus Synchronaktionen wird am folgenden Beispiel deutlich:

Beispiel: Kühlmittel an bestimmter Position einschalten

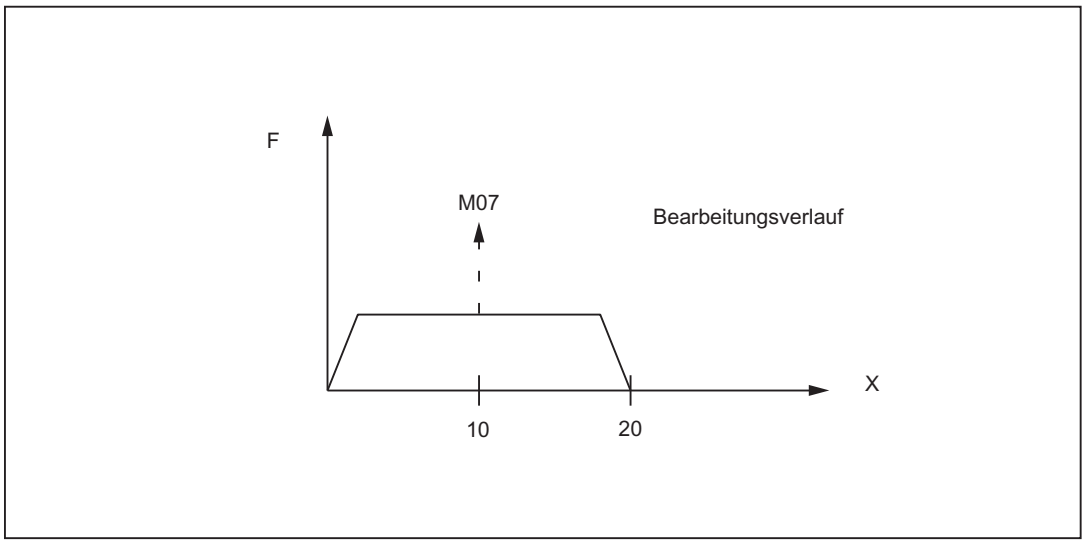
1. Lösung ohne Synchronaktion: 3 Sätze

Programmcode
N10 G1 X10 F150
N20 M07
N30 X20



2. Lösung mit Synchronaktion: 1 Satz

Programmcode
N10 WHEN $AA_IM[X] >= 10 DO M07
N20 G1 X20 F150



Ausgabezeitpunkt

Als Synchronaktionen können M-, S- oder H-Hilfsfunktionen an die PLC ausgegeben werden. Die Ausgabe erfolgt **sofort** (wie ein Interrupt an PLC) im Interpolationstakt, wenn die Bedingung erfüllt ist.

Ausgabezeitpunkte, die ggf. in einem der folgenden Maschinendaten definiert wurden, werden 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 an Hilfsfunktionsausgaben

Es können gleichzeitig (d. h. in einem OB40-Zyklus der PLC) maximal 10 Hilfsfunktionen ausgegeben werden. Die Summe der Hilfsfunktionsausgaben aus Teileprogrammen und Synchronaktionen darf zu keinem Zeitpunkt mehr als 10 pro Kanal betragen.

Höchste Anzahl an Hilfsfunktionen je Synchronaktions-Satz oder Technologiezyklus-Satz:

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

Programmierung

Hilfsfunktionen dürfen nur mit den Häufigkeitsschlüsseln `WHEN` oder `EVERY` in Synchronaktionen programmiert werden.

Beispiel:

Programmcode	Kommentar
`WHEN $AA_IM[X] > 50 DO H15 S3000 M03`	`; Wenn Istwert der X-Achse größer 50 wird, H15 ausgeben, neue Spindeldrehzahl, neue Drehrichtung einstellen.`

Hinweis

Vordefinierte M-Funktionen können nicht über Synchronaktionen programmiert werden. Sie werden mit Alarm abgelehnt.

Erlaubt sind jedoch die Spindel-M-Funktionen `M3`, `M4`, `M5`, `M40`, `M41`, `M42`, `M43`, `M44`, `M45`, `M70` und `M17` als Ende für einen Technologiezyklus.

Quittierung

Technologiezyklensätze (siehe Kapitel "Technologiezyklen") mit Hilfsfunktionsausgaben sind erst dann abgearbeitet, wenn die Quittung aller Hilfsfunktionen des Satzes von PLC erfolgt ist. Die Satzweiterschaltung im Technologiezyklus erfolgt erst dann, wenn alle darin enthaltenen Hilfsfunktionen von PLC quittiert sind.

Das Quittierungsverhalten wurde auf folgende Varianten erweitert:

- Hilfsfunktionsausgabe ohne Satzwechselverzögerung

  Schnelle Hilfsfunktionen (QUICK) vorab, als paralleler Prozess in der PLC, danach Hilfsfunktionsausgabe mit Quittungserwartung.

Der Datentyp für H-Hilfsfunktionen kann zwischen INT und REAL durch den Anwender gewählt werden. Das PLC-Anwenderprogramm muss entsprechend der Festlegung die übergebenen Werte interpretieren. Der INT-Wertebereich für H-Hilfsfunktionen wurde vergrößert auf: 2147483648 bis 2147483647.

2.6.2 Setzen (Schreiben) und Lesen von Hauptlaufvariablen

Schreiben

In Synchronaktionen können die Hauptlaufvariablen in Aktionen **geschrieben** werden, die in der Liste der Systemvariablen in der 8. Zeile im Feld "write:" mit X gekennzeichnet sind. Geschrieben werden auch im Hauptlauf folgende Maschinen- und Settingdaten:

- Maschinen- und Settingdaten z. B. $$MN_..., $$MC_..., $$MA_...

  bzw. $$SN_..., $$SC_..., $$SA_...

  Hinweis

  Maschinen- und Settingdaten, die im Hauptlauf geschrieben werden sollen, müssen mit **$$**.._... programmiert sein.

Wirksamkeit

Aus Synchronaktionen geschriebene Maschinendaten müssen mit Wirksamkeit SOFORT gekennzeichnet sein, andernfalls steht der veränderte Wert für die weitere Bearbeitung noch nicht zur Verfügung. Die Angaben zur Wirksamkeit neuer Maschinendatenwerte nach Änderungen finden Sie in:

Literatur:

/LIS/ Listen (Buch1)

Beispiele:

... DO **$$MN_MD_FILE_STYLE = 3** ;Maschinendatum setzen

... DO **$$SA_OSCILL_REVERSE_POS1 = 10** ;Settingdatum setzen

... DO **$A_OUT[1]=1** ;Digitalen Ausgang setzen

... DO **$A_OUTA[1]= 25** ;Analogwert ausgeben

Lesen

In Synchronaktionen können die Systemvariablen als Hauptlaufvariablen in Aktionen gelesen werden, die in der Liste der Synchronvariablen in der 7. Zeile im Feld "read" mit X gekennzeichnet sind. Gelesen werden auch folgende Maschinen- bzw. Settingdaten:

- Maschinendaten, Settingdaten z. B. $$SN_..., $$SC_..., $$SA_...

Hinweis

Maschinendaten und Settingdaten, die online im Hauptlauf angesprochen werden sollen, müssen mit **$$**.._... programmiert werden. Für Variablen deren Inhalt sich im Hauptlauf nicht ändert, ist ein $-Zeichen vor dem Bezeichner hinreichend.

Beispiele:

WHEN **$AC_DTEB <** 5 DO ... ;Abstand vom Satzende in Bedingung lesen

DO $R5= **$A_INA[2]** ;Wert des Analogeingangs 2 lesen und Rechenvariable zuweisen

2.6.3 Verändern von SW-Nockenpositionen und -zeiten (Settingdaten)

Einführung

Mit der Funktion "Softwarenocken" können positionsabhängige Nockensignale an die PLC oder an die NCK-Peripherie ausgegeben werden.

Literatur:

/FB2/ Funktionshandbuch Erweiterungsfunktionen; Softwarenocken, Wegschaltsignale (N3)

Funktion

Über Synchronaktionen können Nockenpositionen, bei denen die Signalausgänge gesetzt werden, durch Beschreiben bestehender Settingdaten verändert werden.

Folgende Settingdaten können über Synchronaktionen verändert werden:

$$SN_SW_CAM_MINUS_POS_TAB_1[0..7] ;Positionen der Minusnocken

$$SN_SW_CAM_MINUS_POS_TAB_2[0..7] ;Positionen der Minusnocken

$$SN_SW_CAM_PLUS_POS_TAB_1[0..7] ;Positionen der Plusnocken

$$SN_SW_CAM_PLUS_POS_TAB_2[0..7] ;Positionen der Plusnocken

Beispiel 1

Veränderung einer Nockenposition:

ID=1 WHEN $AA_IW[x] > 0 DO $$SN_SW_CAM_MINUS_POS_TAB_1[0] = 50.0

Vorhalte- bzw. Verzögerungszeiten können über die folgenden Settingdaten verändert werden:

$$SN_SW_CAM_MINUS_TIME_TAB_1[0..7]

;Vorhalte- bzw. Verzögerungszeit an den Minusnocken

$$SN_SW_CAM_MINUS_TIME_TAB_2[0..7]

;Vorhalte- bzw. Verzögerungszeit an den Minusnocken

$$SN_SW_CAM_PLUS_TIME_TAB_1[0..7]

;Vorhalte- bzw. Verzögerungszeit an den Plusnocken

$$SN_SW_CAM_PLUS_TIME_TAB_2[0..7]

;Vorhalte- bzw. Verzögerungszeit an den Plusnocken

Beispiel 2

Änderung einer Vorhalte-/Verzögerungszeit:

ID=1 WHEN $AA_IW[x] > 0

DO $$SN_SW_CAM_MINUS_TIME_TAB_1[0] = 1.0

Hinweis

Das Setzen der Softwarenocken über Synchronaktionen darf geschwindigkeitsabhängig nicht unmittelbar vor einer Nocke geschehen, sondern es müssen mindestens noch 2 - 3 Interpolationstakte bis zum Erreichen der Nocke zur Verfügung stehen.

2.6.4 FCTDEF

Anwendung

Die in den folgenden Unterkapiteln beschriebenen Aktionen Online-Werkzeugkorrektur `FTOC` und Polynomauswertung `SYNFCT` benötigen die Beschreibung eines Zusammenhanges zwischen einer Eingangsgröße und einer Ausgangsgröße durch ein Polynom. `FCTDEF` definiert solche Polynome.

Spezielle Beispiele für den Polynomeinsatz für Online-Abrichten einer Schleifscheibe finden Sie unter Kap. "Online-Werkzeugkorrektur FTOC". Beispiele für lastabhängige Vorschübe und Abstandsregelung über Polynome finden Sie unter Kap. "Polynomauswertung SYNFCT".

Eigenschalten der Polynome

Die mit `FCTDEF` definierten Polynome haben die folgenden Eigenschaften:

- Erzeugung durch Aufruf `FCTDEF` im Teileprogramm
- Die Parameter der definierten Polynome sind Hauptlaufvariablen.
- Überschreiben einzelner Parameter der Polynome wie Schreiben Hauptlaufvariablen, zulässig im Teileprogramm allgemein und im Aktionsteil der Synchronaktionen. Siehe Kap. "Setzen (Schreiben) und Lesen von Hauptlaufvariablen".

Hinweis

Aus Synchronaktionen heraus können Gültigkeitsgrenzen und Koeffizienten von bestehenden Polynomen auch verändert werden.

Beispiel: WHEN ... DO $AC_FCT1[1]= 0.5

Anzahl Polynome

Die Anzahl der Polynome die gleichzeitig definiert werden können werden durch folgendes Maschinendatum vorgegeben:

MD28252 $MC_MM_NUM_FCTDEF_ELEMENTS (Anzahl der FCTDEF-Elemente)

Satzsynchrone Polynomdefinition

FCTDEF(
Polynom-Nr.,
Untergrenze,
Obergrenze,
a0,
a1,
a2,
a3)

Der Zusammenhang zwischen Ausgangsgröße y und Eingangsgröße x ist wie folgt:

$y= a_0+ a_1x+ a_2x^2_+ a_3x^3$

Die in der Funktion angegebenen Parameter werden wie folgt in Systemvariablen abgelegt:

$AC_FCTLL[n]:	Untergrenze, n: Polynomnummer
$AC_FCTUL[n]:	Obergrenze, n: Polynomnummer
$AC_FCT0[n]:	a0-Koeffizient, n: Polynomnummer
$AC_FCT1[n]:	a1-Koeffizient, n: Polynomnummer
$AC_FCT2[n]:	a2-Koeffizient, n: Polynomnummer
$AC_FCT3[n]:	a3-Koeffizient, n: Polynomnummer

In Kenntnis dieses Zusammenhangs können die Polynome auch direkt über die Systemvariablen geschrieben oder verändert werde. Der Gültigkeitsbereich des Polynoms wird durch die Grenzen $AC_FCTLL[n] und $AC_FCTUL[n] festgelegt.

Aufruf der Polynomauswertung

Gespeicherte Polynome können mit den folgenden Funktionen verwendet werden:

- Online Werkzeugkorrektur, `FTOC()`
- Polynom-Auswertung, `SYNFCT()`.

Literatur:

/PG/ Programmierhandbuch Grundlagen

/PGA/ Programmierhandbuch Arbeitsvorbereitung

/FB2/ Funktionshandbuch Erweiterungsfunktionen; Schleifen (W4)

2.6.5 Polynomauswertung SYNFCT

Anwendung

Mit einer Auswertefunktion im Aktionsteil der Synchronaktion kann bearbeitungssynchron eine Variable gelesen, mit einem Polynom bewertet und das Ergebnis in eine andere Variable geschrieben werden. Damit können z. B. folgende Aufgabenstellungen gelöst werden:

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

...

Auswertefunktion SYNFCT()

Die Funktion hat die folgenden Parameter:

SYNFCT(Polynom-Nummer,
Hauptlaufvariable-Ausgang,
Hauptlaufvariable-Eingang)

Die Definition eines Polynoms finden Sie im Kapitel "FCTDEF".

Wirkungsweise SYNFCT

Das mit 'Polynom-Nummer' bestimmte Polynom wird mit dem Wert der 'Hauptlaufvariable-Eingang' ausgewertet. Das Ergebnis wird dann nach oben und nach unten begrenzt und der 'Hauptlaufvariable-Ausgang' zugewiesen.

Beispiel:

```
FCTDEF(1,0,100,0,0.8,0,0)                  ;  Polynom 1 Definition sei erfolgt
...
```

Synchronaktion:

```
ID=1 DO SYNFCT(1,$AA_VC[U1], $A_INA[2])    ;  der additive Korrekturwert der Achse
                                              U1 wird in jedem Interpolationstakt
                                              über Polynom 1 aus dem
                                              Analogeingangswert 2 berechnet
```

Als 'Hauptlaufvariable-Ausgang' können Variable gewählt werden, die folgendermaßen in den Bearbeitungsvorgang eingehen:

- mit additiver Beeinflussung (z. B. Vorschub)
- mit multiplikativer Beeinflussung (z. B. Override)
- als Positionsoffset
- direkt

Additive Vorschub-Beeinflussung

Bei der additiven Beeinflussung wird der programmierte Wert (bei der AC-Regelung das F-Wort) **additiv** korrigiert. $F_{wirksam} = F_{programmiert} + F_{AC}$

Als 'Hauptlaufvariable-Ausgang' werden z. B. gesetzt:

$AC_VC	additive Bahnvorschubkorrektur
$AA_VC[Achse]	additive axiale Vorschubkorrektur

Beispiel: Additive Beeinflussung des Bahnvorschubs

Der programmierte Vorschub (gleich ob axial- oder bahnbezogen) soll **additiv** vom Strom (positiven) der X-Achse (z. B. Zustellmoment) geregelt werden. 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.

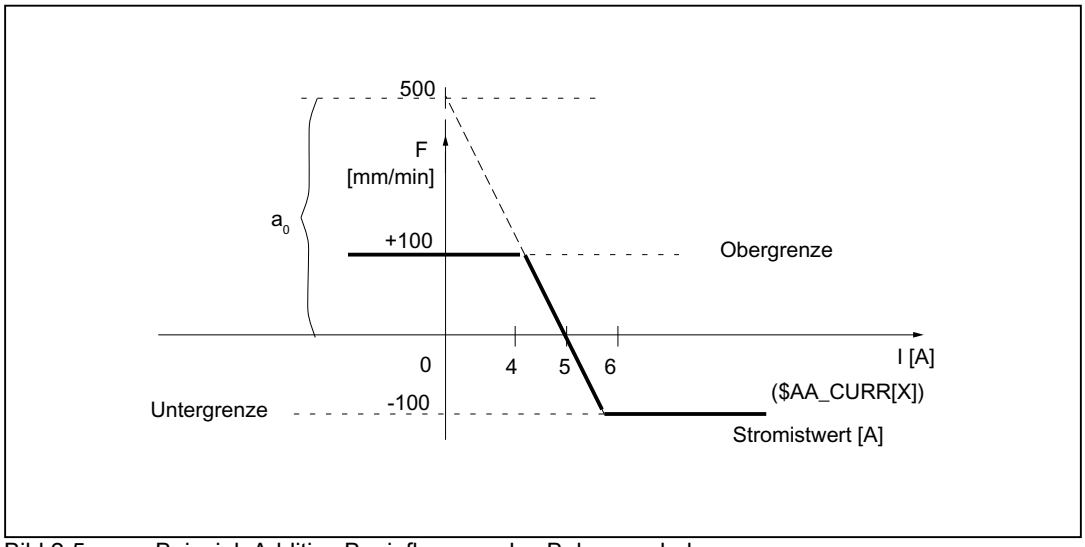


Bild 2-5 Beispiel: Additive Beeinflussung des Bahnvorschubs

Bestimmung der Koeffizienten siehe auch Kap. "FCTDEF":

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

a_1 = 100mm / (1min*A)

a_1 = -100 → Regelkonstante

a_0 = - (-100) *5 = 500

a_2 = 0 (kein quadratisches Glied)

a_3 = 0 (kein kubisches Glied)

Obergrenze = 100

Untergrenze = -100

Damit ist das zu definierende Polynom (Nr. 1):

```
FCTDEF(1, -100, 100, 500, -100, 0, 0)
```

Mit dieser Funktion ist das Bild "Beispiel additive Beeinflussung" vollständig beschrieben.

Mit folgender Synchronaktion wird die *AC-Regelung* eingeschaltet:

```
ID = 1 DO SYNFCT(1, $AC_VC[x],   ; der additive Korrekturwert für den Vorschub der
$AA_LOAD[x])                       Achse x wird in jedem Interpolationstakt über
                                   Polynom 1 aus dem prozentualen Auslastungswert
                                   des Antriebes berechnet
```

Multiplikative Beeinflussung

Bei der multiplikativen Beeinflussung wird das F-Wort mit einem Faktor (bei der AC-Regelung der Override) **multipliziert**:

$F_{wirksam} = F_{programmiert} \_ Faktor_{AC}$

Als 'Hauptlaufvariable-Ausgang' wird die *multiplikativ* auf die Bearbeitung wirkende Variable $AC_OVR verwendet.

Beispiel: Multiplikative Beeinflussung

Der programmierte Vorschub (gleich ob axial- oder bahnbezogen) soll **multiplikativ** in Abhängigkeit von der Antriebsauslastung beeinflusst werden. Der Arbeitspunkt wird dabei auf 100% bei 30%-iger Auslastung des Antriebs festgelegt. Bei 80%-iger Auslastung soll die Achse (n) stehen. Eine Überhöhung der Geschwindigkeit wird mit +20% der programmierten Geschwindigkeit zugelassen.

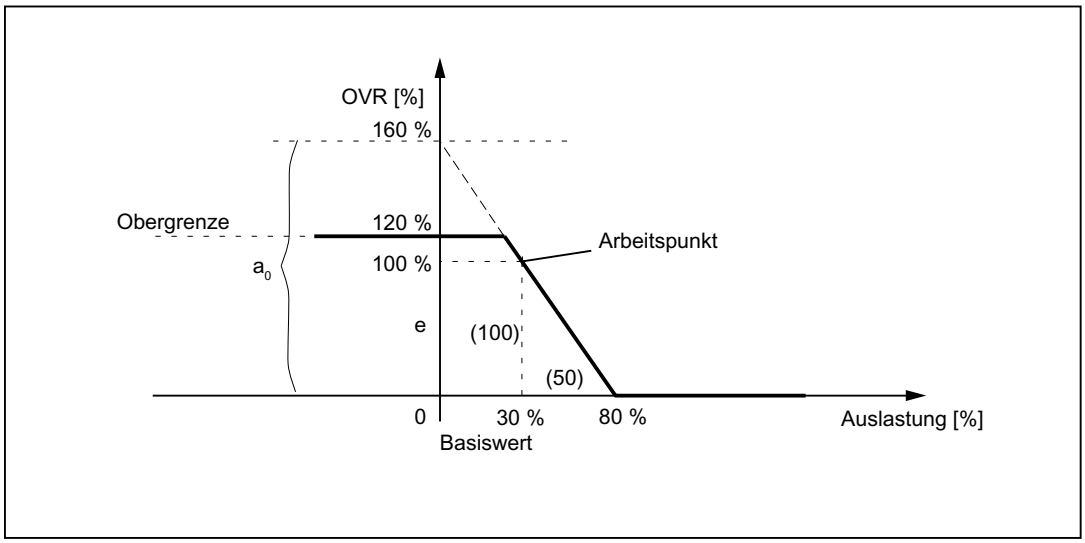


Bild 2-6 Beispiel: Multiplikative Beeinflussung

Bestimmung der Koeffizienten siehe auch Kap. "FCTDEF":

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

$a_1 = - 100\% / (80\% - 30\%) = -2$

$a_0 = 100 + (2 * 30) = 160$

$a_2 = 0$ (kein quadratisches Glied)

$a_3 = 0$ (kein kubisches Glied)

Obergrenze = 120

Untergrenze = 0

Damit kann das Polynom (Nr. 2) definiert werden:

```
FCTDEF(2, 0, 120, 160, -2, 0, 0)
```

Mit dieser Funktion ist das Bild "Beispiel multiplikative Beeinflussung" vollständig beschrieben.

Die dazugehörige Synchronaktion kann wie folgt lauten:

```
ID = 1 DO SYNFCT(2, $AC_OVR,          ;  der Bahnoverride wird in jedem
$AA_LOAD[x])                             Interpolationstakt über Polynom 2 aus der
                                         prozentualen Auslastung des Antriebes für die
                                         x-Achse berechnet
```

Positionsoffset mit Begrenzung

Die Systemvariable $AA_OFF steuert eine achsspezifische Überlagerung, die sofort wirkt (Basis-Koordinatensystem). Die Art der Überlagerung wird festgelegt durch das Maschinendatum:

MD36750 $MA_AA_OFF_MODE (Wirkung der Wertzuweisung für axiale Überlagerung bei Synchronaktion)

0: proportionale Bewertung

1: integrale Bewertung

Es ist auch möglich, den absolut zu korrigierenden Wert (Hauptlaufvariable-Ausgang) auf den Wert zu begrenzen. Hinterlegt ist dieser im Settingdatum:

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

Ob die Begrenzung erreicht wird, kann durch Auswertung der folgenden achsspezifischen Systemvariablen in einer (weiteren) Synchronaktion abgefragt werden:

$AA_OFF_LIMIT[Achse]

Wert	Bedeutung
-1	Limit des Korrekturwertes wurde in negativer Richtung erreicht.
1	Limit des Korrekturwertes wurde in positiver Richtung erreicht.
0	Der Korrekturwert ist nicht im Grenzbereich.

Anwendung:

Die Funktion `SYNFCT` in Verbindung mit der Systemvariablen $AA_OFF kann für eine Abstandsregelung in der Laserbearbeitung benutzt werden.

Beispiel

Aufgabe:

Abstandsregelung als Funktion eines Sensorsignals bei Laser-Bearbeitung.

Der Korrekturwert wird in negativer Z-Richtung begrenzt, damit sich der Laserkopf nicht in bereits gefertigten Blechausschnitten verhakt. Bei erreichtem Grenzwert können Anwenderreaktionen wie Achsen Stoppen (mittels Override 0, siehe Kap. "Sperren einer programmierten Achsbewegung") oder Alarm setzen (siehe. Kap. "Alarm setzen/ Fehlerreaktionen") ausgelöst werden.

Randbedingungen:

Integrierende Bewertung der Eingangsgröße vom Sensor $A_INA[3]. Die Korrektur wirkt im Basiskoordinatensystem, d. h. vor der kinematischen Transformation. Ein evtl. programmierter Frame (`TOFRAME`) wirkt nicht, d. h. die Funktion kann nicht für eine 3D-

Abstandsregelung in Orientierungsrichtung verwendet werden. Eine Abstandsregelung mit hohen Dynamikanforderungen oder eine 3D-Abstandsregelung kann mit der Funktion "Abstandsregelung" realisiert werden. Siehe:

Literatur:
/FB3/ Funktionshandbuch Sonderfunktionen; Abstandsregelung (TE1)
/PG/ Programmierhandbuch Grundlagen

Die Abhängigkeit zwischen Eingangsgröße und Ausgangsgröße sei gegeben durch den im folgenden Bild dargestellten Zusammenhang.

Weitere Beispiele

Im Kapitel "Abstandsregelung mit variabler Obergrenze" finden Sie ein Beispiel mit dynamischer Anpassung einer Grenze des Polynoms bei der AC-Regelung (Abstandsregelung). Im Kapitel "Regelung des Vorschubs" finden Sie ein Beispiel für AC-Regelung des Bahnvorschubes.

Abstandsregelung

Der Abstandswert wird integrierend verrechnet über das Maschinendatum:

MD36750 $MA_AA_OFF_MODE[V]=1 (Wirkung der Wertzuweisung für axiale Überlagerung bei Synchronaktion)

Es wirkt im Basiskoordinatensystem, d. h. vor der Transformation. Damit kann man es als Abstandsregelung in Orientierungsrichtung verwenden (nach Frameanwahl mit `TOFRAME`).

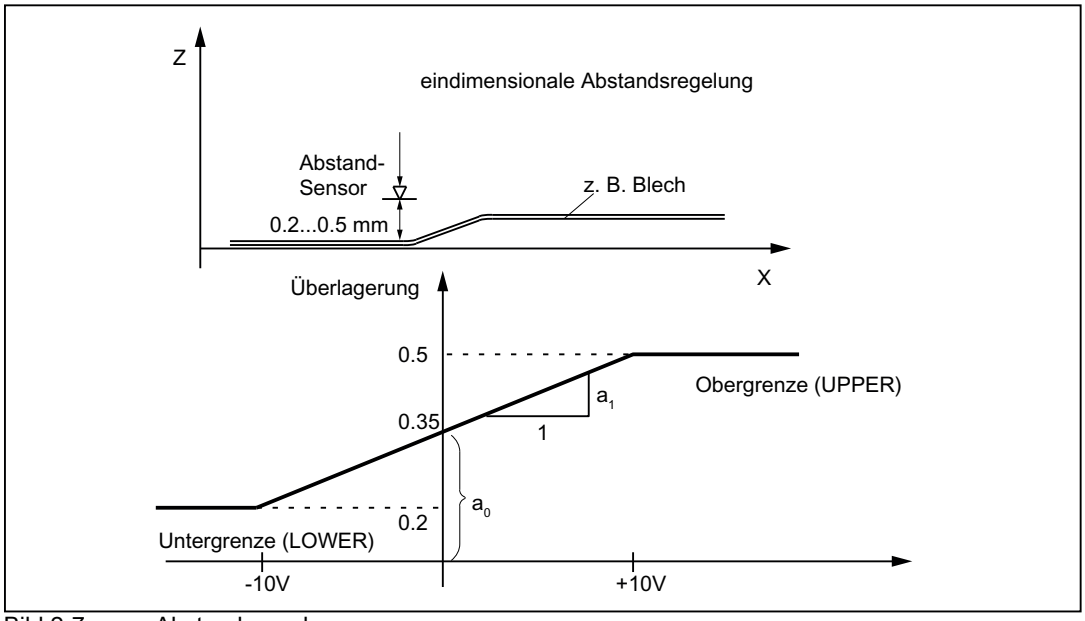


Bild 2-7 Abstandsregelung

```
%_N_AON_SPF

PROC AON                                    ;  Unterprogramm für Abstandsregelung
                                               ein
```

```
FCTDEF(1, 0.2, 0.5, 0.35, 1.5 EX-5)        ;  Polynomdefinition: Die Korrektur
                                              erfolgt im Bereich 0.2 bis 0.5
ID = 1 DO SYNFCT(1,$AA_OFF[Z], $A_INA[3])  ;  Abstandsregelung aktiv
ID = 2 WHENEVER $AA_OFF_LIMIT[Z]<>0 DO     ;  Bei Überschreitung des Grenzbereiches
$AA_OVR[X] = 0                                X sperren
RET
ENDPROC


%_N_AOFF_SPF
PROC AOFF                                  ;  Unterprogramm für Abstandsregelung
                                              aus
CANCEL(1)                                  ;  Synchronaktion Abstandsregl. löschen
CANCEL(2)                                  ;  Grenzbereichsprüfung löschen
RET
ENDPROC


%_N_MAIN_MPF                               ;  Hauptprogramm
                                           ;  MD36750 sei vor Power On auf 1 für
                                              integrierende Bearbeitung gesetzt
$SA_AA_OFF_LIMIT[Z]= 1                     ;  Grenzwert für die Korrektur
AON                                        ;  Abstandsregelung ein
...
G1 X100 F1000
AOFF                                       ;  Abstandsregelung aus
M30
```

2.6.6 Überlagerte Bewegungen $AA_OFF einstellbar (ab SW 6)

Überlagerte Bewegungen bis SW 5.3

Unabhängig vom aktuellen Werkzeug und der Bearbeitungsebene ist über die Systemvariable $AA_OFF eine überlagerte Bewegung für jede Achse des Kanals möglich. Die Verschiebung wird sofort herausgefahren, unabhängig davon, ob die Achse programmiert ist oder nicht. Damit kann eine Abstandsregelung realisiert werden.

Die Art der Verrechnung wird mit dem axialen Maschinendatum wie folgt festgelegt:

MD36750 $MA_AA_OFF_MODE (Wirkung der Wertzuweisung für axiale Überlagerung bei Synchronaktion)

Bit0 = 0: proportionale Verrechnung (absoluter Wert)

Bit0 = 1: integrierende Verrechnung (inkrementeller Wert)

$AC_VACTB und $AC_VACTW als Eingangsvariable für Synchronaktionen und die Ausgabe werden über Optionsbit verriegelt ("Vorschubabhängige Analogwertsteuerung" → Laserleistungssteuerung)!

$AA_OFF, Positionsoffset als Ausgangsvariable für Synchronaktionen für die Abstandsregelung wird über Optionsbit verriegelt!

Geschwindigkeitsbegrenzung mit dem Maschinendatum:

MD32070 $MA_CORR_VELO (Achsgeschwindigkeit für Überlagerung)

Verhalten von $AA_OFF ab SW 6

Nach `RESET` kann der Positionsoffset weiterhin erhalten bleiben

Bisher wurde bei `RESET` der Positionsoffset von $AA_OFF abgewählt. Da dieses Verhalten bei statischen Synchronaktionen IDS = <Nummer> DO $AA_OFF = <Wert> zu einer sofortigen erneuten überlagerten Bewegung mit der Interpolation eines Positionsoffset führt, kann das Verhalten von `RESET` über das folgende Maschinendatum eingestellt werden:

MD36750 $MA_AA_OFF_MODE (Wirkung der Wertzuweisung für axiale Überlagerung bei Synchronaktion)

Bit1 = 0: $AA_OFF wird bei `RESET` abgewählt

Bit1 = 1: $AA_OFF bleibt über `RESET` hinaus erhalten

Überlagerte Bewegung in der Betriebsart JOG

Auch in der Betriebsart JOG kann bei einer Änderung von $AA_OFF eine Interpolation des Positionsoffset als überlagerte Bewegung über das folgende Maschinendatum eingestellt werden:

MD36750 $MA_AA_OFF_MODE (Wirkung der Wertzuweisung für axiale Überlag. bei Synchronaktion)

Bit2 = 0: keine überlagerte Bewegung aufgrund von $AA_OFF

Bit2 = 1: eine überlagerte Bewegung aufgrund von $AA_OFF

Wird ein Positionsoffset aufgrund von $AA_OFF interpoliert, so kann eine Betriebsartenumschaltung nach JOG erst erfolgen, wenn die Interpolation des Positionsoffsets beendet ist. Anderenfalls wird der Alarm 16907 gemeldet.

Aktivierung/Deaktivierung

Die programmierten Bedingungen der aktuellen Bewegungssynchronaktionen werden im IPO-Takt erfasst, bis die Bedingungen erfüllt sind oder das Ende des nachfolgenden Satzes mit Maschinenfunktion erreicht ist.

Ab Software-Stand 3.2 erfolgt mit Einführung einer für Synchronaktionen zugelassenen $$-Hauptvariablen ein Vergleich der Synchronisationsbedingungen im IPO-Takt im Hauptlauf.

Randbedingungen

- **Interruptroutinen/asynchrone Unterprogramme**

  Bei Aktivierung einer Interruptroutine bleiben modale Bewegungssynchronaktionen erhalten und sind auch im asynchronen Unterprogramm 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 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.

- **Programmende**

  Die mit FCTDEF programmierten Polynomkoeffizienten wirken über Programmende hinweg.

- **Satzsuchlauf**

  Bei Satzsuchlauf mit Berechnung werden diese Polynomkoeffizienten aufgesammelt, d. h. in die Settingdaten geschrieben.

CORROF

- Der Teileprogrammbefehl CORROF mit DRFOF wird beim Satzsuchlauf mit aufgesammelt und in einem Aktionssatz ausgegeben. Dabei werden in den letzten vom Suchlauf behandelten Satz mit CORROF oder DRFOF alle abgewählten DRF-Verschiebungen aus Gründen der Kompatibilität aufgesammelt.

  Ein CORROF mit AA_OFF wird beim Satzsuchlauf nicht aufgesammelt und geht verloren. Will ein Anwender diesen Suchlauf weiterhin nutzen, so ist dies mit dem Satzsuchlauf via Programmtest "SERUPRO" möglich. Weitere Details zu diesen Satzsuchlaufs sind beschrieben in:

Literatur:

/FB1/ Funktionshandbuch Grundfunktionen; BAG, Kanal, Programmbetrieb (K1),

- **DRF-Verschiebungen achsspezifisch mit CORROF abwählen**

  MitCORROF sind DRF-Verschiebungen für die einzelnen Achsen nur vom Teileprogramm aus möglich.

- **Abwahl des Positionsoffsets bei aktiven Synchronaktionen**

  Ist bei der Abwahl des Positionsoffsets über den Teileprogrammbefehl COROFF(Achse, "AA_OFF") eine Synchronaktion aktiv, so wird der Alarm 21660 gemeldet. Gleichzeitig wird $AA_OFF abgewählt und nicht wieder gesetzt. Wird die Synchronaktion später im Satz nach CORROF aktiv, so bleibt $AA_OFF gesetzt und es wird ein Positionsoffset interpoliert.

Literatur:

/PG/ Programmierhandbuch Grundlagen

Hinweis

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

2.6.7 Online-Werkzeugkorrektur FTOC

Online-Werkzeugkorrektur

Bei der Technologie Schleifen kann die Bearbeitung des Werkstückes und das Abrichten der Schleifscheibe im gleichen Kanal oder in unterschiedlichen Kanälen (Bearbeitungs- und Abrichtkanal) durchgeführt werden.

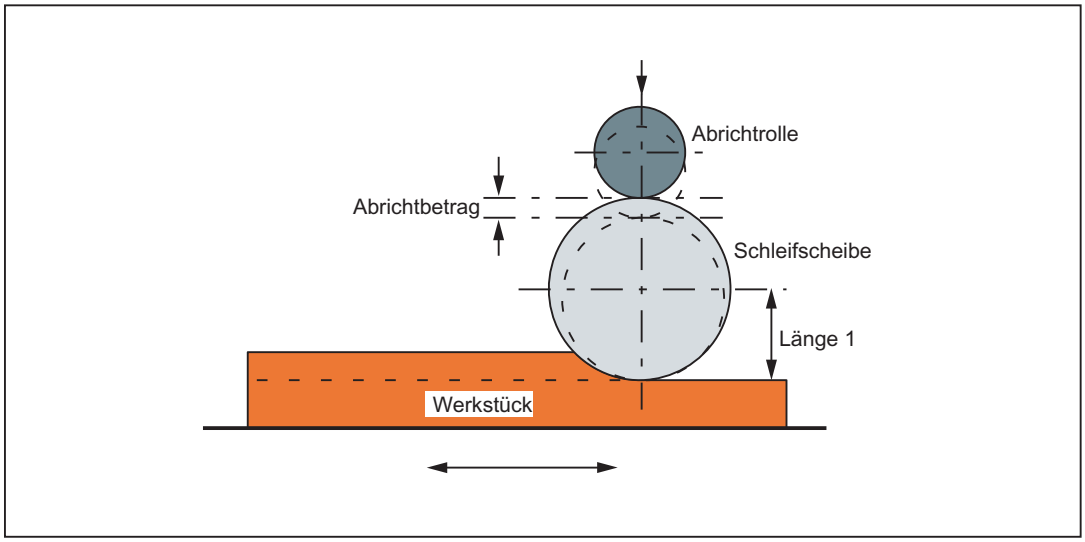


Bild 2-8 Abrichten während der Bearbeitung mit einer Abrichtrolle

Literatur:

/FB2/ Funktionshandbuch Erweiterungsfunktionen; Schleifen (W4)

Randbedingung

Die Synchronaktion `FTOC` steht ab Software-Stand 3.2 zu Verfügung.

Die Online-Korrektur ermöglicht eine überlagerte Bewegung für eine Geometrieachse nach einem mit `FCTDEF` programmierten Polynom (siehe Kap. "FCTDEF") in Abhängigkeit von einem Bezugswert, der z. B. der Istwert einer Achse sein kann.

Der Koeffizient a_0 der Funktionsdefinition `FCTDEF`() wird bei `FTOC` ausgewertet. Ober- und Untergrenze sind abhängig von a_0.

Programmierung FTOC

Die Online-Korrektur wird wie folgt angegeben:

```
FTOC     (Polynom-Nr,
         Real-                      ;  Bezugswert
         Hauptvariable_lesen,
         Länge 1_2_3,
         Kanalnummer,
         Spindelnummer)
```

Parameter

Polynom-Nr:	Nummer der zuvor mit FCTDEF parametrierten Funktion.
Real-Hauptvariable_lesen:	Es sind alle bei Kap. "Liste der für Synchronaktionen bedeutsamen Systemvariablen" aufgeführten Hauptvariablen vom Typ REAL zulässig.
Länge 1_2_3:	Verschleißparameter, in dem der Korrekturwert **addiert** wird.
Kanalnummer:	Zielkanal, in dem die Korrektur wirken soll. Damit ist zeitgleiches Abrichten aus einem parallelen Kanal möglich. Entfällt die Kanalnummer, so wirkt die Korrektur im aktiven Kanal. Im Zielkanal der Korrektur muss die Online-Korrektur mit FTOCON eingeschaltet sein.
Spindelnummer:	Die Spindelnummer wird programmiert, wenn eine nicht aktive Schleifscheibe abgerichtet werden soll. Voraussetzung ist, dass "konstante Scheibenumfangsgeschwindigkeit" oder "Werkzeugüberwachung" aktiv ist. Wird keine Spindelnummer programmiert, so wird das aktive Werkzeug korrigiert.

Beispiel

Länge einer aktiven Schleifscheibe korrigieren

```
%_N_ABRICHT_MPF
FCTDEF(1,-1000,1000,-$AA_IW[V],1)   ;  Definition der Funktion
ID=1 DO FTOC(1,$AA_IW[V],3,1)       ;  Online- Werkzeugkorrektur anwählen:
                                       abgeleitet von der Bewegung der V-Achse wird
                                       in Kanal 1 die Länge 3 der aktiven
                                       Schleifscheibe korrigiert.
WAITM (1,1,2)                       ;  Synchronisation mit Bearbeitungskanal
G1 V-0.05 F0.01, G91
G1 V -...
...
CANCEL(1)                           ;  Online-Korrektur abwählen
...
```

Hinweis

Es wird kein Häufigkeitskennwort und keine Bedingung in der Synchronaktion angegeben, damit wird die Aktion `FTOC` in jedem Interpolationstakt ohne weitere Abhängigkeiten als von $AA_IW[V] wirksam.

2.6.8 Online-Werkzeuglängenkorrektur $AA_TOFF[Index]

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.

Die **Geschwindigkeit,** mit der die Werkzeuglängenkorrektur über $AA_TOFF[ ] für die entsprechende Richtung herausgefahren werden soll, kann eingestellt werden über das Maschinendatum:

MD21194 $MC_TOFF_VELO (Geschw. Online Korrektur in Werkzeugr.)

Die Beschleunigung kann dementsprechend eingestellt werden über das Maschinendatum:

MD21196 $MC_TOFF_ACCEL (Beschl. Online Korrektur in Werkzeugr.)

Hinweis

Die Online-Werkzeuglängenkorrektur ist eine Option, die vorher freigeschaltet werden muss.

Weitere Informationen zur Aktivierung der Funktion im Teileprogramm siehe:

Literatur:

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

Anwendungen in Synchronaktionen

Die Werkzeuglängenkorrekturen werden über eine Synchronaktionsvariable $AA_TOFF[ ] aufgeschaltet. Diese Variable ist 3-dimensional, entsprechend den drei Werkzeugrichtungen.

Als Index werden die drei Geometrieachsbezeichner X, Y, Z verwendet. Damit ist die Anzahl der aktiven Korrekturrichtungen durch die zur selben Zeit aktiven Geometrieachsen festgelegt. Alle Korrekturen können gleichzeitig aktiv sein.

Diese 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 der Abwahl der Online-Werkzeuglängenkorrektur für eine Werkzeugrichtung ist der Wert der Systemvariablen $AA_TOFF[ ] bzw. $AA_TOFF_VAL[ ] für diese Werkzeugrichtung Null.

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 Maschinendatum MD21190 $MC_TOFF_MODE (Wirkungsweise der Korrektur in Werkzeugrichtung) kann über Bit 1 bis 3 eingestellt werden, ob jeweils der Inhalt der drei Komponenten der Synchronaktionsvariable $AA_TOFF[ ]

- Bit 1 bis 3 = 0: als **absoluter Wert** angefahren werden soll, oder ob ein
- Bit 1 bis 3 = 1: **integrierendes Verhalten** erfolgen soll.

Durch das integrierende Verhalten von $AA_TOFF[ ] ist eine 3D-Abstandsregelung möglich. Der aufintegrierte Wert kann über die Variable $AA_TOFF_VAL[ ] gelesen werden.

Beispiel

Anwahl der Online-Werkzeuglängenkorrektur

Maschinendaten für Online-Werkzeuglängenkorrektur setzen:

```
MD21190 $MC_TOFF_MODE = 1               ;  absolute Werte werden angefahren
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:

```
N5 DEF REAL XOFFSET

N10 TRAORI                              ;  Orientierungstransformation aktivieren
N20 TOFFON(Z)                           ;  Aktivierung der Funktion für die
                                        ;  Z-Werkzeugrichtung


Abfrage in Synchronaktion               ;  für die Z-Werkzeugrichtung wird eine
N30 WHEN TRUE DO $AA_TOFF[Z] = 10       ;  Werkzeuglängenkorrektur = 10
G4 F5                                   ;  interpoliert
…
Statische Synchronaktion                ;  für die X-Werkzeugrichtung wird eine
N50 ID=1 DO $AA_TOFF[X] = $AA_IW[X2]    ;  Korrektur abhängig von der
G4 F5                                   ;  Position der eine Achse X2 ausgeführt
```

```
…                                          ;  aktuelle Korrektur
N100 XOFFSET = $AA_TOFF_VAL[X]             ;  in X-Richtung zuweisen
N120 TOFFON(X, -XOFFSET)                   ;  für die X-Werkzeugrichtung wird die
G4 F5                                      ;  Werkzeuglängenkorrektur wieder zu 0
                                              zurückgefahren
```

Beispiel

Abwahl der Online-Werkzeuglängenkorrektur

```
N10 TRAORI                                 ;  Orientierungstransformation aktivieren
N20 TOFFON(X)                              ;  Aktivierung der Funktion für die
N30 WHEN TRUE DO $AA_TOFF[X] = 10          ;  X-Werkzeugrichtung wird eine
G4 F5                                      ;  Werkzeuglängenkorrektur =10 interpoliert
…
N80 TOFFOF(X)                              ;  der Positionsoffset der
                                              X-Werkzeugrichtung wird gelöscht:
                                           ;  ... $AA_TOFF[X] = 0 es wird keine Achse
                                              verfahren, zur aktuellen Position im WKS
                                              wird der Positionsoffset entsprechend der
                                              aktuellen Orientierung hinzugerechnet.
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 dann sofort herausgefahren wird.

Während eine Korrekturbewegung aktiv ist, wird das VDI Signal NCK → PLC NST "TOFF-Bewegung aktiv" auf 1 gesetzt:

DB21, ... DBX318.3 (TOFF Bewegung aktiv).

Nach Abwahl wird das NST "TOFF aktiv" auf 0 gesetzt:

DB21, ... DBX318.2 (TOFF aktiv).

Hinweis

Die Online-Werkzeuglängenkorrektur bleibt solange inaktiv, bis sie über die Anweisung TOFFON() im Teileprogramm wieder angewählt wird.

Werkzeuglängenoffset bei RESET und POWER ON

Das Verhalten bei RESET mit Bit 0 kann eingestellt werden über das Maschinendatum:

MD21190 $MC_TOFF_MODE (Wirkungsweise der Korrektur in Werkzeugrichtung)

Der Werkzeuglängenoffset $AA_TOFF[ ] wird bei MD21190

- Bit 0 = 0: entweder abgewählt, oder er bleibt bei
- Bit 0 = 1: über RESET hinweg erhalten.

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

Ebenso kann eine Transformation oder ein orientierbarer Werkzeugträger **nach** RESET abgewählt werden über das Maschinendatum:

MD20110 $MC_RESET_MODE_MASK (Festlegung der Steuerungs-Grundstellung nach RESET/TP-Ende)

Auch hier muss die Werkzeuglängenkorrektur abgelöscht 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 0 gesetzt.

Nach POWER ON werden alle Werkzeuglängenoffsets auf 0.0 gesetzt.

Die Funktion ist nach POWER ON deaktiviert.

Verhalten bei Betriebsartenwechsel und REPOS

Die Werkzeuglängenkorrektur bleibt auch beim Betriebsartenwechsel aktiv. Die Korrektur kann in allen Betriebsarten außer JOG und REF ausgeführt werden.

Wird beim Betriebsartenwechsel eine Werkzeuglängenkorrektur aufgrund von $AA_TOFF[ ] interpoliert, so kann die Betriebsartenumschaltung erst erfolgen, wenn die Interpolation der Werkzeuglängenkorrektur beendet ist. Es wird der Alarm 16907 "Kanal %1 Aktion %2 <ALNX> nur im Stop-Zustand möglich" ausgegeben.

Die Werkzeuglängenkorrektur ist auch während REPOS aktiv.

Randbedingungen

Bei vorhandenen Werkzeuglängenoffset ist zur Vermeidung von Alarm 21670 "Kanal %1 Satz %2 unzulässige Änderung der Werkzeugrichtung wegen $AA_TOFF aktiv" folgendes zu beachten:

- Transformation TROFOOF abschalten

  Ein Werkzeuglängenoffset muss **vor** einer Transformation im Teileprogramm mit TOFFOF( ) abgelöscht werden.

- CP nach PTP umschalten und PTP-Fahren in der Betriebsart JOG

  Beim Umschalten von CP nach PTP wird eine Transformation abgeschaltet. Ein Werkzeuglängenoffset muss **vor** der Umschaltung abgelöscht werden. Ist beim Wechsel auf achsspezifisches Handverfahren in der Betriebsart JOG eine Werkzeuglängenkorrektur aktiv, so wird der Wechsel nach PTP nicht ausgeführt und es wird der Alarm 21670 gemeldet. CP bleibt solange aktiv, bis die Werkzeuglängenkorrektur über TOFFOF abgelöscht wurde.

- Geometrieachstausch und Ebenenwechsel

  Beim Geo-Achstausch muss ein Werkzeuglängenoffset in Richtung der Geometrieachse vorher über TOFFOF( ) abgelöscht werden.

  Wenn beim Ebenenwechsel ein Werkzeuglängenoffset vorhanden ist, muss dieser vorher über TOFFOF( )abgelöscht werden.

- Satzsuchlauf

  Die Anweisungen TOFFON( ) bzw. TOFFOF( ) werden beim Satzsuchlauf nicht mit aufgesammelt und in einem Aktionssatz ausgegeben.

2.6.9 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 Interpolationstakt 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`	`;`
`...`	`;`

2.6.10 STOPREOF

Beendigung des Vorlaufstop STOPREOF

Eine Bewegungssynchronaktion mit einem `STOPREOF`-Befehl hebt den bestehenden Vorlaufstop auf, wenn die Bedingung erfüllt ist.

`STOPREOF` darf nur mit dem Schlüsselwort 'WHEN' und satzweise wirksam programmiert werden.

Anwendung: Schnelle Programmverzweigung am Satzende.

Beispiel STOPREOF

Programmverzweigungen

```
WHEN $AC_DTEB<5 DO STOPREOF
```

```
G01 X100
IF $A_INA[7]>5000 GOTOF Label 1
```

Wenn die Entfernung zum Satzende 5 mm unterschreitet, beende den Vorlaufstop. Wenn die Spannung 5V am Eingang 7 überschreitet, springe vorwärts bis zum Label 1 (Annahme: Wert 1000 entspricht 1V).

2.6.11 DELDTG

Restweglöschen

Mit Synchronaktionen kann in Abhängigkeit von einer Bedingung Restweglöschen für die **Bahn** und für angegebene **Achsen** ausgelöst werden.

- Schnelles vorbereitetes Restweglöschen

Für die Bahn schnelles, vorbereitetes RWL

Schnelles/vorbereitetes Restweglöschen wird eingesetzt bei zeitkritischen Anwendungen:

- wenn die Zeit zwischen Restweglöschen und Start des Folgesatzes sehr kurz sein soll
- wenn Restweglöschen mit sehr hoher Wahrscheinlichkeit ausgelöst wird

DELDTG

Die Programmierung erfolgt mit der Synchronaktion `DELDTG`.

Nach Ausführung von Restweglöschen steht in der Systemvariable $AC_DELT der Bahnrestweg. Damit wird Bahnsteuerbetrieb am Ende des Satzes mit schnellem Restweglöschen unterbrochen.

Einschränkungen:

Restweglöschen für die Bahn kann nur als satzweise wirksame Synchronaktion programmiert werden.

Bei aktiver Werkzeugradiuskorrektur kann schnelles Restweglöschen nicht verwendet werden.

Befehle:

`MOVE`=1:Geht bei Teilungsachsen mit und ohne Hirthverzahnung MOV=0: Funktioniert bei beiden gleich, es wird die nächste Position angefahren. Befehl: `DELDTG`: Bei Teilungsachsen ohne Hirthverzahnung: Achse steht sofort. Bei Teilungsachsen mit Hirthverzahnung: Achse fährt nächste Position an.

Beispiel DELDTG

```
... DO DELDTG
N100 G01 X100 Y100 F1000
```

```
N110 G01 X...
IF $AC_DELT > 50
...
```

Für Achsen schnelles, vorbereitetes RWL

Schnelles, vorbereitetes Restweglöschen für Achsen kann nur satzweise erfolgen.

Anwendung:

Das Stoppen einer Positionierbewegung, die im Teileprogramm programmiert wurde, erfolgt mit axialem Restweglöschen. Mit einem Befehl können mehrere Achsen gleichzeitig gestoppt werden.

```
... DO DELDTG(Achse1, Achse2, ...)
```

Beispiele DELDTG(Achse)

```
WHEN $A_INA[2]>8000 DO DELDTG(X1)          ;  wenn an Eingang 2 die Spannung von 8
                                              V überschritten wird, Restweg
                                              löschen für Achse X1
POS[X1] = 100                              ;  nächste Position
R10 = $AA_DELT[X 1]                        ;  Übernehmen axialen Restweg in R10
```

Nach erfolgtem Restweglöschen enthält die Variable $AA_DELT[Achse] den axialen Restweg.

(Annahme: Wert 1000 entspricht 1V).

2.6.12 Sperren einer programmierten Achsbewegung

Aufgabe

Die Achse ist innerhalb eines Bearbeitungsprogramms programmiert und soll in speziellen Fällen nicht am Satzbeginn starten.

Lösungsmethode

Per Synchronaktion wird der Override bis zum Startzeitpunkt auf 0 gehalten.

Beispiel:

```
WHENEVER $A_IN[1]==0 DO $AA_OVR[W]=0
G01 X10 Y25 F750 POS[W]=1500 FA[W]=1000    ;  Die Positionierachse wird asynchron
                                              zur
                                           ;  Bahnbearbeitung gestartet;
                                           ;  die Freigabe erfolgt über einen
                                              digitalen Eingang
```

Hinweis

Das Sperren einer Achsbewegung ist auch für PLC-Achsen möglich (z. B. Magazinachse).

2.6.13 Verfahren von Kommandoachsen

In Synchronaktionen können Achsen asynchron zum Teileprogramm positioniert, gestartet gestoppt werden. Eine Achse, die über Synchronaktionen verfahren wird, wird als **Kommandoachse** bezeichnet.

Wechselweises Verfahren aus Teileprogramm und Synchronaktionen

Eine Achse kann nicht gleichzeitig aus dem Teileprogramm und aus Synchronaktionen verfahren werden. Ein abwechselndes Verfahren aus dem Teileprogramm und aus Synchronaktionen ist möglich. Wird eine Achse nach Abschluss der Verfahrbewegung als Kommandochse wieder aus dem Teileprogramm verfahren, kann es zu Wartezeiten kommen.

Achsstatus nach Teileprogrammende oder NC-RESET

Über Maschinendatum kann eingestellt werden, ob nach Teileprogrammende oder NC-RESET eine Achse wieder zur Kanalachse wird oder Kommandoachse bleiben bzw. werden soll:

Achsstatus vor TP-Ende/NC-RESET [1]	MD [2]	Achsstatus nach TP-Ende/NC-RESET [1]
Kanalachse	FALSE	Kanalachse
Kommandoachse	FALSE	Kanalachse
Kanalachse	TRUE	Kommandoachse
Kommandoachse	TRUE	Kommandoachse
1) TP-Ende = Teileprogrammende 2) MD = MD30450 $MA_IS_CONCURRENT_POS_AX		

Beispiele zum Verfahren einer Kommandoachse

Die Verfahrbewegung der Kommandoachse wird direkt in der Synchronaktion programmiert:

Programmcode	**Kommentar**
`ID=1 EVERY G710 $A_IN[1]==1 DO` **POS[X]=100**	`; direkte Programmierung`

Die Verfahrbewegung der Kommandoachse wird in einem Technologiezyklus ACHSE_X programmiert und dieser in der Synchronaktion aufgerufen:

Teileprogramm	**Kommentar**
`ID=1 EVERY G710 $A_IN[1]==1 DO` **POS[X]=100**	`; indirekte Programmierung`

Technologiezyklus: ACHSE_X
N10 M100
N20 G71 **POS[X]=100**
N30 M17

Siehe Kapitel: "Technologiezyklen [Seite 89]"

Definition des Maßsystems

Wird in einer Synchronaktionen (Bedingungsteil und/oder Aktionsteil) nicht explizit ein bestimmtes Maßsystem (Inch / metrisch) programmiert, wirkt in der parallel zum Teileprogramm ausgeführten Synchronaktion das zum Ausführungszeitpunkt im Teileprogramm aktive Maßsystem. Dies hat in der Synchronaktion folgende Auswirkungen:

- 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.

Damit in der Synchronaktion ein definiertes Maßsystem wirkt, muss im Bedingungs- und Aktionsteil das Maßsystem mit G70/G71/G700/G710 explizit festgelegt werden. Es gelten dabei folgende Regeln:

- Wird im Bedingungsteil ein Maßsystem programmiert, wirkt dieses auch im Aktionsteil wenn in diesem kein 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

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

```
Programmcode                                                    Kommentar
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
??: abhängig vom parametrierten Grundsystem oder dem programmierten Maßsystem des Teileprogramms
```

Hinweis

Technologiezyklen

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

Literatur:
/PG/ Programmierhandbuch Grundlagen; Wegangaben

Verfahrbewegungen absolut/inkrementell

Wird in einer Synchronaktionen (Aktionsteil) nicht explizit ein bestimmter Verfahr-Mode (absolut / inkrementell) programmiert, wirkt in der parallel zum Teileprogramm ausgeführten Synchronaktion der zum Ausführungszeitpunkt im Teileprogramm aktive Verfahr-Mode.

Da in einer Synchronaktion nur die G-Funktionen `G70/G71/G700/G710` programmiert werden dürfen, muss, damit in der Synchronaktion ein definierter Verfahr-Mode wirkt, im Aktionsteil der Verfahr-Mode über spezifische Anweisungen programmiert werden.

Anweisung	Beschreibung
`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

Beispiel: Inkrementelles Verfahren um einen festen Wert
Ist die Bedingung erfüllt, wird Achse X inkrementell um 10 mm verfahren:

Programmcode
ID=1 EVERY G710 $AA_IM[B]>75 DO **POS[X]=IC(10)**

Beispiel: Absolutes Verfahren auf eine in Echtzeit berechnete Position
Ist die Bedingung erfüllt, wird Achse X auf die zu diesem Zeitpunkt berechnete Position verfahren:

Programmcode
ID=1 EVERY G710 $AA_IM[B]>75 DO **POS[X]=$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 Maschinendatum:

MD32074 $MA_FRAME_OR_CORRPOS_NOTALLOWED, Bit9 = 1

deaktiviert, wirkt in der parallel zum Teileprogramm ausgeführten Synchronaktion der zum Ausführungszeitpunkt im Teileprogramm aktive Frame und/oder Werkzeuglängenkorrektur.

Beispiel 1
Verfahrbewegungen in Syncronaktionen mit aktiven Frames / Werkzeuglängenkorrekturen (Bit9 = 0):

Programmcode	**Kommentar**
N100 TRANS X20	; Nullpunktverschiebung in X: 20 mm
IDS=1 EVERY G710 $A_IN==1 DO POS[X]=40	; Mit $A_IN==1, X fährt auf Position 60 mm
...	
N130 TRANS X-10	; Nullpunktverschiebung in X: -10 mm
...	; Mit $A_IN=1, X fährt auf Position 30 mm

Beispiel 2
Verfahrbewegungen in Syncronaktionen mit deaktivierten Frames / Werkzeuglängenkorrekturen (Bit9 = 1):

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

Hinweis

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

Kontrolle einer Kommandoachse von PLC

Eine Kommandoachse, die von einer statischen Synchronaktion (IDS) gestartet wurde, wird unabhängig vom Status des Teileprogramms (NC-STOP, Alarmbehandlung, Programmende, -beeinflussungen und NC-RESET) welches die Synchronaktion enthält, wenn die Kontrolle der Kommandoachse von der PLC übernommen wurde:

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

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

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

2.6.14 Axialer Vorschub aus Synchronaktionen

Vorschübe

Zusätzlich zur Endposition kann ein axialer Vorschub programmiert werden:

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

Der axiale Vorschub für Kommandoachsen ist modal wirksam. Er wird unter der Adresse FA programmiert. Der Standardwert wird vergeben über das axiale Maschinendatum:

MD32060 $MA_POS_AX_VELO (Löschstellung für Positionierachsgeschwindigkeit)

Der Vorschubwert wird entweder fest vorgegeben oder in Echtzeit aus Hauptlaufvariablen gebildet:

Beispiel für berechneten Vorschub

ID = 1 EVERY $AA_IM[B] > 75 DO POS[U]=100 **FA[U]=$AA_VACTM[W]+100**

Der Vorschubwert wird entweder als Linear- oder Umdrehungsvorschub programmiert:

Den Vorschubtyp bestimmt das Settingdatum:

SD43300 $SA_ASSIGN_FEED_PER_REV_SOURCE (Umdrehungsvorschub für Positionierachsen/Spindeln)

Das Settingdatum kann per Bedienung oder von PLC, sowie aus dem Teileprogramm verändert werden. Synchron zum Teileprogrammkontext kann der Vorschubtyp über die Sprachbefehle `FPRAON`, `FPRAOF` umgeschaltet werden. Siehe dazu:

Literatur:

/FB1/ Funktionshandbuch Grundfunktionen; Vorschübe (V1)

Vorschubänderung

Zur Übernahme eines geänderten Vorschubs für Kommandoachsen ist nur eine **erneute Vorgabe des Endpunktes** erforderlich, dieser muss nicht geändert sein.

Beispiel:

IDS=1 WHENEVER $A_IN[1] 00 1 DO POS[X] = 100 FA[X] = $R1

Erklärung:

Immer, wenn der analoge Eingang auf 1 wechselt, wir die Position 100 vorgegeben **und** gleichzeitig der **Vorschub** aus dem R-Parameter 1 **übernommen**. Dies gilt auch, wenn bereits ein Anfahren der Position 100 für die Kommandoachse X aktiv ist.

Hinweis

Der axiale Vorschub aus Bewegungssynchronaktionen wird nicht als Hilfsfunktion an die PLC ausgegeben. Parallele axiale Technologiezyklen würden sich sonst gegenseitig blockieren.

2.6.15 Achsen aus Synchronaktionen starten/stoppen

Starten / Stoppen

Kommandoachsen können auch ohne Angabe einer Endposition aus Synchronaktionen gestartet werden. Die Achse wird dann solange in die programmierte Richtung verfahren, bis durch einen erneuten Bewegungs- oder Positionier-Befehl eine andere Bewegung vorgegeben wird, oder die Achse durch einen Stoppbefehl angehalten wird. Damit kann z. B. eine endlos drehende Rundachse programmiert werden.

Die Programmierung erfolgt analog zur Programmierung von Positionierbewegungen.

MOV[Achse]=Wert

Datentyp des Wertes ist INT.

Das Vorzeichen des Wertes bestimmt die Richtung der Bewegung:

>0: Achsbewegung in positive Richtung

<0: Achsbewegung in negative Richtung

==0: Achsbewegung stoppen

Wird eine Teilungsachse aus der Bewegung mit MOV[Achse]=0 gestoppt, so wird wie im konventionellen JOG-Betrieb die nächste Teilungsposition angefahren.

Der **Vorschub** für die Bewegung kann programmiert werden mit **FA[Achse]=Wert**. (S. oben). Ist kein axialer Vorschub programmiert, so ergibt sich der Wert aus einer evtl. bereits aus Synchronaktionen aktivierten Achs-Bewegung oder aus der eingestellten Achsgeschwindigkeit, über das axiale Maschinendatum:

MD32060 $MA_POS_AX_VELO (Löschstellung für Positionierachsgeschwindigkeit)

Beispiel

```
... DO MOV[U]=0                    ;  Achsbewegung stoppen, wenn Bedingung erfüllt
                                      ist.
```

2.6.16 Achstausch aus Synchronaktionen

Anwendung

Für einen Werkzeugwechsel können die betreffenden Kommandoachsen als Aktion einer Synchronaktion mit GET (Achse) angefordert werden. Ist der Werkzeugwechsel vollzogen können diese Kommandoachsen aus der Synchronaktion mit RELEASE (Achse) für den Kanal wieder freigegeben werden.

Achse anfordern

Als Aktion einer Synchronaktion kann mit **GET[Achse]** eine Achse angefordert werden.

Achse freigeben

Als Aktion einer Synchronaktion kann mit **RELEASE[Achse]** eine Achse zum Achstausch freigegeben werden.

Hinweis

Die jeweilige Achse muss dem Kanal über Maschinendaten zugeordnet sein.

Achstyp und Achsstatus bezüglich Achstausch

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

Aus einer Synchronaktion kann eine Achse zum Anforderungszeitpunkt mit **GET[Achse]** 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 Kommando-, 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 kann nicht dem NC-Programm zugeordnet werden. Ebenso eine fest zugeordnete PLC-Achse

Aus einer Synchronaktion kann eine Achse zum Achstausch mit `RELEASE[Achse]` 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 aktiv oder konkurrierende Positionierachse (PLC-Achse) $AA_AXCHANGE_TYP[Achse] == 1, 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.

Eine erneute GET-Aktion fordert die Achse dann für das NC-Programm an $AA_AXCHANGE_TYP[Achse] wird == 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, so ergibt sich **keine** Zustandsänderung.

Achse zum Achstausch freigeben RELEASE(Achse)

Ist die Achse bisher dem NC-Programm des Kanals zugeordnet $AA_AXCHANGE_TYP[Achse] == 0, so 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 der PLC zum Verfahren zugeordnet, PLC-Achse == konkurrierende Positionierachse, $AA_AXCHANGE_TYP[Achse] == 1, so wird die Achse für einen automatischen Achstausch zwischen Kanälen freigegeben. Der Zustand von $AA_AXCHANGE_STAT[Achse] wird von 1 auf 0 zurückgesetzt, falls kein anderer Grund für eine Bindung der Achse an den Kanal vorliegt.

Eine Bindung der Achse liegt z. B. bei Achskopplung, aktives Schnellabheben, aktive Transformation, JOG-Anforderung, rotierendes Frame mit möglicher PLC-, Kommando- oder Pendelachsbewegung vor.

Bereits ein anderer Kanal hat das Schreibrecht

Hat bereits ein anderer Kanal das Schreibrecht bzw. Interpolationsrecht $AA_AXCHANGE_TYP[Achse] == 2, so ergibt sich keine Zustandsänderung. Das bedeutet auch, dass das Warten auf eine Achse (ausgelöst durch NC-Programm $AA_AXCHANGE_TYP[Achse] == 5) oder durch eine vorherige Anforderung `GET(Achse)`

aus Synchronaktion $AA_AXCHANGE_TYP[Achse] == 6 **nicht durch ein** `RELEASE(Achse)` aus eine Synchronaktion abgebrochen werden kann.

Randbedingungen GET, RELEASE

Werden mehrere `GET` und `RELEASE` Aufträge für eine Achse in einer Synchronaktion bzw. in einer Zeile eines Technologiezykluses abgesetzt, so heben sich diese Aufträge gegebenenfalls gegenseitig auf. Es bleibt nur der jeweils letzte Auftrag übrig.

Beispiel:

GET(X,Y) RELEASE(Y,Z) GET(Z) ergibt GET(X) RELEASE(Y) GET(Z).

Innerhalb der Aktionen einer Synchronaktion wird nicht auf die Erfüllung jeweils einer `GET` bzw. `RELEASE` Anforderung gewartet. Dies bedeutet, dass `GET[Achse]POS[Achse]` zu einer Alarmmeldung führen kann, falls die Achse für die Kommandoachsbewegung aktuell nicht zugreifbar ist.

Bei einem Technologiezyklus wird bei einer Aufteilung auf unterschiedliche Zeilen gewartet. D. h. von der Zeile mit `GET(Achse)` wird erst auf die nächste Zeile weitergeschaltet, wenn die Achse z. B. als neutrale Achse von einem anderen Kanal übernommen wurde, siehe nachfolgendes Beispiel `GET`, `RELEASE` im Technologiezyklus.

Beispiel

GET, RELEASE über Synchronaktionen

Achse Z ist im 1. und im 2. Kanal bekannt

Programmablauf im 1. Kanal:

```
WHEN TRUE DO RELEASE(Z)                    ;  Z-Achse wird neutral
WHENEVER $AA_TYP[Z] == 1 DO RDISABLE       ;  Einlesesperre solange
                                           ;  Z-Achse Programmachse ist

N110 G4 F0.1                               ;

WHEN TRUE DO GET(Z)                        ;  Z-Achse wird wieder
                                           ;  NC-Programm-Achse

WHENEVER $AA_TYP[Z] <> 1 DO RDISABLE       ;  Einlesesperre solange
                                           ;  Z-Achse Programmachse ist

N120 G4 F0.1                               ;

WHEN TRUE DO RELEASE(Z)                    ;  Z-Achse wird neutral
WHENEVER $AA_TYP[Z] == 1 DO RDISABLE       ;  Einlesesperre solange
                                           ;  Z-Achse Programmachse ist

N130 G4 F0.1                               ;

N140 START(2)                              ;  2. Kanal starten
```

Programmablauf im 2. Kanal:

```
WHEN TRUE DO GET(Z)                          ;  ; Z-Achse in 2. Kanal holen (neutral)
WHENEVER $AA_TYP[Z] == 0 DO RDISABLE         ;  Einlesesperre solange
                                             ;  Z-Achse in anderem Kanal ist

N210 G4 F0.1                                 ;

WHEN TRUE DO GET(Z)                          ;  Z-Achse wird NC-Programm-Achse
WHENEVER $AA_TYP[Z] <> 1 DO RDISABLE         ;  Einlesesperre bis
                                             ;  Z-Achse Programmachse ist

N220 G4 F0.1                                 ;

WHEN TRUE DO RELEASE(Z)                      ;  Z-Achse im 2. Kanal neutral
WHENEVER $AA_TYP[Z] == 1 DO RDISABLE         ;  Einlesesperre solange
                                             ;  Z-Achse Programmachse ist

N230 G4 F0.1                                 ;

N250 WAITM(10,1,2)                           ;  mit Kanal 1 synchronisieren
N999 M30                                     ;
```

Weiterer Programmablauf im 1. Kanal:

```
N150 WAITM(10,1,2)                           ;  mit Kanal 2 synchronisieren
WHEN TRUE DO GET(Z)                          ;  Z-Achse in diesen Kanal holen
WHENEVER $AA_TYP[Z] == 0 DO RDISABLE         ;  Einlesesperre solange
                                             ;  Z-Achse in anderem Kanal ist

N160 G4 F0.1                                 ;

N199 WAITE(2)                                ;  warte auf Programmende im Kanal 2
N999 M30                                     ;
```

Beispiel

GET, RELEASE im Technologiezyklus

Im 1. und 2. Kanal ist bekannt: Die Achse U, Maschinendatum:

MD30552 $MA_AUTO_GET_TYPE = 2 (Automatisches GET bei Achse holen)

Aktuell hat der Kanal 1 das Interpolationsrecht, im Kanal 2 wird folgender Technologiezyklus gestartet:

```
GET(U)                                   ;  Achse U in Kanal holen
POS[U]=100                               ;  Achse U soll auf Position 100 verfahren
                                            werden
```

Die Zeile mit der Kommandoachsbewegung (POS ...) wird erst ausgeführt, wenn die Achse U in den Kanal 2 geholt wurde.

AXTOCHAN Achse einem anderen Kanal übergeben

Aus einer Synchronaktion kann mit dem NC-Sprachbefehl `AXTOCHAN(Achse, Kanalnummer)[Achse, Kanalnummer[, ...]])` eine Achse für einen bestimmten Kanal angefordert werden.

Dies muss nicht der eigene Kanal sein, der aktuell das Interpolationsrecht für die Achse besitzt. Damit ist es möglich eine Achse in einen Kanal zu schieben.

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

Wird eine Achse für den eigenen Kanal angefordert, so wird mit `AXTOCHAN` aus der Synchronaktion auf ein `GET` aus einer Synchronaktion abgebildet. Es wird bei der

1. **ersten** Anforderung für den eigenen Kanal, die Achse zur neutralen Achse.
2. **zweiten** Anforderung dem NC-Programm zugeordnet, wie dies bei ein `GET` im NC-Programm der Fall ist.

Randbedingungen AXTOCHAN

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

Verfahren über statische Synchronaktionen: Es ist die Stufe 2 notwendig.

Literatur:

/FB2/ Funktionshandbuch Erweiterungsfunktionen; Positionierachsen (P2)

Hinweis

Eine PLC-Achse kann den Kanal nicht wechseln. Dies ist auch nicht über eine Vorgabe von der VDI-Nahtstelle aus möglich.

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

2.6.17 Spindelbewegungen aus Synchronaktionen

Allgemeines

Analog zu Positionierachsen können auch **Spindeln** aus Synchronaktionen gestartet, positioniert, gestoppt werden. Der Start von Spindelbewegungen zu definierten Zeitpunkten kann erreicht werden durch Blockieren einer im Teileprogramm programmierten Bewegung oder durch Steuerung der Achsbewegung aus Synchronaktionen.

Starten/Stoppen

Die Benutzung dieser Funktionen empfiehlt sich für zyklische Abläufe oder Abläufe, die sehr stark ereignisgesteuert sind.

Stoppen bis Ereignis eintrifft

Anwendungsfall:

Die Spindel ist innerhalb eines Bearbeitungsprogramms programmiert und soll in speziellen Fällen nicht am Satzbeginn starten. Per Synchronaktion wird der Override bis zu Startzeitpunkt auf 0 gehalten.

Beispiel:

```
ID=1 WHENEVER $A_IN[1]==0 DO
$AA_OVR[S1]=0
G01 X100 F1000 M3 S1=1000              ;  Die Spindel wird asynchron zur
                                       ;  Bahnbearbeitung gestartet
                                       ;  der Start erfolgt über einen digitalen
                                          Eingang
```

Hilfsfunktionen, Drehzahl, Position

Die Programmierung erfolgt im Aktionsteil der Synchronaktion identisch mit der Programmierung im Teileprogramm.

Befehle: `S= ..., M3, M4, M5, SPOS= ...`

Beispiel:

```
ID = 1 EVERY $A_IN[1]==1 DO M3 S1000
ID = 2 EVERY $A_IN[2]==1 DO SPOS=270
```

Ohne numerische Erweiterung gelten die Befehle jeweils für die Masterspindel. Durch Angabe einer numerischen Erweiterung kann jede Spindel aktiviert werden:

```
ID = 1 EVERY $A_IN[1]==1 DO M1=3 S1=1000 SPOS[2]=90
```

Für die Programmierung der Art der Positionierung gelten dieselben Regeln wie für Positionierachsen (s. o.).

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

Beispiel:

```
ID=1 EVERY $A_IN[1]==1 DO M3 S300      ;  Drehrichtung und Drehzahl
ID = 2 EVERY $A_IN[2]==1 DO M4 S500    ;  Drehrichtung und Drehzahl
ID=3 EVERY $A_IN[3]==1 DO S1000        ;  Neue Drehzahlvorgabe
                                       ;  für aktive Spindeldrehung
ID=4 EVERY ($A_IN[4]==1 ) AND          ;  Spindel positionieren
($A_IN[1]==0) DO SPOS=0
```

Vorschub

Der Vorschub für Spindeln Positionieren kann aus der Synchronaktion programmiert werden mit:

FA[Sn]= ...

Hinweis

Für die Vorschubgeschwindigkeit aus Synchronaktionen steht nur ein modales Datum für Spindelbetrieb und Achsbetrieb zur Verfügung. Hierbei wird FA[S] bzw. FA[C] gleichermaßen versorgt.

SW-Endschalter Arbeitsfeldbegrenzungen

Für Achs-/Spindelbewegungen aus Synchronaktionen gelten auch die Beschränkungen durch SW-Endschalter und Arbeitsfeldbegrenzungen.

Beachtung durch Bewegungen aus Synchronaktionen

Die mit `G25/G26` programmierten Arbeitsfeldbegrenzungen werden berücksichtigt, abhängig vom Settingdatum:

SD43400 $SA_WORKAREA_PLUS_ENABLE (Arbeitsfeldbegrenzung in positiver Richtung aktiv).

Das Ein- und Ausschalten der Arbeitsfeldbegrenzung über G-Funktionen `WALIMON/WALIMOF`**im Teileprogramm** wirkt nicht auf Kommandoachsen.

Beschleunigung korrigieren

Die im folgenden Maschinendatum vorgegebene Beschleunigung kann in einem Bereich von 0% bis 200% verändert werden mit ACC[Achse]=0..200 (**ACC[Achse]**=<Wert> (wirkt im Teileprogramm und in Synchronaktionen)):

MD32300 $MA_MAX_AX_ACCEL (Achsbeschleunigung)

Achskoordinierung

Wird aus Synchronaktionen ein Positionierbefehl `(POS, MOV)` gestartet, während die Achse bereits als Bahnachse oder als PLC-Achse belegt ist, so wird die Bearbeitung mit Alarm abgebrochen.

Achsbewegung wechselweise durch TP und SA

Typischerweise wird eine Achse entweder aus dem Teileprogramm (TP) im Bewegungssatz oder als Positionierachse aus der Synchronaktion (SA) bewegt. Soll dieselbe Achse jedoch wechselweise aus dem Teileprogramm als Bahnachse oder Positionierachse und aus Synchronaktionen verfahren werden, so erfolgt eine koordinierte Übergabe zwischen beiden Achsbewegungen.

Beispiel

; X-Achse wahlweise aus Teileprogramm und Synchronaktionen fahren

```
N10 G01 X100 Y200 F1000              ;  X-Achse im Teileprogramm programmiert
...
N20 ID=1 WHEN $A_IN[1]==1 DO POS[X]=100 FA[X]=200
                                     ;  Positionieren aus Synchronaktion starten,
                                     ;  wenn digitaler Eingang ansteht
...
CANCEL(1)                            ;  Synchronaktion abwählen
...
N100 G01 X100 Y200 F1000             ;  X: Bahnachse
                                     ;  Wartezeit vor Bewegung, wenn digitaler
                                     ;  Eingang 1 war und somit X aus
                                     ;  Synchronaktion positioniert wurde
```

Fliegende Übergänge

Zwischen Kommandoachsen und Spindeln sind Übergänge möglich.

Ausgangssituation

Da mehrere Synchronaktionen gleichzeitig aktiv sein können, ist es möglich, dass eine Achsbewegung gestartet wird, während die Achse bereits aktiv ist.

Verhalten

In diesem Fall wird die **zuletzt aktivierte** Bewegung wirksam. `POS`- und `MOV`-Bewegungen können sich gegenseitig ablösen.

Bei einer so erzwungenen Umkehr der Bewegungsrichtung wird die Achse zunächst abgebremst und dann in die entgegen gesetzte Richtung positioniert.

Beispiele:

```
ID=1 EVERY $AC_TIMER[1] >= 5 DO POS[V]=100 FA[V]=560
ID=2 EVERY $AC_TIMER[1] >= 7 DO POS[V]=$AA_IM[V] + 2 FA[V]=790
```

; Aufgrund der Programmierung mit $AC_TIMER[1] ist die Synchronaktion mit ID=2 die zuletzt aktivierte, ihre Vorgaben werden wirksam und lösen die Vorgaben aus ID=1 ... ab.

Endposition und Vorschub einer Kommandoachse können somit während laufender Bewegung nachgestellt werden.

Beispiel: Auslösung mit Signal

```
ID=1 EVERY $A_IN[1]==1 DO POS[U]=$AA_IM[U]+$AA_IM[V]*.5
                          FA[U]=$AA_VACTM[U]+10
```

Erlaubte Übergänge

im ↓	nach →	POS	MOV=1 MOV= - 1	MOV=0	SPOS	M3 M4	M5	LEADON	TRAIL-ON
Achse steht									
Achsbetrieb		x	x	x	x	x	x	x	x
lagegeregelte Spindel		x	x	x	x	x	x		
drehzahlgere-gelte Spindel					x	x	x		
Achse in Bewegung									
Achsbetrieb		x	x	x				x	x
lagegeregelte Spindel									
drehzahlgere-gelte Spindel					x	x	x		

Die mit x gekennzeichneten Übergänge sind zulässig. Nicht gekennzeichnete Übergänge werden mit Alarm abgewiesen.

Beispiel: Erlaubter Übergang

```
N10 WHEN $AA_IM[Y] >= 5 DO MOV[Y]=-1          ;  Bei Position +5 Achse in
                                              ;  negativer Richtung
                                              ;  starten
N20 WHEN TRUE DO POS[Y]=20 FA[Y]=500          ;  Y-Achse starten, wenn
                                              ;  Satz eingewechselt wird
```

Fliegende Übergänge bei Achskopplungen

Positionierachsbewegungen und Bewegungen als Folge von Achskopplungen über Synchronaktionen können sich gegenseitig ablösen.

Siehe dazu Kapitel 2.4.02 "Mitschleppen und Kopplungen aktivieren, deaktivieren" und:

Literatur:

/FB3/ Funktionshandbuch Sonderfunktionen; Achskopplungen und ESR (M3)

Die erlaubten Übergänge in Leitwertkopplung sind in der Tabelle oben mit `LEADON` gekennzeichnet, die Übergänge in das Mitschleppen mit `TRAILON`.

2.6.18 Istwertsetzen aus Synchronaktionen

Anwendung

Mit der Funktion PRESETON kann der Steuerungsnullpunkt im Maschinenkoordinatensystem neu definiert werden.

Funktion

Bei Preset findet keine Bewegung der Achse statt, es wird für die momentane Achsposition lediglich ein neuer Positionswert eingetragen.

Programmierung

In Synchronaktionen kann jeweils der Wert für **eine** Achse programmiert werden.

Beispiel:

WHEN $AA_IM[a] >= 89.5 DO PRESETON(a, 10.5)

mit PRESETON (Achse,Wert)

Achse: Achse, deren Steuerungsnullpunkt verändert werden soll

Wert: Wert, um den der Steuerungsnullpunkt verändert wird.

Zulässige Anwendungen

PRESETON aus Synchronaktionen ist möglich für:

- Modulo-Rundachsen, die aus dem Teileprogramm gestartet wurden
- alle Kommandoachsen, die aus der Synchronaktion gestartet wurden

Einschränkung

PRESETON ist nicht möglich für Achsen, die an der Transformation beteiligt sind.

Beispiel

Im Kap. "Fliegendes Trennen" finden Sie ein Beispiel für die Verwendung von PRESETON im Zusammenhang mit einer Anwendung Fliegendes Trennen.

Hinweis

Das Istwertsetzen PRESETON darf nur mit dem Schlüsselwort WHEN oder EVERY erfolgen.

2.6.19 Mitschleppen und Kopplungen aktivieren, deaktivieren

Einführung

Literatur:

/FB3/ Funktionshandbuch Sonderfunktionen; Achskopplungen und ESR (M3)

In oben genannter Funktionsbeschreibung sind im Detail folgende Funktionen beschrieben.

- Mitschleppen

  Folgeachse(n) sind über einen Koppelfaktor mit einer Leitachse verbunden.

- Kurventabellen

  Kurventabellen stellen einen (komplexen) Zusammenhang zwischen Leitwert und Folgewert dar. Als Leitwert sind möglich:

  - von der Steuerung erzeugte Sollwerte
  - vom Geber ermittelte Istwerte
  - extern vorgegebene Größen

    Im Zusammenhang mit den Synchronaktionen ist besonders der Fall von Bedeutung, dass eine Folgeachse über Kurventabelle mit einer Leitachse verbunden wird.

- Leitwertkopplung

  Folgende, von den für Teileprogramme möglichen, Leitwertkopplungen stehen für die Nutzung in Synchronaktionen nur Achsleitwertkopplungen zur Verfügung:

  - Achsleitwertkopplung
  - Bahnleitwertkopplung

- Elektronisches Getriebe

  Mit Hilfe der Funktion "Elektronisches Getriebe" kann die Bewegung einer Folgeachse FA abhängig von bis zu fünf Leitachsen LA interpretiert werden. Damit wird ein Getriebeverband aus dem Teileprogramm:

  - Definiert
  - Eingeschaltet
  - Ausgeschaltet
  - Gelöscht

- Generische Kopplung

  Mit Hilfe eines Koppelmoduls kann die Bewegung einer Achse (_FA Folgeachse) abhängig von anderen Achsen (_Leitachsen) interpoliert werden. Koppelmodule können im Teileprogramm und implizit Synchronaktionen angelegt und aktiviert werden. Diese in Synchronaktionen angelegten und aktivierten Koppelmodule werden als Hauptlaufkopplungen bezeichnet. Die Zusammenhänge zwischen den Leitachsen bzw.

Leitwerten und der Folgeachse sind je Leitachse bzw. Leitwert durch ein Koppelgesetz, entweder einen Koppelfaktor oder eine Kurventabelle definiert.

Über Schlüsselwörter kann jede Kopplungseigenschaft der Generischen Kopplung programmiert werden. In Synchronaktionen stehen folgende Schlüsselwörter zur Verfügung:

- CPLON Einschalten einer Leitachse eines Koppelmoduls
- CPLOF Ausschalten einer Leitachse eines Koppelmoduls
- CPOF Ausschalten eines Koppelmoduls
- CPLNUM Zähler des Koppelfaktors
- CPLDEN Nenner des Koppelfaktors
- CPLCTID Nummer der Kurventabelle
- CPSETTYPE Kopplungstyp der bisher bestehenden Kopplungsarten

Hinweis

Bei der Programmierung ist darauf zu achten, dass die Abarbeitung der verwendeten CP-Schlüsselworte innerhalb einer Synchronaktion **von links nach rechts** erfolgt.

D. h. im Gegensatz zur Programmierung im Teileprogramm ist die Wirkung der verschiedenen Schlüsselwörter von der Reihenfolge in der Synchronaktion abhängig.

TRAILON - Mitschleppen aus einer Synchronaktion

Aus einer Synchronaktion heraus kann die Zuordnung einer Folgeachse zu einer Leitachse mit einem Koppenfaktor definiert und gleichzeitig aktiviert werden:

```
... DO TRAILON(FA, LA, Kf)
```

mit:

FA	Folgeachse
LA	Leitachse
Kf	Koppelfaktor

Die Auflösung des Mitschleppverbandes erfolgt mit:

```
... DO TRAILOF(FA, LA, LA2)
```
oder

```
... DO TRAILOF(FA)
```
in verkürzter Form

mit:

FA	Folgeachse
LA	Leitachse
LA2	Leitachse2, optional

Kurventabellen

Der in Kurventabellen hinterlegte Zusammenhang zwischen einer Leitwertgröße und einer Folgewertgröße kann in Synchronaktionen benutzt werden wie andere `REAL`-Funktionen (z. B. `SIN, COS`).

Folgewert ermitteln

Der sich aus einem Leitwert über die Kurventabelle n ergebende Folgewert soll einer Rechenvariablen zugewiesen werden.

Beispiel:

```
... DO $R17=CTAB(LW, n, grad)
```

mit:

LW	Leitwert
n	Nummer der Kurventabelle
grad	Steigungsparameter, Ergebnis 2 weitere opt. Parameter für Skalierung: - Folgeachse - Leitachse

Beispiel:

```
DEF REAL GRADIENT
...
WHEN $A_IN[1] == 1 DO $R17 = CTAB(75.0, 2, GRADIENT)
```

Leitwert ermitteln

Aus einer Synchronaktion heraus kann ein konkreter Leitwert anhand einer definierten Kurventabelle für einen Folgewert ermittelt werden.

Beispiel:

```
... DO $R18=CTABINV(FW, aprLW, n, grad)
```

mit:

FW	Folgewert
aprLW	angenäherter Leitwert, mit dem bei mehrdeutiger Umkehrfunktion der Kurventabelle ein eindeutiger LW bestimmt werden kann
n	Nummer der Kurventabelle
grad	Steigungsparameter, Ergebnis: 2 weitere opt. Parameter für Skalierung: - Folgeachse - Leitachse

Die Funktionen `CTAB` und `CTABINV` sind sowohl in Bedingungen als auch im Aktionsteil von Synchronaktionen möglich.

LEADON Achsleitwertkopplung aus Synchronaktionen

Im Aktionsteil der Synchronaktion wird die Ankopplung der Folgeachse FA an die Leitachse LA über die gespeicherte Kurventabelle mit der Nummer NR wie folgt aufgerufen:

```
... DO LEADON(FA; LA, NR)
```

mit:

FA	Folgeachse
LA	Leitachse
NR	Nummer der Kurventabelle

LEADOF Achskopplung aus Synchronaktion ausschalten

Soll die Achs-Leitwertkopplung beim Eintreffen einer weiteren Bedingung wieder aufgehoben werden, so lautet die Aktion:

`... DO LEADOF(FA, LA)` oder

`... DO LEADOF(FA)` in der verkürzten Form

Systemvariablen

Vom Teileprogramm und aus Synchronaktionen sind die Systemvariablen der Leitwertkopplung laut Liste der Systemvariablen lesbar/schreibbar.

Literatur:

PGA1 / Listenhandbuch Systemvariablen

Erkennen des Synchronlaufes

Die aus Teileprogramm und Synchronaktion lesbare Systemvariable $AA_SYNC[ax] zeigt an, ob und wie die Folgeachse FA synchronisiert ist:

0: nicht synchron

1: Synchronlauf grob (gemäß MD37200 $MA_COUPLE_POS_TOL_COARSE)

2: Synchronlauf fein (gemäß MD37210 $MA_COUPLE_POS_TOL_FINE)

Anwendungsabgrenzung

Im Teileprogramm direkt aktivierte Kopplungen werden an Satzgrenzen aktiviert. Mit der Möglichkeit der Kopplungsaktivierung durch Synchronaktionen wird eine ereignisgesteuerte differenzierte Aktivierung ermöglicht z. B.

- bei bestimmten Achsweg ab Satzanfang
- bei bestimmten Restweg bis Satzende
- Eintreffen von Digitaleingangssignalen
- Kombinationen aus diesen

  Siehe Kapitel "Komponenten von Synchronaktionen", Bedingungen

Weitere Hinweise zu Programmierung der Kopplungsfunktionen und Kurventabellen finden Sie in:

Literatur:

/PGA/ Programmierhandbuch Arbeitsvorbereitung

Hinweis

Achsen, die beim Einkoppeln über Synchronaktionen in einem beliebigen Bewegungszustand angetroffen werden, werden durch die Steuerung synchronisiert. Details hierzu finden Sie in:
/FB3 / Funktionshandbuch Sonderfunktionen; Achskopplungen und ESR (M3).

Beispiele

Im Kapitel Beispiele "Fliegendes Trennen" finden Sie ein Beispiel von Achskopplung über Kurventabellen.

- Generische Kopplung - Koppelmodule in Synchronaktionen aktivieren

  Die Programmierung von Schlüsselwörtern ist in Synchronaktionen möglich. Damit sind Kopplungsmodule auch in Synchronaktionen verwendbar. Bei der Aktivierung des Koppelmoduls in einer Synchronaktion muss die Folgeachse bereits im Kanal aktiv sein und sich im Zustand "Neutrale Achse oder "Achse bereits dem NC-Programm des Kanals zugeordnet" befinden.

  Dieser Achszustand muss bei Bedarf vor der Aktivierung des Koppelmoduls bereitgestellt werden. Dies kann auch in Synchronaktionen mittels `GET [Achse]` erfolgen.

- Achstausch - Kanalübergreifende 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 ei dieser Art von Kopplung somit nicht möglich.

  Weitere Erläuterungen zum Achstausch in Synchronaktionen siehe auch Kapitel "Achstausch aus Synchronaktionen"; `GET / RELEASE [Achse]`

- Kopplung aktivieren / deaktivieren

  Die CP-Schlüsselwörter werden in Synchronaktionen direkt durch das Koppelmodul verarbeitet. Damit wird ein CP-Schlüsselwort sofort wirksam. Aktivierung der Kopplung einer Leitachse zu einer Folgeachse:

  CPLON[Fax]=<Leitachse bzw. Leitspindel>

  Deaktivierung der Kopplung einer Leitachse zu einer Folgeachse:

  CPLOF[Fax]=<Leitachse bzw. Leitspindel>

  Mit dem folgenden Schlüsselwort werden alle Leitachsen zur Folgeachse in Synchronaktionen deaktiviert

  SPOF=<FAx>

- Koppelfaktor

  Mit der Programmierung eines Koppelfaktors wird ein zuvor aktiviertes nichtlineares Koppelverhältnis z.B. einer Kurventabelle deaktiviert. Zähler des Koppelfaktors in Synchronaktionen:

  CPLNUM[FAx.Lax]=<Wert>

  Nenner des Koppelfaktors in Synchronaktionen:

  CPLDEN[FAx.Lax]=<Wert>

- Kurventabelle

  Mit der Programmierung einer Tabellennummer wird ein zuvor aktiviertes nichtlineares Koppelverhältnis z. B. einer Kurventabelle deaktiviert. In Synchronaktionen wird zum Leitwert der Leitachse / -spindel mittels der angegebenen Kurventabelle der spezifische Koppelteil für die Leitachse / -spindel berechnet:

  CPLCTID[FAx.Lax]=<Wert>

- Beispiel:

  Programmierung mit Schlüsselwörtern in Synchronaktionen:

  Beispiel 1:

  Definition einer Achskopplung mit einer Leitachse

  DO CPDEF=YCPLEDF[Y]=X CPLNUM[Y;X]=1,5

  Beispiel 2:

  N10 WHEN TRUE DO CPLON[X]=X CPLNUM[X,Y]=2; OK

  N20 WHEN TRUE DO CPLNUM [A,B]=" CPLON [A=B] ; Alarm

  Die Reihenfolge im Satz N20 ist nicht erlaubt, da CPLNUM gesetzt werden soll, bevor das Koppelmodul mittels CPDEF im Teileprogramm angelegt wurde.

  Beispiel 3:

  N10 WEHN TRUE DO CPLON [X]=Y CPLNUM[X,Y]=3

  N15 Y= 100 F100

  N20= WHEN TRUE DO CPOF=X CPLON[X]=YCPLNUM[X,Y]=3

  In diesem Beispiel wird das in N10 aktive Koppelmodul wieder angelegt und wieder aktiviert und hat eine Neusynchronisation zur Folge.

  Beispiel 4:

  N10 WHEN TRUE DO CPLON[X]=Y CPLNUM[X,Y]

  N15 Y=100 F100

  N20 WHEN TRUE DO CPOF0X MOV[X]=1

  Im Satz N20 wird das Koppelmodul mit CPOF ausgeschaltet und gelöscht. Die Folgeachse steht somit wieder für den Befehl MOV zur Verfügung.

- Bisherige Kopplungstypen verwenden - Bestehende Kopplungsarten TRAIL, LEAD, EG und COUP.

  Wird eine Voreinstellung der bisher bestehenden Kopplungsarten wie Mitschleppen, Leitwertkopplung, Elektronisches Getriebe oder Synchronspindel gewünscht, ist beim Anlegen oder Definieren des Koppelmoduls zusätzlich das Schlüsselwort

  CPSETTYPE[FAx]=<Wert>

  In Synchronaktionen erlaubt.

  Als Wertebereich sind möglich:

  - "CP" Freie Programmierbarkeit (Standartwert)
  - "TRAIL" Kopplungstyp "Mitschleppen"
  - "LEAD" Kopplungstyp "Leitwertkopplung"
  - "EG" Kopplungstyp "Elektronisches Getriebe"
  - "COUP" Kopplungstyp "Synchronspindel"

2.6.20 Messen aus Synchronaktionen

Einführung

Für die Teileprogramme verfügbare Messfunktionen:

`MEAS, MEAW, MEASA, MEAWA, MEAC`

Literatur:

/PGA/ Programmierhandbuch Arbeitsvorbereitung

/FB2/ Funktionshandbuch Erweiterungsfunktionen; Messen (M5)

Davon stehen in Synchronaktionen nur folgende zur Verfügung:

- `MEAWA` axiales Messen ohne Restweglöschen
- `MEAC` axiales, kontinuierliches Messen

Während die Messfunktion bei Bewegungssätzen im Teileprogramm jeweils auf einen Satz begrenzt ist, kann die Messfunktion aus Synchronaktionen beliebig ein- und ausgeschaltet werden:

Hinweis

Mit statischen Synchronaktionen steht Messen auch in der Betriebsart JOG zur Verfügung.

Programmierung

```
MEAWA[Achse]=    (Modus, Triggerereignis_1, Triggerereignis_2, Triggerereignis_3,
                 Triggerereignis_4)
                 ; axiales Messen ohne Restweglöschen einschalten
MEAC[Achse]=     (Modus, Messspeicher,Triggerereignis_1, Triggerereignis_2,
                 Triggerereignis_3, Triggerereignis_4)
                 ; axiales, kontinuierliches Messen einschalten
Achse:           Achse, für die gemessen wird
```

Tabelle 2-2 Modus-Bedeutungen

Zehnerdekade	Einerdekade	Bedeutung
	0	Messauftrag wird abgebrochen
	1	bis zu 4 **gleichzeitig** aktivierbare Triggerereignisse
	2	bis zu 4 **nacheinander** aktivierbare Triggerereignisse
	3	bis zu 4 **nacheinander** aktivierbare Triggerereignisse, jedoch keine Überwachung von Triggerereignis1 beim START
0		aktives Messsystem
1		1. Messsystem
2		2. Messsystem
3		beide Messsysteme

Triggerereignis_1 bis Triggerereignis_4:

	1:	steigende Flanke Messtaster 1	
-	1:	fallende Flanke Messtaster 1	*optional*
	2:	steigende Flanke Messtaster 2	*optional*
-	2:	fallende Flanke Messtaster 2	*optional*

Messspeicher: Nummer einer FIFO-Variablen

Messwerte werden ausschließlich für das **Maschinen**koordinatensystem bereitgestellt.

MEAWA

`... DO MEAWA[Achse]=( , , , , );` axiales Messen ohne Restweglöschen

Restweglöschen kann bei Bedarf in der Synchronaktion explizit aufgerufen werden. Siehe Kap. "DELDTG" und Beispiel unten.

GEO-Achsen und an Transformationen beteiligte Achsen können einzeln programmiert werden.

Programmierung:

Die Programmierung entspricht derjenigen im Teileprogramm.

Hinweis

Die Systemvariable $AC_MEA liefert für eine aus der Synchronaktion aufgerufene Messung keine auswertbare Information über die Gültigkeit der Messung.

Pro Achse darf nur ein Messauftrag aktiv sein.

Systemvariablen:

$AA_MEAACT[Achse]	liefert den augenblicklichen Messstatus einer Achse.
1	Messung aktiv
0	Messung nicht aktiv
$A_PROBE[Messtaster]	liefert den momentanen Zustand des Messtasters.
1	Taster geschaltet, High Signal
0	Taster nicht geschaltet, Low Signal

Messwerte im Maschinenkoordinatensystem mit 2 Messtastern (Gebern):

$AA_MM1[Achse]	Triggerereignis1, Geber 1
$AA_MM2[Achse]	Triggerereignis 1, Geber 2
$AA_MM3[Achse]	Triggerereignis 2, Geber 1
$AA_MM4[Achse]	Triggerereignis 2, Geber 2

MEAC

`... DO MEAC[Achse]=(Modus, Nr_FIFO, Triggerereignisse)`

Die Variablen $AC_FIFO (siehe Kap. "FIFO-Variablen (Durchlaufspeicher)".) sind dafür vorgesehen, Messwerte aus zyklischen Messvorgängen aufzunehmen. Modus und Triggerereignisse s. o.

Beispiele:

Für die folgenden Beispiele wurden per Maschinendaten 2 FIFO's eingerichtet.

Maschinendaten:

```
MD28050 $MC_MM_NUM_R_PARAM = 300
MD28258 $MC_MM_NUM_AC_TIMER = 1
MD28260 $MC_NUM_AC_FIFO = 2              ;  2 FIFO's
MD28262 $MC_START_AC_FIFO = 100          ;  erster FIFO beginnt ab R100
MD28264 $MC_LEN_AC_FIFO = 22             ;  jeder FIFO kann 22 Werte aufnehmen
MD28266 $MC_MODE_AC_FIFO = 0             ;  keine Summenbildung
```

Beispiel 1:

Auf einer Strecke zwischen X0 und X100 sollen alle steigenden Flanken von Messtaster 1 aufgenommen werden. Es wird angenommen, dass nicht mehr als 22 Flanken auftreten können.

Programm 1:

```
DEF INT ANZAHL
DEF INT INDEX_R
N0   G0 X0                               ;  Modus = 1, gleichzeitig
N1   MEAC[X]=( 1, 1, 1) POS[X]=100       ;  Nr-FIFO = 1
                                         ;  Triggerereignis 1= steigende Flanke,
                                         ;  Messgeber 1
N2   STOPRE                              ;  Anhalten Vorverarbeitung
N3   MEAC[X]=( 0)                        ;  Abbrechen kontinuierliche Messung
N4   ANZAHL= $AC_FIFO1[4]                ;  Anzahl eingetroffener Messwerte in
                                         ;  der FIFO-Variablen
N5   ANZAHL= ANZAHL - 1
N6   FOR INDEX_R= 0 TO ANZAHL
N7   R[INDEX_R]= $AC_FIFO1[0]            ;  FIFO-Inhalt in R0 - ... eintragen
N8   ENDFOR                              ;  Nach Auslesen ist FIFO-Variable leer
```

Beispiel 2:

Auf einer Strecke zwischen X0 und X100 sollen alle steigenden und fallenden Flanken von Messtaster 1 aufgenommen werden. Die Anzahl der erreichbaren Triggerereignisse ist unbekannt. Daraus folgt: Es müssen parallel in einer Synchronaktion die Messwerte abgeholt und ab R1 aufsteigend abgelegt werden. Die Anzahl der abgelegten Messwerte wird im R0 eingetragen.

Programm 2:

```
N0   G0 X0                                 ; Eilgang zum Startpunkt
N1   $AC_MARKER[1]=1                       ; Merker 1 als Index für Rechenvariablen
                                             R[..]
N2   ID=1 WHENEVER $AC_FIFO1[4]>=1         ; Synchronaktion als Prüfung: wenn 1
     DO $R[$AC_MARKER[1]]= $AC_FIFO1[0]    ; oder mehr Messwerte in FIFO-Variable
     $AC_MARKER[1]=$AC_MARKER[1]+1         ; stehen ältesten Wert aus FIFO auslesen
                                           ; und in aktuelle R[..] ablegen,
                                           ; Index für R um 1 erhöhen
```

```
N3   MEAC[X]=( 1, 1, 1, -1) POS[X]=100      ; Kontinuierliches Messen aktivieren,
                                            ; Bewegung nach X = 100
                                            ; Modus = 1, gleichzeitig
                                            ; Nr_FIFO = 1; Triggerereignis 1= 1,
                                            ; steigende Flanke Messgeber 1
                                            ; Triggerereignis 2= -1,
                                            ; fallende Flanke Messgeber 1
N4   MEAC[X]=(0)                            ; Messung abwählen
N5   STOPRE                                 ; Vorverarbeitung stoppen
N6   R0= $AC_MARKER[1]                      ; Anzahl der erfassten Werte in R0
```

Beispiel 3:

Kontinuierliches Messen mit explizitem Restweglöschen nach 10 Messungen

Programm 3:

```
N1   WHEN $AC_FIFO1[4]>=10                ;  Schlussbedingung als Synchronaktion:
     DO MEAC[X]=(0) DELDTG(X)             ;  Wenn 10 oder mehr Messwerte in
                                          ;  der FIFO-Variablen
                                          ;  vorliegen,
                                          ;  kontinuierliche Messung abwählen und
                                          ;  Restweg löschen
N2   MEAC[X]=( 1,1,1,-1) G01 X100 F500    ;  kontinuierliche Messung aus dem
                                          ;  Teileprogramm aktiv.
                                          ;  Modus = 1, gleichzeitig
                                          ;  Nr_FIFO = 1, FIFO-Variable 1
                                          ;  Triggerereignis 1= 1,
                                          ;  steigende Flanke Messgeber 1
                                          ;  Triggerereignis 2= -1,
                                          ;  fallende Flanke Messgeber 1
N3   MEAC[X]=(0)                          ;  Kontinuierliche Messung abwählen
N4   R0= $AC_FIFO1[4]                     ;  tatsächliche Anzahl Messwerte
```

Priorität bei mehreren Messungen

Zu einem Zeitpunkt kann pro Achse genau ein Messauftrag aktiv sein.

Der Start eines Messauftrags für dieselbe Achse bewirkt, dass die Triggerereignisse erneut aktiviert und die Messergebnisse zurückgesetzt werden. Wird Messauftrag ausschalten (Modus 0) programmiert, ohne dass vorher ein Messauftrag aktiviert wurde, so erfolgt keine gesonderte Reaktion. Messaufträge, die aus dem Teileprogramm gestartet wurden, können aus Synchronaktionen nicht beeinflusst werden. Wird aus Synchronaktionen ein Messauftrag für eine Achse gestartet, und für diese Achse ist bereits ein Messauftrag aus dem Teileprogramm aktiv, so wird ein Alarm generiert. Ist ein Messauftrag aus Synchronaktionen aktiv, kann Messen aus dem Teileprogramm heraus nicht mehr gestartet werden.

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.

2.6.21 Setzen und Löschen von Wartemarken der Kanalsynchronisation

Einführung

Die Koordination von Abläufen in den Kanälen ist beschrieben in:

Literatur:

/FB1/ Funktionshandbuch Grundfunktionen; BAG, Kanal, Programmbetrieb (K1)

Von den dort genannten Funktionen sind die folgenden in Synchronaktionen zulässig:

Wartemarke setzen

Der Befehl `SETM`(MarkerNummer) kann im Teileprogramm und im Aktionsteil einer Synchronaktion gegeben werden. Er setzt die Marke MarkerNummer für den Kanal, in dem der Befehl läuft. (Eigener Kanal).

Wartemarke löschen

Der Befehl `CLEARM`(MarkerNummer) kann im Teileprogramm und im Aktionsteil einer Synchronaktion gegeben werden. Er löscht die Marke MarkerNummer für den Kanal, in dem der Befehl läuft. (Eigener Kanal).

2.6.22 Alarm setzen/Fehlerreaktionen

Fehlersituationen

Alarm setzen ist eine Möglichkeit auf Fehlerzustände zu reagieren.

Anwendung:

Mit dem SETAL-Befehl können Zyklen-Alarme aus Synchronaktionen gesetzt werden.

Weitere Möglichkeiten auf Fehler zu reagieren sind:

- Achse stoppen, siehe Kapitel 2.4.12 "Sperren einer programmierten Achsbewegung"
- Ausgang setzen, siehe Kapitel 2.4.2 "Setzen (Schreiben) und Lesen von Hauptlaufvariablen"
- Sonstige im Kapitel 2.4 "Aktionen in Synchronaktionen" aufgeführte Aktionen

Beispiel Alarm setzen

```
D=67 WHENEVER $AA_IM[X1] - $AA_IM[X2] < 4.567 DO SETAL(61000)
             ;  Alarm setzen, wenn Abstand (Istwert der Achse X1 - Istwert der Achse
                X2)
             ;  den kritischen Wert 4.567 unterschreitet.
```

Zyklen und Zyklenalarme

Hinweise zu Zyklen und Zyklenalarmen finden Sie in:

Literatur:

/PGZ/ Programmierhandbuch Zyklen

2.6.23 Auswertung der Daten zur Maschinenwartung

Funktion

Maschinenbetreiber erhalten die Möglichkeit, sich in Teileprogrammen, Synchronaktionen und über die BTSS-Schnittstelle auch von PLC bzw. HMI mit Systemvariablen über die Maschinennutzung zu informieren.

Abhängig von den ausgelesenen Werten können dann Wartungsmaßnahmen direkt eingeleitet oder angefordert werden.

Speicherung

Die Systemvariablen zur Maschinenwartung werden im SRAM geführt. Dadurch bleiben sie über Power On hinweg erhalten.

Hinweis

Das Signal Schmierimpuls wird im Gegensatz dazu immer dann gesetzt, wenn seit Power On eine im Maschinendatum hinterlegte Wegstrecke einer Achse überschritten ist. Siehe:

Literatur:

/FB1/ Funktionshandbuch Grundfunktionen; Diverse NC/PLC-Nahtstellensignale und Funktionen (A2), Kapitel "Signale von Achse/Spindel".

Verfügbarkeit

Die Werte zur Maschinenwartung stehen zur Verfügung, wenn das globale NCK-Maschinendatum gesetzt ist:

MD18860 $MN_MM_MAINTENANCE_MON (Aktivierung der Aufzeichnung von Wartungsdaten)

Und wenn je interessierende Achse mit achsspezifischen Maschinendaten angegeben wurde, welche Daten bereitgestellt werden sollen.

Das folgende Maschinendatum sorgt für die Aktivierung der Funktion und die Bereitstellung des Speichers für die in den achsspezifischen MD angegebenen Werte. Änderungen des folgenden Maschinendatums wirken mit Power On.

MD18860 $MC_MM_MAINTENANCE_MON

Achsspezifische Werte:

Im MD33060 $MA_MAINTENANCE_DATA kann bitcodiert angegeben werden:

Bit 0:	Gesamtverfahrweg, Gesamtverfahrzeit und die Anzahl der Verfahrvorgänge der Achse
Bit 1:	Gesamtverfahrweg, Gesamtverfahrzeit und die Anzahl der Verfahrvorgänge der Achse bei **großen Geschwindigkeiten** der Achse. Große Geschwindigkeiten sind >= 80 % der Maximalgeschwindigkeit der Achse
Bit 2:	Gesamtsumme des Rucks der Achse, Verfahrzeit mit Ruck und Anzahl der Verfahrvorgänge mit Ruck
Blt 3-15:	reserviert

Konfigurationsbeispiel

MD18860 $MN_MM_MAINTENANCE_MON = TRUE (Aktivierung der Aufzeichnung von Wartungsdaten)

MD33060 $MA_MAINTENANCE_DATA[0]=1 (Konfig. der Aufz. von Wartungsdaten)

MD33060 $MA_MAINTENANCE_DATA[1]=1

MD33060 $MA_MAINTENANCE_DATA[2]=1

… aktiviert die Systemvariablen für Gesamtverfahrweg, Gesamtverfahrzeit und die Anzahl der Verfahrvorgänge für die ersten 3 Achsen.

Systemvariablen

Folgende Systemvariablen können aus dem Teileprogramm und aus Synchronaktionen gelesen werden:

$AA_TRAVEL_DIST	Gesamtverfahrweg in mm bzw. Grad
$AA_TRAVEL_TIME	Gesamtverfahrzeit in Sekunden
$AA_TRAVEL_COUNT	Gesamtanzahl der Verfahrvorgänge
$AA_TRAVEL_DIST_HS	Gesamtverfahrweg bei großen Geschwindigkeiten in mm bzw. Grad
$AA_TRAVEL_TIME_HS	Gesamtverfahrzeit in Sekunden bei großen Geschwindigkeiten

$AA_TRAVEL_COUNT_HS	Gesamtanzahl der Verfahrvorgänge bei großen Geschwindigkeiten
$AA_JERK_TOT	Gesamtsumme des Rucks der Achse in m/s^3
$AA_JERK_TIME	Verfahrzeit der Achse mit Ruck in Sekunden
$AA_JERK_COUNT	Anzahl der Verfahrvorgänge der Achse mit Ruck

Beispiel: Weg während der Teileprogrammbearbeitung

Durch wiederholtes Auslesen können z. B. Gesamtfahrwege einer Achse innerhalb eines Teileprogrammbereiches bestimmt werden.

```
; Anfang des Bearbeitungsbereiches im Teileprogramm
R1 = $AA_TRAVEL_DIST[X]
…
…                                   ;  Ende des Bearbeitungsbereiches
R2 = $AA_TRAVEL_DIST[X]
R3 = R2 -R1                         ;  Gesamter Verfahrweg der X-Achse
                                    ;  während der Bearbeitung des
                                    ;  Bearbeitungsbereiches im Teileprogramm
```

2.7 Technologiezyklen

2.7.1 Eigenschaften von Technologiezyklen

Definition

Ein Technologiezyklus ist eine Folge von Aktionen, die sequentiell im Interpolationstakt abgearbeitet werden. Die im Kapitel "Aktionen in Synchronaktionen [Seite 27]" dargestellten Aktionen können zu Programmen zusammengefasst werden. Aus Anwendersicht handelt es sich bei diesen Programmen um Unterprogramme ohne Parameter.

Parallelität im Kanal

In einem Kanal können gleichzeitig mehrere Technologiezyklen oder Aktionen bearbeitet werden. Die Bearbeitung der Technologiezyklen und Aktionen des Kanals erfolgt parallel in einem Interpolationstakt.

Unterschiede in der Bearbeitung

Bezüglich der Bearbeitungsfolge existieren folgende Möglichkeiten:

- Mehrere Aktionen in einer Synchronaktion

  Die Aktionen werden alle gleichzeitig in dem Interpolationstakt ausgeführt, in dem die Bedingung erfüllt ist.

- Aktionen sind zu einem Technologiezyklus zusammengefasst

  Die Aktionen im Technologiezyklus werden im Interpolationstakt sequentiell abgearbeitet. Pro Interpolationstakt wird ein Satz abgearbeitet. Es müssen ein- und mehrtaktige Aktionen unterschieden werden. Ein Technologiezyklus ist dann beendet, wenn seine letzte Aktion ausgeführt ist (in der Regel nach mehreren Interpolationstakten).

Befehle wie Variablenzuweisungen werden im Technologiezyklus in einem Interpolationstakt abgearbeitet. Andere Befehle (z. B. Bewegung einer Kommandoachse, siehe Kapitel "Starten von Kommandoachsen") dauern mehrere Interpolationstakte. Ist die Funktion beendet (z. B. Genauhalt bei Positionieren einer Achse), so wird im darauf folgenden Interpolationstakt der nächste Satz ausgeführt.

Jeder Satz benötigt mindestens einen Interpolationstakt. Stehen mehrere eintaktige Aktionen in einem Satz, so werden diese in einem Interpolationstakt abgearbeitet.

Anwendung

Mit Technologiezyklen ist es beispielsweise möglich, jede Achse durch ein eigenes Achsprogramm zu bewegen.

Programmierung

In einer modalen/statischen Synchronaktion kann ein Technologiezyklus in Abhängigkeit von einer Bedingung aktiviert werden.

Für den Aufruf eines Technologiezyklus gilt der Suchpfad wie bei Unterprogrammen und Zyklen.

Das Programmende im Technologiezyklus wird mit `M02/M17/M30/RET` programmiert.

Hinweis

Ist die Bedingung erneut erfüllt, während der Technologiezyklus abgearbeitet wird, so wird er nicht von neuem gestartet. Wird der Technologiezyklus aus der Synchronaktion vom Typ WHENEVER gestartet und ist die Bedingung bei Ende des Technologiezyklus noch erfüllt, so wird der Technologiezyklus von neuem gestartet.

Beispiele

Beispiel 1: Aufruf eines Technologiezyklus

Hauptprogramm:

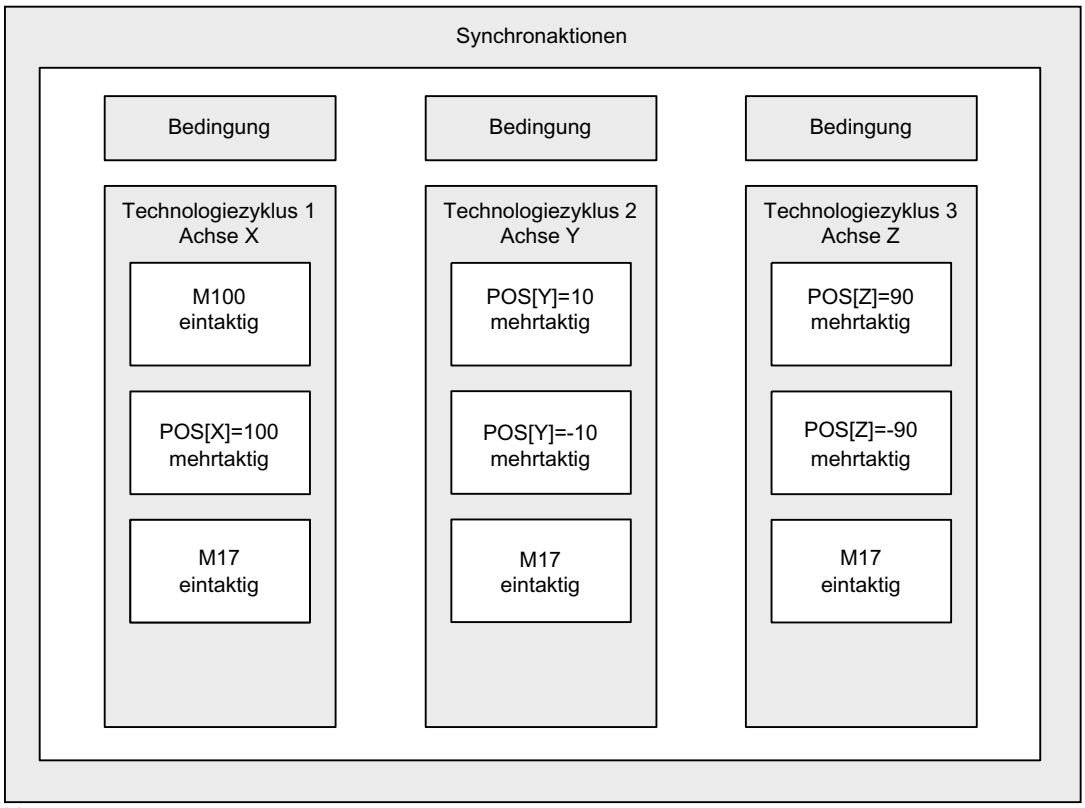
```
Programmcode                                Kommentar
...
ID=1 EVERY $AA_IM[Y]>=10 DO AX_X            ;  AX_X: Unterprogrammname für
                                               Achsprogramm für X-Achse.
...
```

Achsprogramm AX_X:

```
Programmcode
POS[X]=$R[7] FA[X]=377
$A_OUT[1]=1
POS[X]=R10
POS[X]=-90
M30
```

Beispiel 2: Koordinierte Achsbewegungen

Durch Setzen digitaler NC-Eingänge werden verschiedene Achs-Programme gestartet:



Hauptprogramm:

Programmcode

```
...
ID=1 WHEN $A_IN[1]==1 DO ACHSE_X
ID=2 WHEN $A_IN[2]==1 DO ACHSE_Y
ID=3 WHEN $A_IN[3]==1 DO ACHSE_Z
M30
```

Achsprogramm ACHSE_X:

Programmcode

```
M100
POS[X]=100
M17
```

Achsprogramm ACHSE_Y:

Programmcode

```
POS[Y]=10
POS[Y]=-10
M17
```

Achsprogramm ACHSE_Z:

Programmcode
POS[Z]=90
POS[Z]=-90
M17

2.7.2 Koordinierungen zwischen Synchronaktionen, Technologiezyklen, Teileprogramm (und PLC)

Beeinflussung von Technologiezyklen

Technologiezyklen/Synchronaktionen werden über die Identifikationsnummer der Synchronaktionen beeinflusst, in denen sie als Aktion angegeben sind.

Hinweis

Eine Synchronaktion enthält einen Technologiezyklus-Aufruf. Weitere Aktionen sind in diesem Satz nicht erlaubt. Damit besteht Eindeutigkeit zwischen beeinflusster ID-Nummer und dem zugehörigen Technologiezyklus.

Darüber hinaus stehen für die Koordinierung von Technologiezyklen folgende Schlüsselwörter zur Verfügung:

Schlüsselwort	Bedeutung	TP[1)]	SA[2)]
LOCK(ID)	Technologiezyklus sperren. Die ggf. aktive Aktion wird unterbrochen.		+
UNLOCK(ID)	Mit UNLOCK wird der Technologiezyklus an der Stelle der Unterbrechung fortgesetzt. Ein unterbrochener Positioniervorgang wird fortgesetzt.		+
RESET(ID)	Technologiezyklus abbrechen. Aktive Positioniervorgänge werden abgebrochen. Wird der Technologiezyklus neu gestartet, so beginnt seine Bearbeitung mit dem 1. Satz im Technologiezyklus. Je nach Synchronaktionstyp werden die Aktionen nach erneutem Eintreten der Bedingung wieder ausgeführt. Bereits ausgeführte Synchronaktionen vom WHEN-Typ werden nach RESET nicht mehr bearbeitet.		+
CANCEL(ID)	Die Synchronaktion wird gelöscht.	+	

1) Aufruf zulässig im Teileprogramm

2) Aufruf zulässig in Synchronaktion/Technologiezyklus

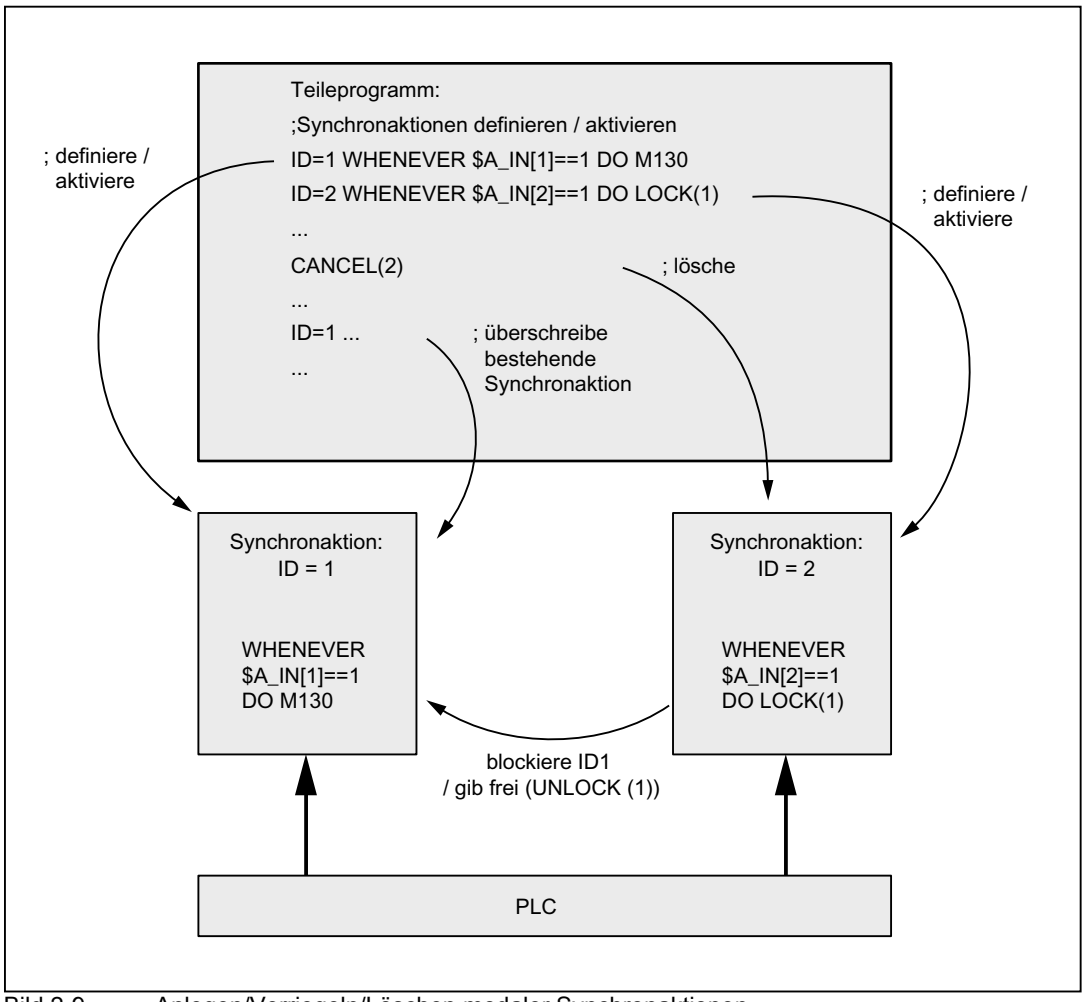


Bild 2-9 Anlegen/Verriegeln/Löschen modaler Synchronaktionen

Hinweis

`LOCK(ID), UNLOCK(ID)` durch PLC siehe Kapitel "Beeinflussung von PLC".

2.8 Beeinflussung und Schutz von Synchronaktionen

2.8.1 Beeinflussung durch die PLC

Funktion

Modale Synchronaktionen (ID, IDS) können von PLC verriegelt bzw. freigegeben werden.

- Sperrung aller modalen Synchronaktionen
- Gezielte Sperrung einzelner Synchronaktionen

Einflussbereich

Die PLC kann auf maximal die ersten 64 modalen Synchronaktionen mit Sperren Einfluss nehmen (ID, IDS 1-64). Die durch PLC **sperrbaren** Synchronaktionen sind in einem 64 Bit großen Feld der fogenden Nahtstelle durch die NC mit 1 gekennzeichnet:

DB21, … DBB308-315

Geschützte Synchronaktionen sind nie als sperrbar gekennzeichnet. Siehe Kapitel 2.6.2 "Geschützte Synchronaktionen".

Alle Synchronaktionen sperren

Durch das PLC-Anwendungsprogramm können alle modalen Synchronaktionen, die in der NC bereits definiert und gespeichert sind, von der Aktivierung ausgeschlossen werden, durch Setzen der NC/PLC-Nahtstelle:

DB21, … DBX1.2 (Synchronaktion aus)

Eine Ausnahme bilden geschützte Synchronaktionen siehe Kap. "Geschützte Synchronaktionen".

Die pauschale Sperrung wird durch PLC wieder aufgehoben, durch auf 0 Setzen der NC/PLC-Nahtstelle:

DB21, … DBX1.2.

Benutzung gezielte Sperren

Für die ersten 64 IDs (1-64) ist je ein Bit in der PLC-Nahtstelle reserviert:

DB21, … DBX300.0 (Synchronaktionen sperren Nr. 1)

bis

DB21, … DBX307.7 (Synchronaktionen sperren Nr. 63)

Standardmäßig sind die Funktionen freigegeben (Bits = 0). Durch Setzen des zugeordneten Bits werden die Auswertung der Bedingung und die Ausführung der dazugehörigen Funktion im NCK verriegelt.

Aufhebung gezielter Sperren

Eine zuvor gesperrte Synchronaktion wird wieder von PLC freigegeben, durch Setzen des, der ID-, IDS-Nummer entsprechenden Bits auf 0 in der Nahtstelle:

DB21, … DBX300.0 (Synchronaktionen sperren Nr. 1)

bis

DB21, … DBX307.7 (Synchronaktionen sperren Nr. 64)

Aktualisieren der gezielten Sperren

Wenn das PLC-Anwenderprogramm im Bereich von DB21-30,DBB 300. Bit 0 bis DB21-30 BB307 Bit 7 Änderungen vorgenommen hat, muss es diese aktivieren mit:

DB21, … DBX280.1

Literatur

Listenhandbuch 2, Kapitel PLC-Anwendernahtstelle

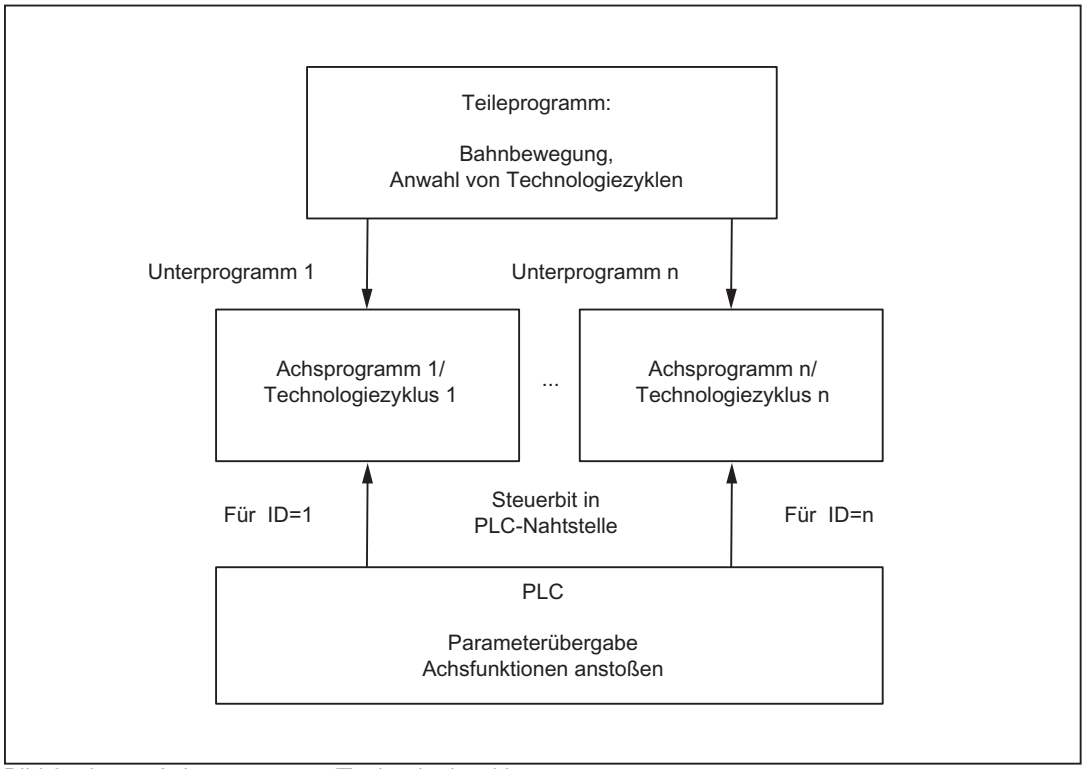


Bild 2-10 Achsprogramme/Technologiezyklen

Lesen / Schreiben von PLC-Daten

Das Lesen / Schreiben von PLC-Daten kann auch aus dem Teileprogramm mit Parameterübergabe zwischen NCK und PLC über VDI-Nahtstelle erfolgen. Dies ist eine Option: PLC-Variablen.

Literatur:

/FB1/ Funktionshandbuch Grundfunktionen; PLC-Grundprogramm (P3)

Die Parameter sind auch aus Synchronaktionen zugänglich. Damit ist es möglich, vor Anstoß einer Achs-Funktion von PLC Daten zur Parametrierung an NCK zu geben. Die anzusprechenden Systemvariablen finden Sie in:

Literatur:

/PGA/ Listenhandbuch Systemvariablen

2.8.2 Geschützte Synchronaktionen

Globaler Schutz

Es kann ein Bereich von schreibgeschützten Synchronaktionen festgelegt werden über das Maschinendatum:

MD11500 $MN_PREVENT_SYNACT_LOCK (Geschützte Synchronaktionen)

Synchronaktionen mit ID-Nummern, die im geschützten Bereich liegen, können, wenn sie einmal definiert sind, **nicht** mehr:

- überschrieben,
- gelöscht `(CANCEL)` oder
- gesperrt `(LOCK)` werden.

Geschützte Synchronaktionen können auch durch PLC nicht gesperrt werden. Sie werden der PLC an der Nahtstelle als nicht sperrbar angezeigt. Vergleiche Kap. "Beeinflussung von PLC".

Hinweis

Die Funktionalität wird auch für Safety Integrated Systeme benutzt.

Anwendungen

Vom Maschinenhersteller definierte Reaktionen auf bestimmte Zustände sollen vom Endkunden nicht mehr beeinflusst werden können.

Die Inbetriebnahme beim Maschinenhersteller erfolgt noch ohne Schutz. Damit kann die Verknüpfungslogik definiert und getestet werden. Vor Auslieferung der Maschine erklärt der Maschinenhersteller den von ihm verwendeten Bereich von Synchronaktionen als geschützt. Damit ist es dem Endkunden nicht mehr möglich, eigene Synchronaktionen in diesem Bereich zu definieren.

Notation des Maschinendatums: MD11500 $MN_PREVENT_SYNACT_LOCK

```
MD11500 $MN_$MN_PREVENT_SYNACT_LOCK[0]=   ;  i Nummer der 1. zu sperrenden ID
i
MD11500 $MN_PREVENT_SYNACT_LOCK[1]= j     ;  j Nummer der letzten zu sperrenden ID
```

ID i und j können auch vertauscht angegeben werden.

Mit i = 0 und j = 0 gibt es keinen geschützten Bereich.

Kanalspezifischer Schutz

Es kann ein Bereich von schreibgeschützten Synchronaktionen für den Kanal festgelegt werden über das kanalspezifische Maschinendatum:

MD21240 $MC_PREVENT_SYNACT_LOCK_CHAN (Geschützte Synchronaktionen)

Synchronaktionen mit ID-Nummern, die im geschützten Bereich liegen, können, wenn sie einmal definiert sind, **nicht** mehr:

- überschrieben,
- gelöscht (CANCEL) oder
- gesperrt (LOCK) werden.

Geschützte Synchronaktionen können auch durch PLC nicht gesperrt werden. Sie werden der PLC an der Nahtstelle als nicht sperrbar angezeigt. Vergleiche Kap. "Beeinflussung von PLC".

Anwendung

s. o.

Notation des MD21240 $MC_PREVENT_SYNACT_LOCK_CHAN

```
CHANDATA(C)                                    ;  mit C Kanalnummer
MD21240 $MC_PREVENT_SYNACT_LOCK_CHAN[0]=k      ;  k Nummer der 1. für den Kanal zu
                                                  sperrenden ID
MD21240 $MC_PREVENT_SYNACT_LOCK_CHAN[1]=l      ;  l Nummer der letzten für den Kanal zu
                                                  sperrend. ID
```

k und l können auch vertauscht angegeben werden.

Mit k = 0 und l = 0 gibt es keinen geschützten Bereich.

Mit k = -1 und l = -1 wird angegeben, dass für den Kanal der globale Bereich von geschützten Synchronaktionen gelten soll und mit dem folgenden Maschinendatum festgelegt:

MD11500 MN_PREVENT_SYNACT_LOCK (Geschützte Synchronaktionen)

Hinweis

Während der Erstellung von geschützten statischen Synchronaktionen sollte der Schutz aufgehoben sein, da sonst bei jeder Änderung Power On notwendig ist, um die Logik neu definieren zu können.

Die Wirksamkeit der Sperren ist identisch, unabhängig davon, ob sie angegeben wurden als:

- globale Sperren oder
- kanalspezifische Sperren.

Beispiel

In einem System mit 2 Kanälen sollen Synchronaktionen wie folgt geschützt werden:

Im 1. Kanal sollen die IDs 20 bis 30 und im Kanal 2 sollen die IDs 25 bis 35 geschützt werden. Es wird globale und kanalspezifische Angabe gemischt verwendet.

```
MD11500 $MN_PREVENT_SYNACT_LOCK[0] = 25          ;  globale Angabe
MD11500 $MN_PREVENT_SYNACT_LOCK[1] = 35          ;  globale Angabe
CHANDATA(1)
MD21240 $MC_PREVENT_SYNACT_LOCK_CHAN[0] = 20
           ;  im 1. Kanal wirkt nur das kanalspez. MD(1. zu schützende ID-Nummer)
MD21240 $MC_PREVENT_SYNACT_LOCK_CHAN1] = 30
           ;  im 1. Kanal wirkt nur das kanalspez. MD(letzte zu schützende ID-
              Nummer)
CHANDATA(2)
MD21240 $MC_PREVENT_SYNACT_LOCK_CHAN[0] = -1
           ;  im 2. Kanal wirkt das globale Maschinendatum MD11500
           ;  $MN_PREVENT_SYNACT_LOCK!
MD21240 $MC_PREVENT_SYNACT_LOCK_CHAN[1] = -1
...
```

2.9 Steuerungsverhalten in bestimmten Betriebszuständen

2.9.1 Power On

Bei Power On sind keine Synchronaktionen aktiv. Statische Synchronaktionen, die sofort nach Power On aktiv sein sollen, müssen in einem von PLC gestarteten ASUP aktiviert werden.

Literatur:

/FB1/ Funktionshandbuch Grundfunktionen; PLC-Grundprogramm

/FB1/ Funktionshandbuch Grundfunktionen; BAG, Kanal, Programmbetrieb (K1)

Voraussetzung dafür ist die Funktionalität: ASUP in allen Betriebsarten.

Beispiele:

- AC-Regelung
- Safety Integrated, Verknüpfungslogik durch Synchronaktionen formuliert

2.9.2 RESET

Bei Positionierachsbewegungen

Mit NC-Reset werden alle durch Synchronaktionen gestarteten Positionierbewegungen abgebrochen. Aktive Technologiezyklen werden zurückgesetzt.

ID

Programmlokale Synchronaktionen (mit ID=... programmiert) werden mit NC-Reset abgewählt.

IDS

Statische Synchronaktionen (mit IDS = ... programmiert) bleiben über NC-Reset hinaus erhalten. Aus diesen können nach NC-Reset wieder Bewegungen gestartet werden.

Weitere Reaktionen, abhängig von Aktionen

`RESET` Fortsetzung

Synchronaktion/ Technologiezyklus	modale und satzweise aktive Aktion wird abgebrochen, Synchronaktionen werden gelöscht	statisch (IDS) aktive Aktion wird abgebrochen, Technologiezyklus wird zurückgesetzt
Achse/ positionierende Spindel	Bewegung wird abgebrochen	Bewegung wird abgebrochen

drehzahlgeregelte Spindel	MD35040 $MA_SPIND_ACTIVE_AFTER_RESET== TRUE: Spindel bleibt aktiv MD35040 $MA_SPIND_ACTIVE_AFTER_RESET==FALSE: Spindel stoppt.	MD35040 $MA_SPIND_ACTIVE_AFTER_RESET== TRUE: Spindel bleibt aktiv MD35040 $MA_SPIND_ACTIVE_AFTER_RESET==FALSE: Spindel stoppt.
Leitwertkopplung	MD20110 $MC_RESET_MODE_MASK, Bit13 == 1: Leitwertkopplung bleibt aktiv MD20110 $MC_RESET_MODE_MASK, Bit13 == 0: Leitwertkopplung wird aufgelöst	MD20110 $MC_RESET_MODE_MASK, Bit13 == 1: Leitwertkopplung bleibt aktiv MD20110 $MC_RESET_MODE_MASK, Bit13 == 0: Leitwertkopplung wird aufgelöst
Messvorgänge	aus Synchronaktionen gestartete Messvorgänge werden abgebrochen	aus statischen Synchronaktionen gestartete Messvorgänge werden abgebrochen

2.9.3 NC-STOP

Bewegungsstart aus statischen Synchronaktionen

Aus statischen Synchronaktionen gestartete Bewegungen bleiben bei `NC-STOP` aktiv.

Verhalten einer Kommandoachse ab SW 6.3:

Hinweis

Ab SW 6.3 ist es möglich, eine Kommandoachse, die über statische Synchronaktion gestartet wurde, zu einer PLC-kontrollierten Achse umzuwandeln. VDI-Nahtstelle:

DB31, ... DBX28.7 (PLC kontrolliert Achse/P5)

Eine solche Achse wird durch `NC-STOP` **nicht** mehr angehalten, sondern nur durch einen axialen `STOP`.

Bewegungsstart aus satzweisen und modalen Synchronaktionen

Aus satzweisen und modalen Synchronaktionen gestartete Achsbewegungen werden unterbrochen und mit `NC-Start` fortgesetzt. Drehzahlgeregelte Spindeln bleiben aktiv.

Die zum aktiven Satz gehörenden Synchronaktionen bleiben weiter aktiv.

Beispiel:

Ausgang setzen: `... DO $A_OUT[1] = 1`

2.9.4 Betriebsartenwechsel

Es wird unterschieden zwischen programmlokalen und statischen Synchronaktionen. Mit dem Schlüsselwort **IDS** aktivierte Synchronaktionen bleiben über Betriebsartenwechsel hinweg aktiv. Alle übrigen Synchronaktionen werden bei Betriebsartenwechsel inaktiv und mit dem Repositionieren bei Wechsel nach AUTO-Betrieb wieder aktiv.

Beispiel:

```
N10     WHEN $A_IN[1] == 1 DO DELDTG
N20     G1    X10 Y 200 F150 POS[U]=350
```

In Satz `N20` wird gestoppt. Es erfolgt Betriebsartenwechsel nach JOG. War Restweglöschen vor der Unterbrechung noch nicht aktiv, so ist die im Satz N10 programmierte Synchronaktion nach der Rückkehr in Betriebsart AUTO und Fortsetzen des Programms weiter aktiv.

2.9.5 Programmende

Statische Synchronaktionen bleiben über Programmende hinaus aktiv. Satzweise und modale Synchronaktionen werden abgebrochen. Im `M30`-Satz wirken statische und modale Synchronaktionen weiter. Sie können vor `M30` mit `CANCEL` abgebrochen werden. Die mit `FCTDEF` programmierten Polynomkoeffizienten wirken über Programmende hinweg.

2.9.6 Verhalten der aktiven Aktionen bei Programmende und Betriebsartenwechsel

Siehe Kapitel 2.7.4 "Betriebsartenwechsel" und Kapitel 2.7.5 "Programmende".

Synchronaktion/ Technologiezyklus	modale und satzweise werden abgebrochen	statisch (IDS) bleiben erhalten
Achse/positionierende Spindel	M30 wird verzögert, bis die Achse/Spindel steht.	Bewegung läuft weiter
drehzahlgeregelte Spindel	Programmende: MD35040 $MA_SPIND_ACTIVE_AFTER_RESET== TRUE: Spindel bleibt aktiv MD35040 $MA_SPIND_ACTIVE_AFTER_RESET==FALSE: Spindel stoppt. Bei Betriebsartenwechsel bleibt Spindel aktiv	Spindel bleibt aktiv
Leitwertkopplung	MD20110 $MC_RESET_MODE_MASK, Bit13 == 1: Leitwertkopplung bleibt aktiv MD20110 $MC_RESET_MODE_MASK, Bit13 == 0: Leitwertkopplung wird aufgelöst	aus statischer Synchronaktion gestartete Kopplung bleibt erhalten
Messvorgänge	aus Synchronaktionen gestartete Messvorgänge werden abgebrochen	aus statischen Synchronaktionen gestartete Messvorgänge bleiben aktiv

2.9.7 Satzsuchlauf

Allgemein

Die im Satzsuchlauf interpretierten Synchronaktionen des Programms werden aufgesammelt. Die Bedingungen werden jedoch nicht ausgewertet. Aktionen werden nicht ausgeführt. Die Bearbeitung der Synchronaktionen beginnt erst mit NC-Start.

IDS

Mit dem Schlüsselwort IDS programmierte Synchronaktionen, die bereits aktiv sind, wirken auch während des Satzsuchlaufs.

Polynomkoeffizienten

Die mit `FCTDEF` programmierten Polynomkoeffizienten werden bei Satzsuchlauf **mit Berechnung** aufgesammelt, d. h. in die Systemvariablen geschrieben.

2.9.8 Programmunterbrechung durch ASUP

ASUP-Anfang

Modale und statische Bewegungssynchronaktionen bleiben erhalten und sind auch im asynchronen Unterprogramm wirksam.

ASUP-Ende

Wird das asynchrone Unterprogramm nicht mit `REPOS` fortgesetzt, so wirken die im asynchronen Unterpogramm geänderten modalen und statischen Bewegungssynchronaktionen im Hauptprogramm weiter.

Aus Synchronaktionen gestartete Positionierbewegungen verhalten sich wie bei Betriebsartenwechsel:

Aus satzweisen und modalen Aktionen gestartete Bewegungen werden gestoppt und evtl. mit `REPOS` fortgesetzt. Aus statischen Synchronaktionen gestartete Bewegungen laufen weiter.

2.9.9 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.

2.9.10 Verhalten bei Alarmen

Über Synchronaktionen gestartete Achs- und Spindelbewegungen werden abgebremst, wenn ein Alarm mit Bewegungsstopp aktiv ist. Alle weiteren Aktionen (wie z. B. Ausgang setzen) werden weiter ausgeführt.

Löst eine Synchronaktion selbst einen Alarm aus, dann kommt es zum Bearbeitungsabbruch und die nachfolgenden Aktionen dieser Synchronaktion werden nicht mehr ausgeführt. Ist die Synchronaktion modal wirksam, wird sie im nächsten Interpolationstaktakt nicht weiter bearbeitet. Der Alarm wird also nur einmal abgesetzt. Alle weiteren Synchronaktionen werden weiter bearbeitet.

Alarme, die Interpreterstopp als Alarmreaktion haben, wirken erst nach Abarbeiten der vordekodierten Sätze.

Löst ein Technologiezyklus einen Alarm mit Bewegungsstopp aus, so wird der Technologiezyklus nicht weiter bearbeitet.

2.10 Projektierung

2.10.1 Projektierbarkeit

Anzahl Synchronaktionselemente

Die Anzahl der programmierbaren Synchronaktionssätze hängt nur von der projektierbaren Anzahl von Synchronaktionselementen ab. Die Anzahl der Speicherelemente von Bewegungssynchronaktionen (Synchronaktionselementen) wird festgelegt über das Maschinendatum:

MD28250 $MC_MM_NUM_SYNC_ELEMENTS (Anzahl Elemente für Ausdrücke in Synchronaktionen)

Die Festlegung ist unabhängig von der Anzahl der steuerungsintern verfügbaren Satzanzahl. Damit ist die Komplexität der in Echtzeit ausgewerteten Ausdrücke sowie die Anzahl der Aktionen flexibel einstellbar.

Verwendung der Elemente

Je **ein** Synchronaktionselement wird benötigt für:

- einen Vergleichsausdruck in der Bedingung
- eine elementare Aktion
- den Synchronaktionssatz.

Beispiel:

Für den nachfolgenden Synchronaktionssatz werden insgesamt vier Elemente verbraucht.

```
WHENEVER ($AA_IM[x] > 10.5) OR ($A_IN[1]==1) DO
|_______|     |_________________|     |_______________|
Element 1      Element 2               Element 3
$AC_PARAM[0]=$AA_im[y]+1
|_____________________|
Element 4
```

Der Standardwert des folgenden Maschinendatums ist so eingestellt, dass die bis SW-Stand 3 fest vorgegebene Anzahl von max. 16 Synchronaktionen aktiviert werden kann:

MD28250 $MC_MM_NUM_SYNC_ELEMENTS (Anzahl Elemente für Ausdrücke in Synchronaktionen)

Hinweis

Programmiert der Anwender keine Synchronaktionen, so kann er den Wert auf 0 gesetzt werden im Maschinendatum:

MD28250 $MC_MM_NUM_SYNC_ELEMENTS

So können ca. 16 kByte an DRAM Speicher eingespart werden.

Anzeige

Mit der Statusanzeige für Synchronaktionen (siehe Kapitel 2.9 "Diagnose nur mit HMI Adv.") lässt sich die Auslastung des Speichers für Synchronaktionen verfolgen oder aus Synchronaktionen über die variable "AC_SYNA_MEM lesen.

Alarm

Gehen die Elemente im Programmablauf aus, so wird ein Alarm abgesetzt. Der Anwender kann daraufhin die Anzahl der Synchronaktionselemente erhöhen oder sein Programm entsprechend abändern.

Anzahl FCTDEF-Funktionen

Die Anzahl der Programmierbaren `FCTDEF`-Funktionen pro Satz wird über das folgende Maschinendatum projektiert:

MD28252 $MC_MM_NUM_FCTDEF_ELEMENTS (Anzahl der FCTDEF-Elemente)

Der Standardwert liegt für alle Steuerungstypen bei 3. Den steuerungsabhängigen Maximalwert finden Sie in:

Literatur:

/LIS1/ Listen (Buch1); MD- / SD-Listen

Interpolationstakt

Bei großer Anzahl von Synchronaktionen erhöht sich der Zeitbedarf für die Interpolationsebene. Der Interpolationstakt muss ggf. bei der Inbetriebnahme dem Bedarf entsprechend verlängert werden.

Richtwerte IPO-Takt Verlängerung

Als Orientierungshilfe werden einzelne Zeiten für Operationen innerhalb von Synchronaktionen (gemessen auf 840D mit NCU 573.x) angegeben:

Für andere Steuerungstypen sind Abweichungen möglich.

NC-Sprache	**Zeitbedarf**	
	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**
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**
Positionierachs-Bewegung starten: WHEN TRUE DO **POS[z]=10**	83 µs	**~73 µs**

2.11 Diagnose (nur mit 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 Interpolationstakt-Raster aufgezeichnet werden. (Hauptlaufvariablen protokollieren)

Diese Funktionalität ist in der Bedienoberfläche wie folgt strukturiert:

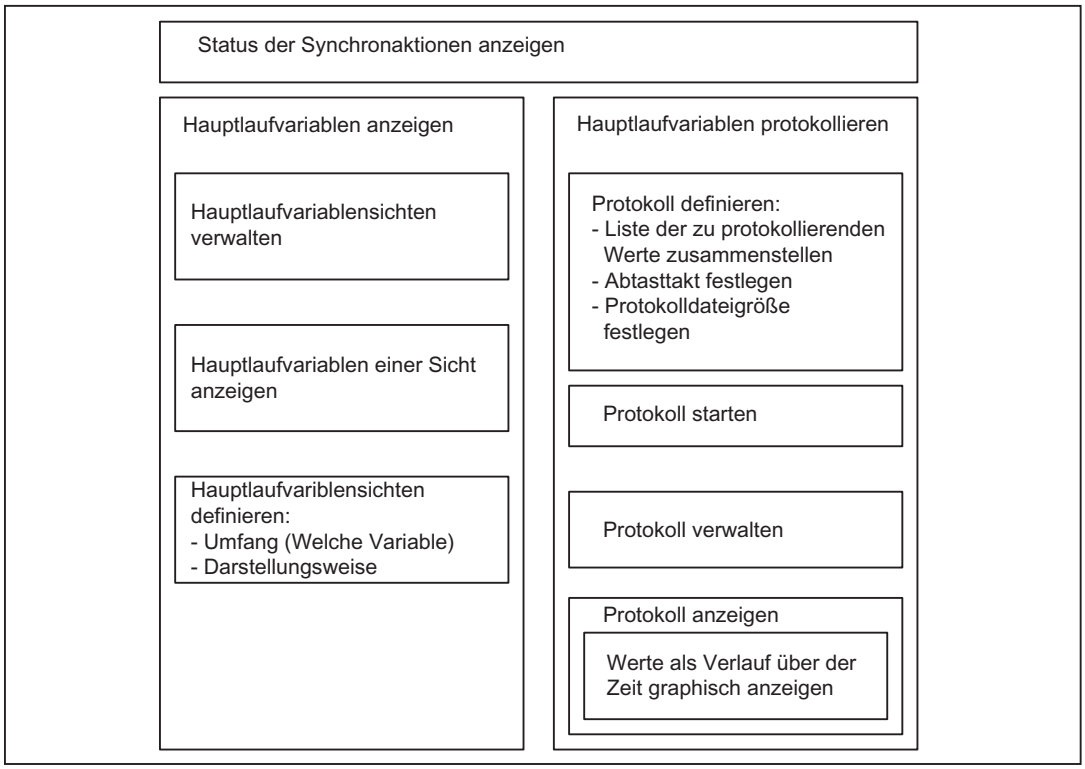


Bild 2-11 Funktionalität der Testmittel für Synchronaktionen

Die Beschreibung der Bedienung dieser Funktionen finden Sie in:

Literatur:

/BAD/ Bedienungshandbuch HMI Advanced.

2.11.1 Status der Synchronaktionen anzeigen

Statusbild

Das Statusbild zeigt an:

- Den aktuellen Ausschnitt des angewählten Programms

Alle programmierten Synchronaktionen nach:

- Zeilennummer
- Kennzeichen der Synchronaktionsart
- ID-Nummer der Synchronaktion (bei modalen Synchronaktionen)
- Status

Synchronaktionsart

Es werden unterschieden:

ID	Modale Synchronaktion
IDS	Statische modale Synchronaktion
	Satzweise Synchronaktion für den nächsten ausführbaren Satz (nur im AUTOMATIK-Betrieb)

Status

Unter Status können auftreten:

Keine Angabe:	Bedingung wird im Interpolationstakt überprüft
gesperrt	Für die Synchronaktion wurde `LOCK` gesetzt
aktiv	Die Aktion läuft gerade ab. Besteht die Aktion aus einem Technologiezyklus, so wird zusätzlich die aktuelle Zeilennummer in diesem angezeigt.

Vollständige Synchronaktionen

Durch eine Suchfunktion kann zu jeder angezeigten Synchronaktion die ursprünglich programmierte Zeile in der NC-Sprache angezeigt werden.

2.11.2 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.

2.11.3 Hauptlaufvariablen protokollieren

Ausgangssituation

Die genaue Verfolgung der Abläufe in Synchronaktionen erfordert die Beobachtung der Zustände im Interpolationstakt.

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.

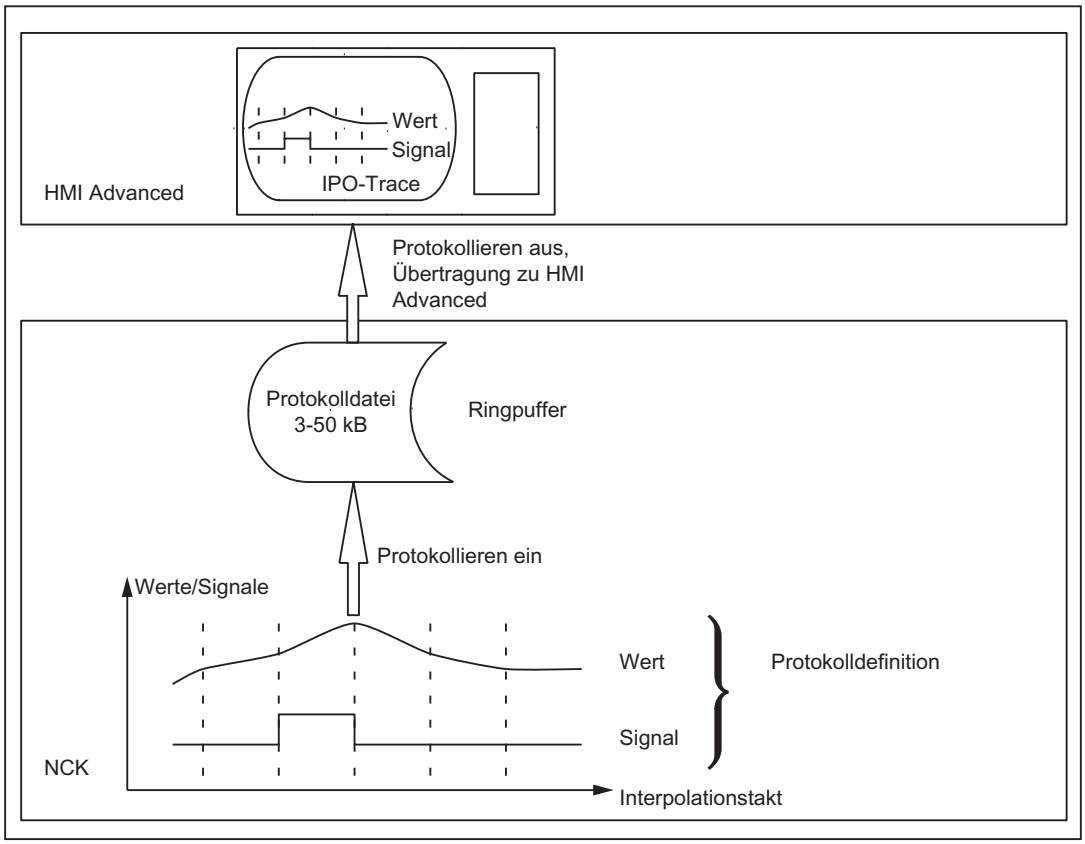


Bild 2-12 Schematischer Ablauf Hauptlaufvariablen protokollieren

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 Interpolationstaktes 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.

Randbedingungen 3

Verfügbarkeit / Leistungsumfang

Die möglichen Leistungen des Funktionspakets Synchronaktionen hängen ab von:

- dem Typ der SINUMERIK-Steuerung:
  - HW
  - SW (Exportvariante/Standardvariante)
- der Verfügbarkeit der durch "Aktionen" auslösbaren Funktionen:
  - immer vorhandene Funktionen
  - als Option zu beziehende Funktionen

Die Leistungen der Steuerungen und ihrer Varianten und die als Optionen verfügbaren Funktionen sind beschrieben in den SW-Stand-spezifischen Katalogen:
Literatur:
Katalog NC60 und NC61

Darüber hinaus hängen die Funktionen der Synchronaktionen von der Liste der aus Synchronaktionen lesbaren/änderbaren Systemvariablen einschließlich Maschinen- und Settingdaten ab.

Die für einen bestimmten SW-Stand nutzbaren Systemdaten sind beschrieben in:
Literatur:
Handbuch der Systemvariablen
Listen (Buch1)

Erweiterungen

Das Funktionspaket Synchronaktionen wurde im Laufe der Entwicklung kontinuierlich erweitert. In der folgenden Liste sind die einzelnen Erweiterungen zusammengefasst (mit den aktuellsten Erweiterungen am Ende der Liste):

- Diagnosemöglichkeiten für Synchronaktionen
- Verfügbarkeit zusätzlicher Hauptlaufvariablen
- Komplexe Bedingungen in Synchronaktionen
  - Grundrechenarten
  - Funktionen
  - Indizierung mit Hauptlaufvariablen
  - Zugriff auf Settingdaten und Maschinendaten
  - logische Operatoren
- Projektierbarkeit
  - Anzahl gleichzeitig aktiver Synchronaktionen
  - Anzahl spezieller Variablen für die Synchronaktionen
- Kommandoachsen/Achsprogramme/Technologiezyklen aus Synchronaktionen aktivieren

- `PRESET` aus Synchronaktionen
- Kopplungen und Mitschleppen aus Synchronaktionen
  - Einschalten
  - Ausschalten
  - Parametrieren
- Messfunktionen benutzen aus Synchronaktionen
- SW-Nocken
  - Umdefinieren Position
  - Umdefinieren Vorhaltzeiten
- Restweglöschen ohne Vorlaufstopp
- Statische Synchronaktionen (andere Betriebsarten als AUTO möglich)
- Synchronaktionen:
  - schützen gegen Überschreiben und Löschen
  - anhalten, fortsetzen, löschen
  - Technologiezyklen rücksetzen
  - von PLC parametrieren, aktivieren, sperren
- Überlagerte Bewegung/Abstandsregelung verfeinert
- Kanalkoordination aus Synchronaktionen
- ASUP starten aus Synchronaktionen
- Hilfsfunktionsausgabe satzunabhängig
- alle erforderlichen Leistungen für Safety Integrated zur Formulierung der erforderlichen sicherheitsgerichteten logischen Verknüpfungen, geschützt gegen Veränderungen.
- 16 Synchronaktionen sind in der Grundausführung enthalten
- Für PLC gekennzeichnete sperrbare Synchronaktionen
- Verfügbarkeit zusätzlicher Hauptlaufvariablen
- Zugriff auf PLC-E/A (Option)
- Mit der Option "Synchronaktionen Stufe 2" sind 255 parallele Synchronaktionen pro Kanal möglich.
- Über das Programmende hinaus und in allen Betriebsarten wirkende statische Synchronaktionen IDS sind mit der Option "Betriebsartübergreifende Aktionen, ASUPs und Synchronaktionen" möglich.
- Online-Berechnungen und Online-Werkzeugkorrekturen
- Achstausch über Synchronaktionen und im Technologiezyklus
- Anlegen von Koppelmodulen für die Generische Kopplung
- Verfügbarkeit von 3-dimensionale Variablen (GUD-, LUD- und Systemvariablen)

Signalbeschreibungen

4

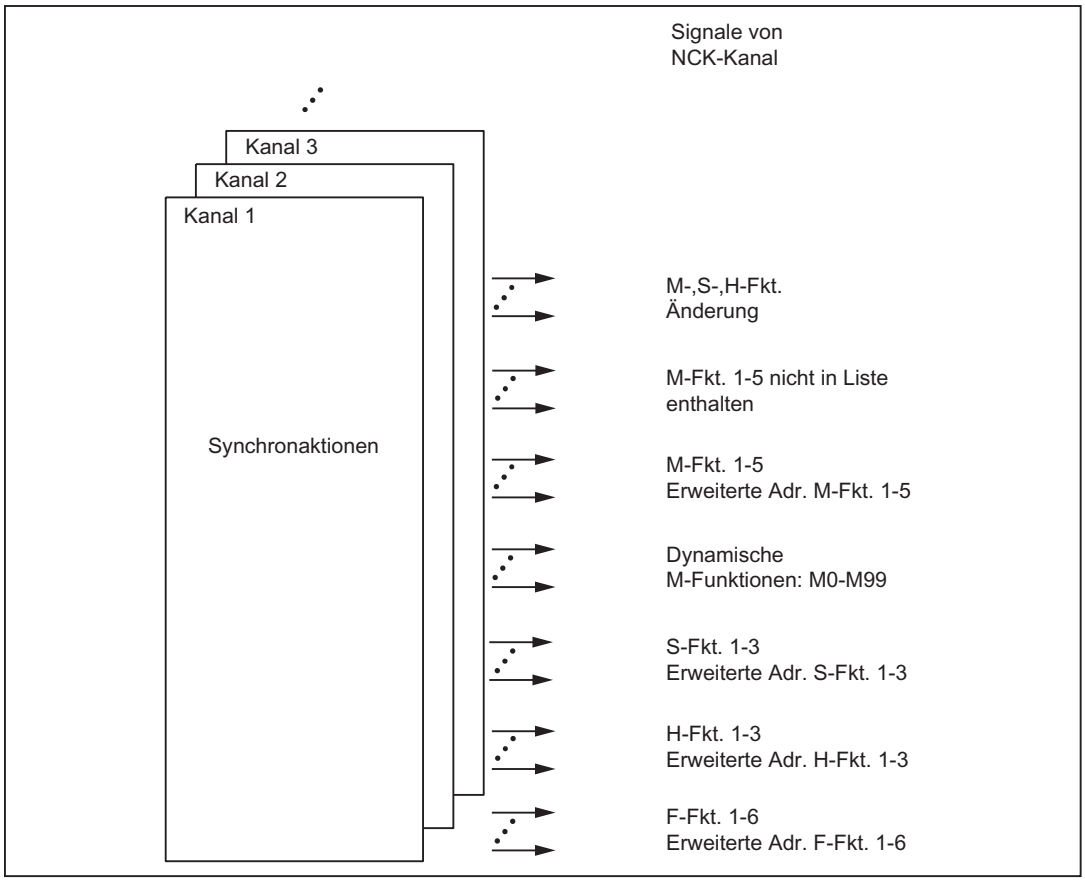


Bild 4-1 NC/PLC–Nahtstellensignale für Synchronaktionen

Zu den durch Hilfsfunktionsausgabe aus Synchronaktionen erzeugten Signalen siehe:
Literatur:
Funktionshandbuch Grundfunktionen; Hilfsfunktionsausgaben an PLC (H2)

Zu sperrende Synchronaktionen

Mit den folgenden Signalen fordert das PLC-Anwendungsprogramm die Sperrung der zugeordneten Synchronaktionen an:

DB21, … DBX300.0 (Synchronaktion Nr. 1 sperren)

...

DB21, … DBX307.7 (Synchronaktionen Nr. 64 sperren)

Dabei entspricht DBX300.0 der ersten modalen Synchronaktion (ID=1/IDS=1) und DBX307.7 der 64. modalen Synchronaktion (ID=64/IDS=64).

Hinweis

Nur die Instanz (NCK oder PLC), die eine Sperre setzt, kann auch wieder die Sperre aufheben.

Sperrbare Synchronaktionen

Mit den folgenden Signalen zeigt der Kanal dem PLC-Anwendungsprogramm an, welche Synchronaktionen durch PLC gesperrt werden dürfen:

DB21, … DBX308.0 (Synchronaktion Nr. 1 sperrbar)

...

DB21, … DBX315.7 (Synchronaktion Nr. 64 sperrbar)

Dabei entspricht DBX308.0 der ersten modalen Synchronaktion (ID=1/IDS=1) und DBX315.7 der 64. modalen Synchronaktion (ID=64/IDS=64).

Alle Synchronaktionen sperren

Alle modalen/statischen Synchronaktionen, soweit sie nicht geschützt sind, werden gesperrt durch das globale Signal:

DB21, … DBX1.2 (Synchronaktion aus)

Markierte Synchronaktionen sperren

Die in DBB308 bis DBB315 als sperrbar und in DBB300 bis DBB307 als zu sperrend markierten Synchronaktionen werden gesperrt durch das Signal:

DB21, … DBX280.1 (Synchronaktionen sperren)

Beispiele

5

5.1 Beispiele für Bedingungen in Synchronaktionen

Bahnabstand vom Satzende

Axialer Abstand 10 mm oder weniger vom Satzende (Werkstück-Koordinatensystem):

```
... WHEN $AC_DTEW <= 10 DO ...
G1 X10 Y20
```

Achsabstand vom Bahnende

```
 ... WHEN $AA_DTEW[X]<= 10 DO ...
POS[X]= 10
```

Bahnabstand vom Satzanfang

Bahnweg 20 mm oder mehr nach Satzanfang im Basis-Koordinatensystem:

```
...WHEN $AC_PLTBB >= 20 DO ...
```

Bedingung mit Funktion im Vergleich

Istwert für Achse y im MKS größer als 10-mal Sinus des Wertes in `R10`:

```
... WHEN $AA_IM[y] > 10*SIN (R10) DO...
```

Schrittweises Positionieren

Jedes Mal, wenn der Eingang 1 gesetzt wird, wird Achse um einen Schritt weiter positioniert. Der Eingang muss dann wieder zurückgesetzt werden, damit Neustart möglich ist.

```
G91
EVERY $A_IN[1]==1 DO POS[X]= 10
```

In jedem Interpolationstakt OVR

Um eine Bahnbewegung gezielt festzuhalten, bis ein erwartetes Signal eintrifft, muss $AC_OVR in jedem Interpolationstakt (Schlüsselwort `WHENEVER`) auf Null gesetzt werden.

```
WHENEVER $A_IN[1]==0 DO $AC_OVR= 0
```

Weitere Möglichkeiten

Die Liste der in Synchronaktionen lesbaren Systemvariablen erschließt die volle Menge der in Bedingungen von Synchronaktionen auswertbaren Größen.

Literatur:

/PGA1/ Listenhandbuch Systemvariablen

5.2 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 mit ihrem gewöhnlichen Namen angesprochen.

Beispiel: Pendeln aus Synchronaktionen

```
NC-Sprache                                          Kommentar
N610 ID=1 WHENEVER $AA_IM[Z]>$SA_OSCILL_REVERSE_POS1[Z]
DO $AC_MARKER[1]=0
                                                    ;  Immer wenn die aktuelle Position der
                                                       Pendelachse
                                                    ;  im Maschinenkoordinatensystem
                                                    ;  kleiner  als der Beginn des Umkehrbereichs 2
                                                               ist,
                                                    ;  dann     setze den axiale Override der
                                                    ;           Zustellachse auf 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 Maschinenkoordinatensystem
                                                    ;           gleich der Umkehrposition 1 ist,
                                                    ;  dann     setze den axialen Override der
                                                    ;           Pendelachse auf 0
                                                    ;  und      setze den axialen Override der
                                                    ;           Zustellachse auf 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 der Restweg der Teilzustellung
                                                    ;  gleich   0 ist,
                                                    ;  dann     setze den axialen Override der
                                                               Pendel-
                                                    ;           achse auf 100% (damit wird die vorher-
                                                    ;           gehende Synchronaktion aufgehoben!)
```

NC-Sprache	**Kommentar**
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 die aktuelle Position der Pendelachse im
	; Werkstückkoordinatensystem
	; gleich der Umkehrposition 1 ist,
	; dann setze den axialen Override der
	; Pendelachse auf 100%
	; und setze den axialen Override der
	; Zustellachse auf 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 die aktuelle Position der Pendelachse
	; im Maschinenkoordinatensystem
	; kleiner als der Beginn des Umkehrbereichs 2 ist,
	; dann setze den axiale Override der
	; Zustellachse auf 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 Maschinenkoordinatensystem
	; gleich der Umkehrposition 1 ist,
	; dann setze den axialen Override der
	; Pendelachse auf 0
	; und setze den axialen Override der

NC-Sprache	**Kommentar**
	; Zustellachse auf 100% (damit wird die
	; vorhergehende Synchronaktion
	; aufgehoben)
N630 ID=3 WHENEVER $AA_IM[Z]==**$$SA_OSCILL_REVERSE_POS1[Z]**	
DO $AA_OVR[Z]=0 $AA_OVR[X]=100	
	; Immer wenn der Restweg der Teilzustellung
	; gleich 0 ist,
	; dann setze den axialen Override der
	; Pendelachse auf 100% (damit wird die
	; vorhergehende Synchronaktion
	; aufgehoben)
N640 ID=4 WHENEVER $AA_DTEPW[X]==0	
DO $AA_OVR[Z]=100 $AC_MARKER[0]=1 $AC_MARKER[1]=1	
N650 ID=5 WHENEVER $AC_MARKER[0]==1 DO $AA_OVR[X]=0	
N660 ID=6 WHENEVER $AC_MARKER[1]==1 DO $AA_OVR[X]=0	
	Wenn die aktuelle Position der Pendelachse im Werkstückkoordinatensystem
	gleich der Umkehrposition 1 ist,
	dann setze den axialen Override der Pendelachse auf 100%
	und setze den axialen Override der
	; Zustellachse auf 0 (damit wird die
	; zweite Synchronaktion einmalig
	; aufgehoben)
N670 ID=7 WHEN $AA_IM[Z]==**$$SA_OSCILL_REVERSE_POS1[Z]**	
DO $AA_OVR[Z]=100 $AA_OVR[X]=0	

5.3 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

5.3.1 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:

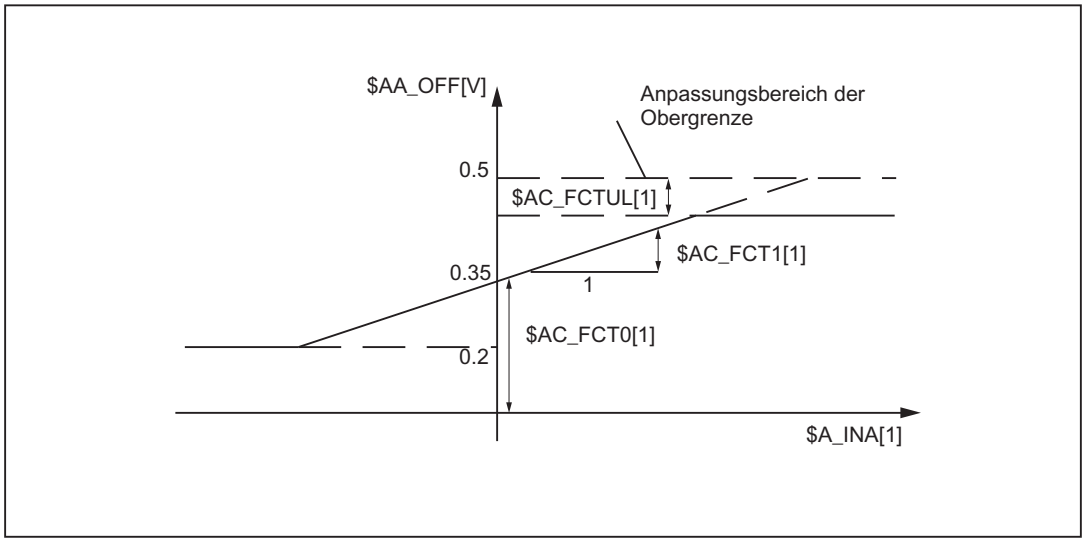


Bild 5-1 Abstandsregelung mit variabler Obergrenze

```
$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).
```

5.3.2 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.

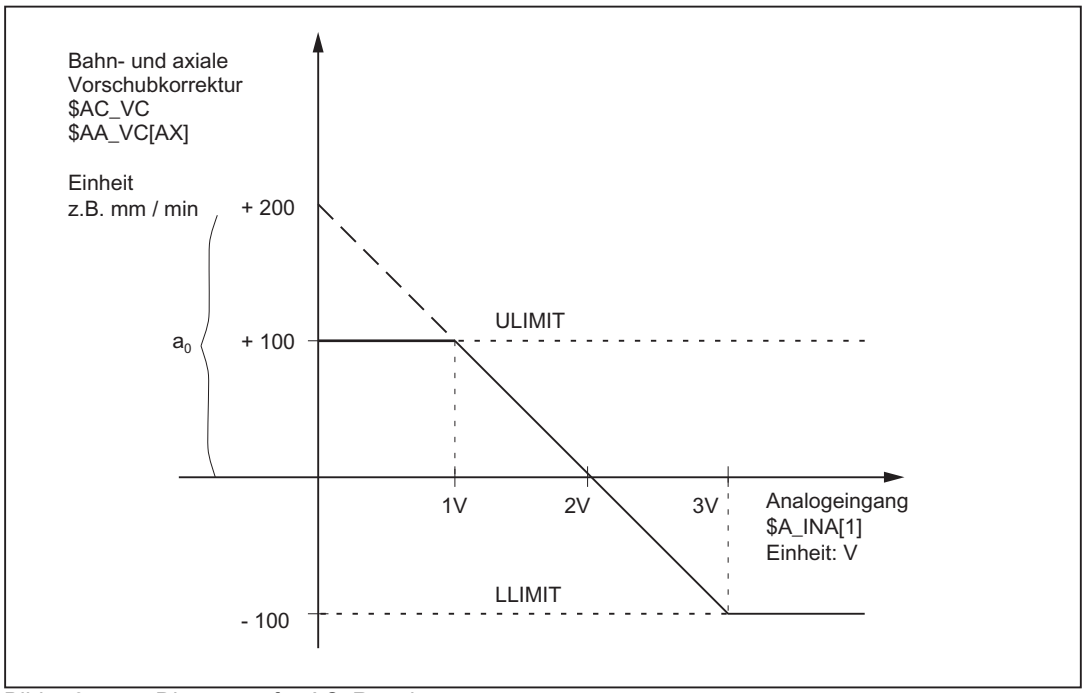


Bild 5-2 Diagramm für AC–Regelung

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])
```

5.3.3 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.

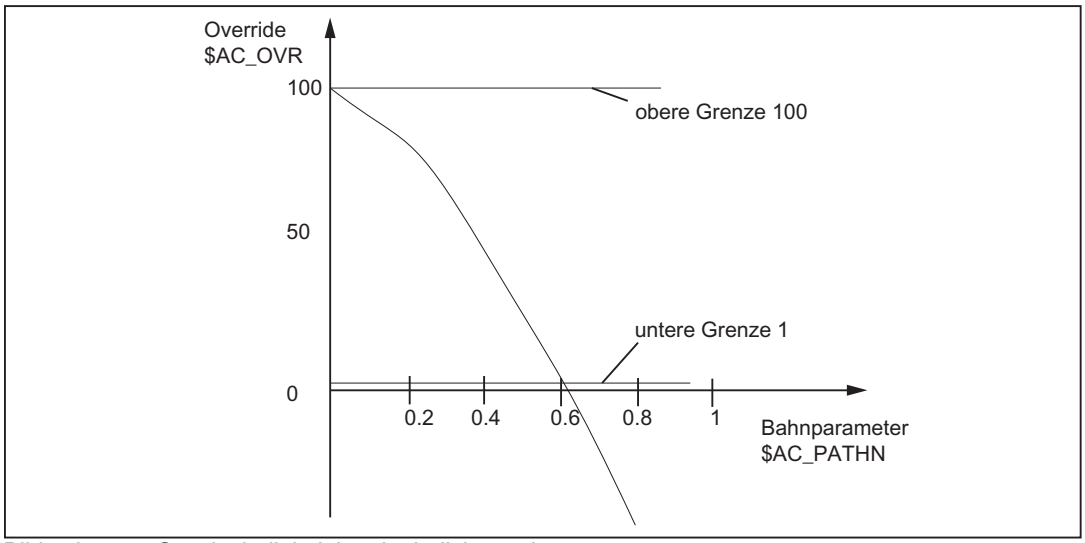


Bild 5-3 Geschwindigkeit kontinuierlich regeln

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
```

5.4 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`

5.5 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	; bei Satzwechsel R-Parameter-
$AC_MARKER[0] = $AC_MARKER[0] + 1	; zeiger weiterstellen
IDS=2 DO $R[10+$AC_MARKER[0]] =	; Schreibe jeweils die aktuelle Zeit
$AC_TIME	; 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 =	; bei Satzwechsel R-Parameter-
INDEX + 1	; zeiger weiterstellen
IDS=2 DO $R[10+INDEX] = $AC_TIME	; Schreibe jeweils die aktuelle Zeit
	; vom Satzanfang in R-Parameter

5.6 "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.

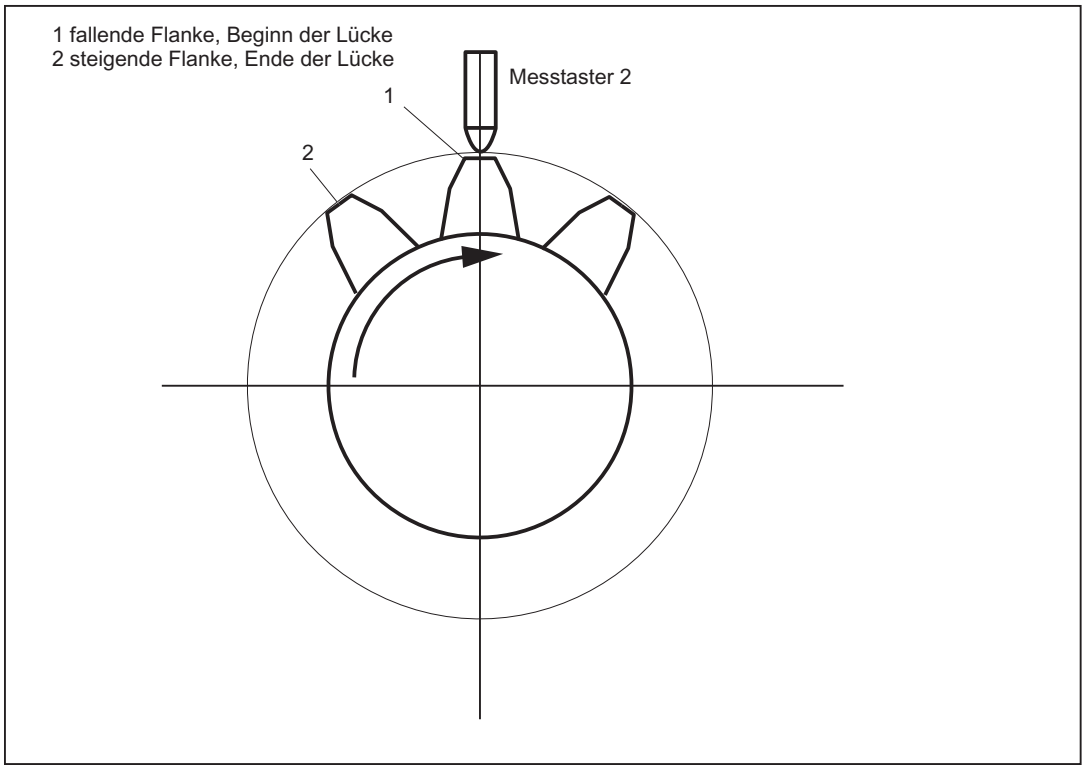


Bild 5-4 Schemabild zum Messen der Zahnradlücken

%_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-
   Variable 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
                                        Messtrace
                                        ;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
```

5.7 Achskopplungen über Synchronaktionen

5.7.1 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:
                                        LEADON (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            ;  Über Tabelle 4 einkoppeln
LEADON(CACH, BACH, 4)
N250 ID=2 EVERY $R1==0 DO            ;  Kopplung ausschalten
LEADOF(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
```

5.7.2 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.

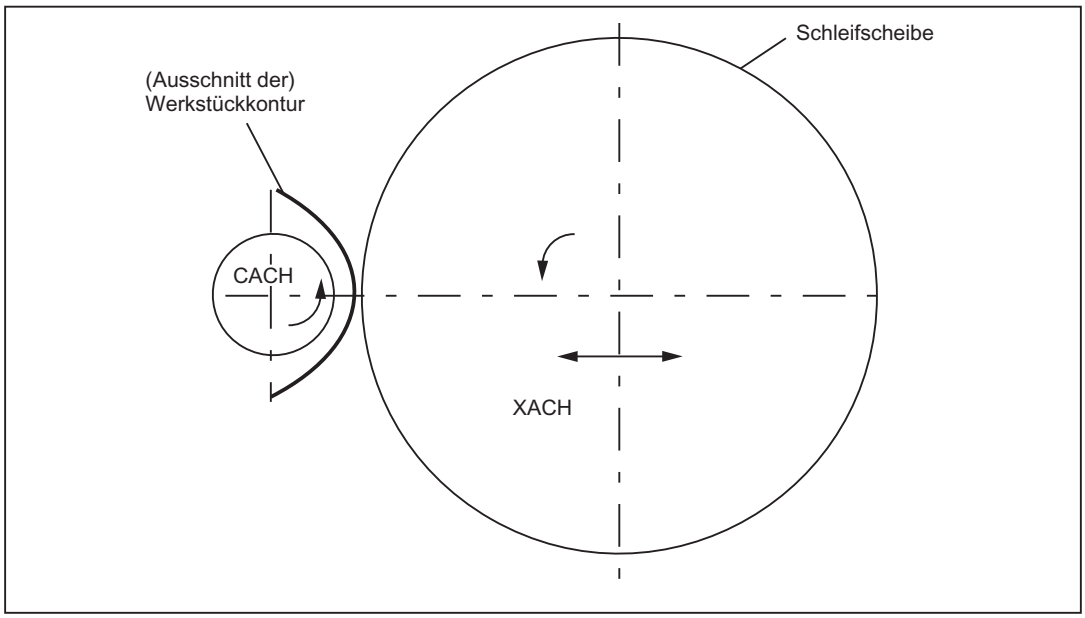


Bild 5-5 Schema Schleifen einer Unrund–Kontur

```
%_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
LEADON(XACH,CACH,2)
N2000 ID=4 EVERY $R2==0 DO
LEADOF(XACH,CACH)
```

```
N2100                                ;  *** Einschalter der LEADON CASW
                                     ;  Overridetabelle ***
N2200 WAITP(CASW)
N2300 ID=5 EVERY $R3==1 DO           ;  CTAB Kopplung EIN Leitachse CACH
LEADON(CASW,CACH,1)
N2400 ID=6 EVERY $R3==0 DO           ;  CTAB Kopplung AUS Leitachse CACH
LEADOF(CASW,CACH)
```

```
N2500                                ;  *** Override der CACH von Position
                                     ;  CASW mit ID 10 beeinflussen ***
N2700 ID=11 DO                       ;  "Achsposition" CASW auf OVR CACH zuweisen
$$AA_OVR[CACH]=$AA_IM[CASW]
```

```
N2900 WAITP(CACH)
N3000 ID=7 EVERY $R4==1 DO           ;  Als endlos drehende Rundachse starten
MOV[CACH]=1 FA[CACH]=R5
N3100 ID=8 EVERY $R4==0 DO           ;  Als endlos drehende Rundachse anhalten
MOV[CACH]=0
```

```
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                         ;  Vorlaufstop
N3800 R1=0                           ;  Stop 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".

5.7.3 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`

NC-Programm	**Kommentar**
`%_N_SCHERE1_MPF`	
`;$PATH=/_N_WKS_DIR/_N_DEMOFBE_WPD`	
`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) ; löschen einer evtl. bestehenden Kopplg.`	
`N700 CTABDEF(Y,X,1,0) ; Tabellendefinition`	
`N800 X=30 Y=30 ; Wertepaare`	

NC-Programm	Kommentar
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	; Startpos. Y Achse
	; Achse ist Linear
N1400 ID=1 EVERY $AA_IW[X]>$R3 DO PRESETON(X1,0)	; PRESET nach Länge R3, PRESTON 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
N1900 ID=10 EVERY $AA_IM[X]>$R3-30 DO LEADOF(Y,X)	; > 30 vor gefahrener Trennlänge abkoppeln
N2000 WAITP(X)	
N2100 ID=7 WHEN $R1==1 DO MOV[X]=1 FA[X]=$R4	; Strangachse stetig in Bewegung setzen
N2200 M30	

5.8 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          ;  ; wenn $A_DBB[0] von PLC gesetzt,
NULL_POS                             ;  Grundposition einnehmen
IDS=2 EVERY $A_DBB[1]==1 DO          ;  wenn $A_DBB[1] von PLC gesetzt,
ZIEL_POS                             ;  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
```

5.9 Synchronaktionen im Bereich WZW/BAZ

Einführung

Das folgende Bild zeigt den schematischen Ablauf Werkzeugwechselzyklus.

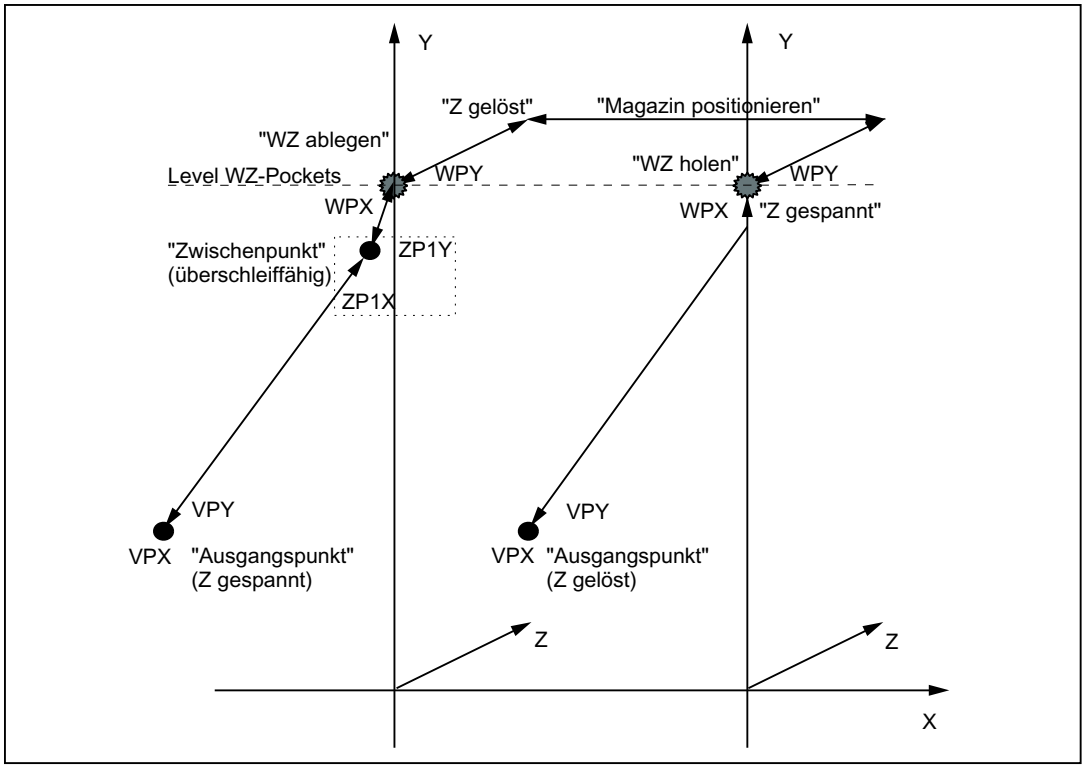


Bild 5-6 Schematischer Ablauf Werkzeugwechselzyklus

Ablaufdiagramm

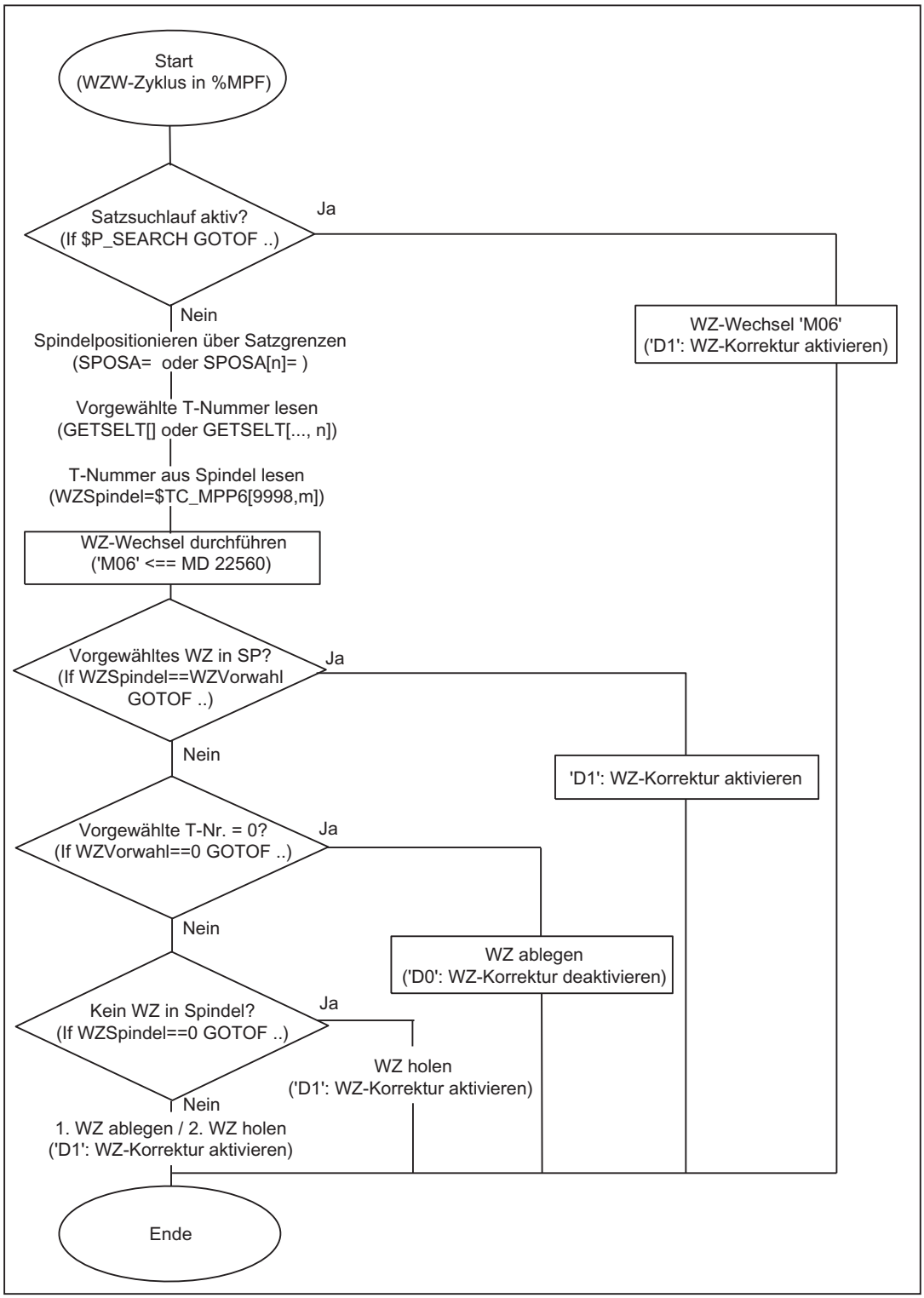


Bild 5-7 Ablaufdiagramm Werkzeugwechselzyklus

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

6

6.1 Maschinendaten

6.1.1 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

6.1.2 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

6.1.3 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
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

Nummer	Bezeichner: $MA_	Beschreibung
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"

6.2 Settingdaten

6.2.1 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

6.3 Signale

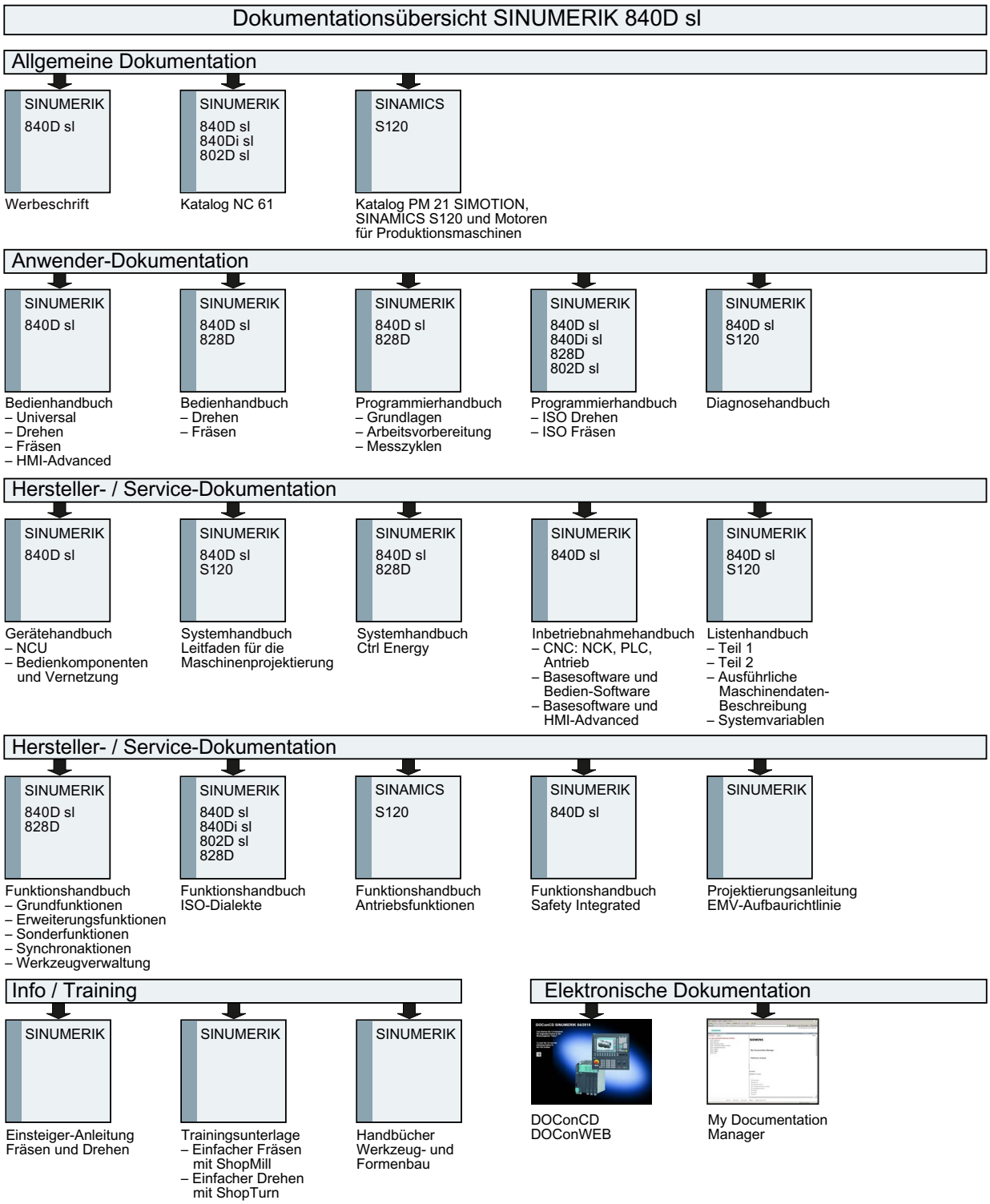
6.3.1 Signale von Kanal

DB-Nummer	Byte.Bit	Beschreibung
21, …	280.1	Modale Synchronaktionen gem. DBX300.0-307.7 sperren
21, …	300.0 -	Modale Synchronaktionen gem. DBX300.0-307.7 gesperrt, Quittung von NCK
21, …	300.0 -	Modale Synchronaktionen ID oder IDS 1 -
21, …	307.7	64 sperren. Anforderung an Kanal der NCK
21, …	308.0 -	Modale Synchronaktionen ID oder IDS 1 -
21, …	315.7	64 sperrbar. Mitteilung von NCK.

Anhang

A

A.1 Dokumentationsübersicht



Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

Z