SIEMENS

SINUMERIK 840D sl/
SINUMERIK 840D/840Di/810D

Synchronaktionen

Funktionsbeschreibung

Gültig für

Steuerung	*Softwarestand*
SINUMERIK 840D sl/ 840DE sl	1.3
SINUMERIK 840D powerline	7.4
SINUMERIK 840DE powerline (Exportvariante)	7.4
SINUMERIK 840Di sl	1.0
SINUMERIK 840DiE powerline (Exportvariante)	3.3
SINUMERIK 810D powerline	7.4
SINUMERIK 810DE powerline (Exportvariante)	7.4

Ausgabe 03/2006

Sicherheitshinweise

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 zugehörige Gerät/System darf nur in Verbindung mit dieser Dokumentation eingerichtet und betrieben werden. Inbetriebsetzung und Betrieb eines Gerätes/Systems dürfen nur von **qualifiziertem Personal** vorgenommen werden. Qualifiziertes Personal im Sinne der sicherheitstechnischen Hinweise dieser Dokumentation sind Personen, die die Berechtigung haben, Geräte, Systeme und Stromkreise gemäß den Standards der Sicherheitstechnik in Betrieb zu nehmen, zu erden und zu kennzeichnen.

Bestimmungsgemäßer Gebrauch

Beachten Sie bitte Folgendes:

Warnung

Das Gerät darf nur für die im Katalog und in der technischen Beschreibung vorgesehenen Einsatzfälle und nur in Verbindung mit von Siemens empfohlenen bzw. zugelassenen Fremdgeräten und –komponenten verwendet werden. Der einwandfreie und sichere Betrieb des Produktes setzt sachgemäßen Transport, sachgemäße Lagerung, Aufstellung und Montage, sowie sorgfältige Bedienung und Instandhaltung voraus.

Weitere Hinweise

Wichtig

Dieser Hinweis bedeutet, dass ein wichtiger Sachverhalt zu beachten ist.

Hinweis

Dieses Symbol erscheint in dieser Dokumentation immer dann, wenn weiterführende Sachverhalte angegeben werden.

Marken

Alle Erzeugnisbezeichnungen können eingetragene Marken oder Erzeugnisnamen der Siemens AG oder anderer Unternehmen 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.

Die Erstellung dieser Unterlage erfolgte mit Interleaf V 7

Siemens AG
Automation and Drives
Postafach 48 48
90437 NÜRNBERG
DEUTSCHLAND

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

Technische Änderungen vorbehalten.

Dokumentbestell–Nr. 6FC5397-5BP10-1AA0
Ausgabe 03/2006

Siemens–Aktiengesellschaft

Platz für Notizen

Vorwort

SINUMERIK–Dokumentation

Die SINUMERIK–Dokumentation ist in 3 Ebenen gegliedert:

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

Eine monatlich aktualisierte Druckschriften–Übersicht mit den jeweils verfügbaren Sprachen finden Sie im Internet unter:

http://www.siemens.com/motioncontrol

Folgen Sie den Menüpunkten “Support” → “Technische Dokumentation” → “Druckschriften–Übersicht”.

Die Internet–Ausgabe der DOConCD, die DOConWEB, finden Sie unter:

http://www.automation.siemens.com/doconweb

Informationen zum Trainingsangebot und zu FAQs (frequently asked question) finden Sie im Internet unter:

http://www.siemens.com/motioncontrol und dort unter Menüpunkt ”Support”

Zielgruppen

Die vorliegende Dokumentation wendet sich an den

- Projekteur der Anlage
- Technologen von Maschinenherstellern
- Inbetriebnehmer nach der Projektierung und Aufbau der Anlage
- PLC–Programmierer bei der Erstellung des PLC–Anwenderprogramms mit den aufgelisteten Signalen
- Servicetechniker zur Überprüfung und Interpretierung der Statusanzeigen und Alarme

Nutzen

Die Funktionsbeschreibung beschreibt die Funktionen, so dass die Zielgruppe die Funktionen kennt und auswählen kann. Es befähigt die Ziegruppe die Funktionen in Betrieb zu nehmen.

Standardumfang

In der vorliegenden Funktionsbeschreibung 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

Bei Fragen wenden Sie sich bitte an folgende Hotline:

Zeitzone Europa und Afrika

A&D Technical Support
Tel.: +49 (0) 180 5050 – 222
Fax: +49 (0) 180 5050 – 223

Internet: http://www.siemens.com/automation/support–request

E–Mail: mailto:adsupport@siemens.com

Zeitzone Asien und Australien

A&D Technical Support
Tel.: +86 1064 719 990
Fax: +86 1064 747 474

Internet: http://www.siemens.com/automation/support–request

E–Mail: mailto:adsupport@siemens.com

Zeitzone Amerika

A&D Technical Support
Tel.: +1 432 262 2522
Fax: +1 423 262 2289

Internet: http://www.siemens.com/automation/support–request

E–Mail: mailto:adsupport@siemens.com

Hinweis

Landesspezifische Telefonnummern für technische Beratung finden Sie im Internet:

http://www.siemens.com/automation/service&support

Fagen zum Handbuch

Bei Fragen zur Dokumentation (Anregungen, Korrekturen) senden Sie bitte ein Fax oder eine E–Mail an folgende Faxadresse:

Fax: +49 (0) 9131 98–63315

E–Mail: mailto:motioncontrol.docu@siemens.com

Faxformular: siehe Rückmeldeblatt am Schluss der Druckschrift

Internetadresse für SINUMERIK

http://www.siemens.com/sinumerik

EG–Konformitäts–erklärung

Die EG–Konformitätserklärung zur EMV–Richtlinie finden/erhalten Sie

- im Iternet:
  http://www.ad.siemens.de/csinfo

  und der Produkt–/Bestellnummer 15257461
- bei der zuständigen Zweigniederlassung des Geschäftsgebiets A&D MC der Siemens AG

Aufbau des Handbuches

Das vorliegende Funktionshandbuch ist wie folgt aufgebaut:

- Innentitel (Seite 3) mit dem Titel des Funktionshandbuches, den SINUMERIK Steuerungen sowie der Software und Version, für die diese Ausgabe des Funktionshandbuches gültig ist, und der Übersicht der Funktionsbeschreibungen.
- Gesamtinhaltsverzeichnis des Handbuches
- Funktionsbeschreibungen in alphanumerischer Reihenfolge gemäß den Funktionsbeschreibungs–Kurzzeichen
- Anhang mit Stichwortverzeichnis

Eine Funktionsbeschreibung enthält folgende Kapitel:

- Kurzbeschreibung
- Ausführliche Beschreibung
- Randbedingungen
- Beispiele
- Datenlisten

Hinweis

Ausführliche Daten– und Alarm–Beschreibungen finden sich:

– für Maschinen– und Settingdaten:
nur elektronisch auf DOConCD oder DOConWEB

– für NC/PLC–Nahtstellensignale: /FB1/ NC/PLC–Nahtstellensignale (Z1)

– für Alarme: /DA/ Diagnoseanleitung

Zielsetzung

Die vorliegende Dokumentation beschreibt die Funktion Synchronaktionen für SINUMERIK 840D sl ab SW 1.3,
SINUMERIK 840D ab SW 4 und für SINUMERIK 810D ab SW 2.

Die Funktionsbeschreibungen sind nur für den speziellen bzw. bis zum aufgeführten Softwarestand gültig. Bei neuen Softwareständen sind die dazu gültigen Funktionsbeschreibungen anzufordern. Alte Funktionsbeschreibungen sind für neue Softwarestände nur noch teilweise verwendbar.

Nähere Informationen zu weiteren Druckschriften über SINUMERIK 840D/840Di/810D sowie zu Druckschriften, die für alle SINUMERIK–Steuerungen gelten (z.B. Universalschnittstelle, Meßzyklen ...) erhalten Sie von Ihrer Siemens–Niederlassung.

Die Funktionsbeschreibungen vermitteln die für die Projektierung und Inbetriebnahme benötigten Informationen.

Technische Hinweise

Schreibweisen

In dieser Dokumentation gelten folgende Schreibweisen und Abkürzungen:

- PLC–Nahtstellensignale –> NST ”Signalname” (Signaldatum)
  Bsp.: – NST ”MMC–CPU1 ready” (DB10, DBX108.2) d.h. das Signal ist im Datenbaustein 10, Datenbyte 108, Bit 2 abgelegt.
  – NST ”Vorschub–/Spindelkorrektur” (DB31–48, DBB0) d.h. die Signale liegen achs–/spindelweise in den Datenbausteinen 31 bis 48, Datenbausteinbyte 0.
- Maschinendatum –> MD: MD_NAME (deutsche Bezeichnung)
- Settingdatum –> SD: SD_NAME (deutsche Bezeichnung)
- Das Zeichen ”≙” bedeutet ”entspricht”
- NEW_CONF (cf) – Funktion ”Neu Konfigurierung” der PLC–Nahtstelle
  – Taste ”RESET” auf der Bedieneinheit, oder
- RESET (re) Taste ”RESET” auf der Bedieneinheit, oder
- Sofort (im) nach der Eingabe des Wertes

Datentypen

In der Steuerung werden folgende Datentypen verwendet:

- DOUBLE
  Real– oder Integerwerte (Kommawerte oder ganzzahlige Werte)
  Eingabegrenzen von $+/-4{,}19*10^{-307}$ bis $+/-1{,}67*10^{308}$
- DWORD
  Integerwerte (ganzzahlige Werte)
  Eingabegrenzen von $-2{,}147*10^{9}$ bis $+2{,}147*10^{9}$
- BOOLEAN
  Mögliche Eingabewerte: true oder false bzw. 0 oder 1
- BYTE
  Integerwerte (ganzzahlig) von –128 bis +127
- STRING
  bestehend aus max. 16 ASCII–Zeichen (Großbuchstaben, Ziffern und Unterstrich)

Mengengerüst

Die Erläuterungen der PLC–Nahtstelle in den einzelnen Funktionsbeschreibungen gehen von einer theoretischen Maximalanzahl der Komponenten aus:

- Betriebsartengruppen (zugehörige Signale abgelegt in DB11)
- Kanäle (zugehörige Signale abgelegt in DB21, ...)
- Achsen (zugehörige Signale abgelegt in DB31, ...)

Die wirklich realisierbare Komponentenanzahl des jeweiligen Softwarestandes entnehmen Sie bitte
Literatur: /BU/ ”Bestellunterlage” Katalog NC 60 und NC 61.

■

■

Platz für Notizen

1 Kurzbeschreibung

Definition Synchronaktionen

Bewegungssynchronaktionen (kurz Synchronaktionen) sind vom Anwender programmierte Anweisungen, die synchron zur Bearbeitung des Teileprogrammes vom NCK ausgewertet werden. Der Ausführzeitpunkt der Aktionen kann über Bedingungen definiert werden. Die Bedingen werden im Interpolationstakt überwacht. Ist die in der Synchronaktion enthaltene Bedingung erfüllt oder keine angegeben, so werden zugeordnete Aktionen synchron zur weiteren Bearbeitung aktiviert.

Anwendungen

Der folgende Auszug aus den vielfältigen Möglichkeiten gibt Hinweise zu den Einsatzmöglichkeiten für Aktionen in Synchronaktionen.

- 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

In Kapitel ”Ausführliche Beschreibung” werden alle Möglichkeiten dargestellt.

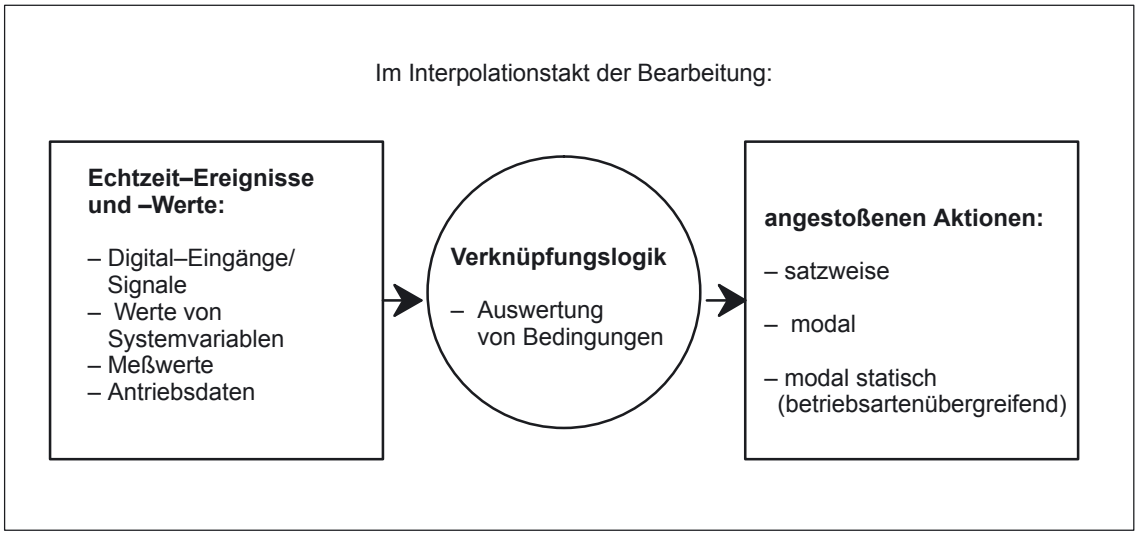


Bild 1-1 Synchronaktionen schematisch

Die Details der Programmierung von Synchronaktionen finden Sie in

Literatur: /PGA/, Programmierhanbuch Arbeitsvorbereitung.

Die folgenden Kapitel beschreiben die:

- funktionalen Zusammenhänge für Synchronaktionen in Kapitel 2,
- sowie die erforderlichen Maschinendaten in Kapitel 4,
- Anwendungsbeispiele in Kapitel 6.

Hinweis

Die vorliegende Beschreibung umfaßt die Funktionalität des aktuellen Software–Standes. Die Leistungen der Synchronaktionen bis einschließlich SW–Stand 3 sind beschrieben in:

Literatur:
/FB2/ Funktionshandbuch Erweiterungsfunktionen; Synchronaktionen (S5).

■

2 Ausführliche Beschreibung

2.1 Komponenten von Synchronaktionen

Struktur einer Synchronaktion

Komponente:	Gültigkeit, Identifikationsnummer	Häufigkeit	G–Code für Bedingung und Aktion	Bedingung	Aktionskennwort (fest)	G–Code für Aktion	Aktion oder Technologiezyklus s. Kap. 2.5
Beispiel:	IDS=1	EVERY	G70	$AAA_IM[B] > 15	DO	G71	POS[X]= 100

Die Bestandteile der Synchronaktion:

- Gültigkeit:
  - mit Identifikationsnummer
  - ohne Identifikationsnummer
- Häufigkeit
- G–Code für Bedingung und Aktion
- Bedingung
- G–Code für Aktionen (ab SW 5)
- Aktion(en) / Technologiezyklus

werden im Folgenden einzeln erklärt.

Gültigkeit, ID–Nummer

Für die Festlegung des Gültigkeitsbereiches einer Synchronaktion bieten sich drei Möglichkeiten:

- keine Angabe
- ID
- IDS

Keine Angabe

Synchronaktionen ohne Gültigkeitsangabe wirken satzweise, d.h. nur für den darauffolgenden Satz.

Satzweise wirksame Synchronaktionen wirken

- nur im **AUTOMATIK**–Betrieb.
- modal über alle Vorlauf–Stopp–Sätze (auch implizit erzeugte) und über implizit erzeugte Zwischensätze.

ID

Synchronaktionen mit Gültigkeitskennung **ID** wirken **modal** in darauffolgend programmierten Sätzen. Sie wirken nur im **AUTOMATIK**–Betrieb.

Begrenzung:

- bis eine andere Synchronaktion mit gleicher Identifikationsnummer programmiert wird.
- bis zum Löschen, CANCEL(i), siehe Kapitel 2.5.1.

IDS

Statisch wirksame Synchronaktionen, die mit dem Schlüsselwort "**IDS**" programmiert werden, sind **in allen Betriebsarten** aktiv. Sie werden auch als statische Synchronaktionen bezeichnet. Option.

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

Identifikations-nummern

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 Kapitel 2.5.1. Modale / statische Synchronaktionen mit Identifikationsnummern von 1 – 64 können von PLC aus verriegelt und freigegeben werden, siehe Kapitel 2.6.1.

Die Identifikationsnummern müssen im Kanal eindeutig vergeben werden.

Anwendung für statische Synchronaktionen:
– AC–Schleifen auch in Betriebsart JOG aktiv
– Verknüpfungslogik für Safety Integrated
– Überwachungsfunktionen, Reaktion auf Maschinenzustände in allen Betriebsarten
– Optimierung des Werkzeugwechsels
– Zyklische Maschinen

Beispiele:

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 AUTOMATIC bei aktivem Programm wirksam:
STOPREOF,
DELDTG

Häufigkeit

Über das Schlüsselwörter (siehe Tabelle) wird angegeben, wie oft die darauffolgende Bedingung abgefragt und die zugehörende Aktion bei erfüllter Bedingung ausgeführt werden soll. Die angegebenen Schlüsselworte sind Bestandteil der Synchronaktionsbedingung.

Tabelle 2-1 Wirkung der Schlüsselwörter bei zyklischer Prüfung der Bedingung einer Synchronaktion

Schlüsselwort	Abfrage–Häufigkeit
keins	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 auch nach Ausführung der Aktion aktiv solange, bis die Synchronaktion gelöscht oder inaktiv geschaltet wird.

Details zu Technologiezyklen finden Sie unter Kapitel 2.5.

Löschen mit CANCEL

Wird eine aktive Synchronaktion mit **CANCEL** aus dem Teileprogramm abgewählt (gelöscht), so wird die aktive Aktion nicht beeinflußt. Positionierbewegungen werden wie programmiert beendet. Mit dem Befehl CANCEL kann eine modal oder statisch wirksame Synchronaktion gelöscht werden.
Wird eine Synchronaktion gelöscht, während die daraus aktivierte Positionierachsbewegung noch aktiv ist, so wird die Positionierachsbewegung abgeschlossen. Ein Kanal–Stopp bricht auch die Positionierbewegung aus Synchronaktionen/Technologiezyklen ab.

G–Code für Bedingung und Aktion

Mit programmierbaren G–Codes in Synchronaktionen kann erreicht werden, 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.

Anwendungsfälle:
Festlegung der Maßsysteme für Bedingungsauswertung und Aktion durch G–Codes G70, G71, G700, G710.

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.

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 IPO–Takt ausgeführt.

Mögliche Bedingungen sind z.B.:

- Vergleich von Hauptlaufvariablen wie digitale oder analoge Ein–/Ausgänge
- Boole'sche Verknüpfungen zwischen Vergleichsergebnissen
- Berechnung von Echtzeitausdrücken
- Zeit/Entfernung vom Satzanfang
- Messwerte, Messergebnisse
- Servo–Werte
- Geschwindigkeiten, Achsstatus

Bis Softwarestand 3 ist als Bedingung der Vergleich einer Hauptlaufvariable mit einem im Vorlauf berechneten Ausdruck oder der Vergleich zweier Hauptlaufvariablen zulässig.

Beispiele:
WHENEVER **$AA_IM[X] > 10.5*SIN(45**) DO .... oder
WHENEVER **$AA_IM[X] > $$AA_IM[X1]** DO ...

Es können auch zusätzlichVergleiche
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.**

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

Innerhalb einer Bedingung können zwei oder mehr Echtzeit–Ausdrücke miteinander verglichen werden. Vergleiche sind jeweils zwischen **typgleichen** Variablen oder Teilausdrücke möglich.
Beispiel:
WHEN $AA_IM[X2] <= **$AA_IM[X1] +.5** DO $AA_OVR[X1]=0
; Anhalten, wenn Sicherheitsabstand überschritten ist.
Weitere Beispiele zu Bedingungen finden Sie im Kapitel 6.1.

Die Berechnungen von Echtzeit–Ausdrücken sind beschrieben im
Kapitel "Auswertungen und Berechnungen in Echtzeit".
Bei den Bedingungen können alle die Systemvariablen angesprochen werden, dessen Daten von der NC über Synchronaktionen gelesen bzw. geschrieben werden. Darüberhinaus alle Maschinendaten und Settingdaten mit Lesen des Wertes im Hauptlauf:

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

Hinweis

- Neben den vordefinierten Variablen können spezielle GUD–Variablen in Synchronaktionen verwendet werden, siehe
  Kapitel "Spezielle Hauptlaufvariablen für Synchronaktionen" .
- Rechenparameter werden in Synchronaktionen mit $R... adressiert.

G–Code für die Aktion

Dieser G–Code gibt für alle Aktionen im Satz und Technologiezyklen gegenbenenfalls einen anderen G–Code als den bei der Bedingung gesetzten vor.

Sind Technologiezyklen im Aktionsteil, so gilt der G–Code auch nach Abschluß des Technologiezyklus für alle darauffolgenden Aktionen bis zum nächsten G–Code modal weiter.

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

Aktionen

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

Beispiel: 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 ein Programm (Name) 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. Ein Technologiezyklus ist eine Folge von Aktionen, die sequentiell im Interpolationstakt abgearbeitet werden, siehe Kapitel 2.5. ”Aufruf von Technologiezyklen”.

Anwendung: 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 Variable erfolgt während der Aufbereitung.

Bei Verwendung von **Hauptlauf**variablen
(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 Hauptlaufvariable werden im **Interpolationstakt** ausgewertet. Die Satzaufbereitung wird nicht unterbrochen.

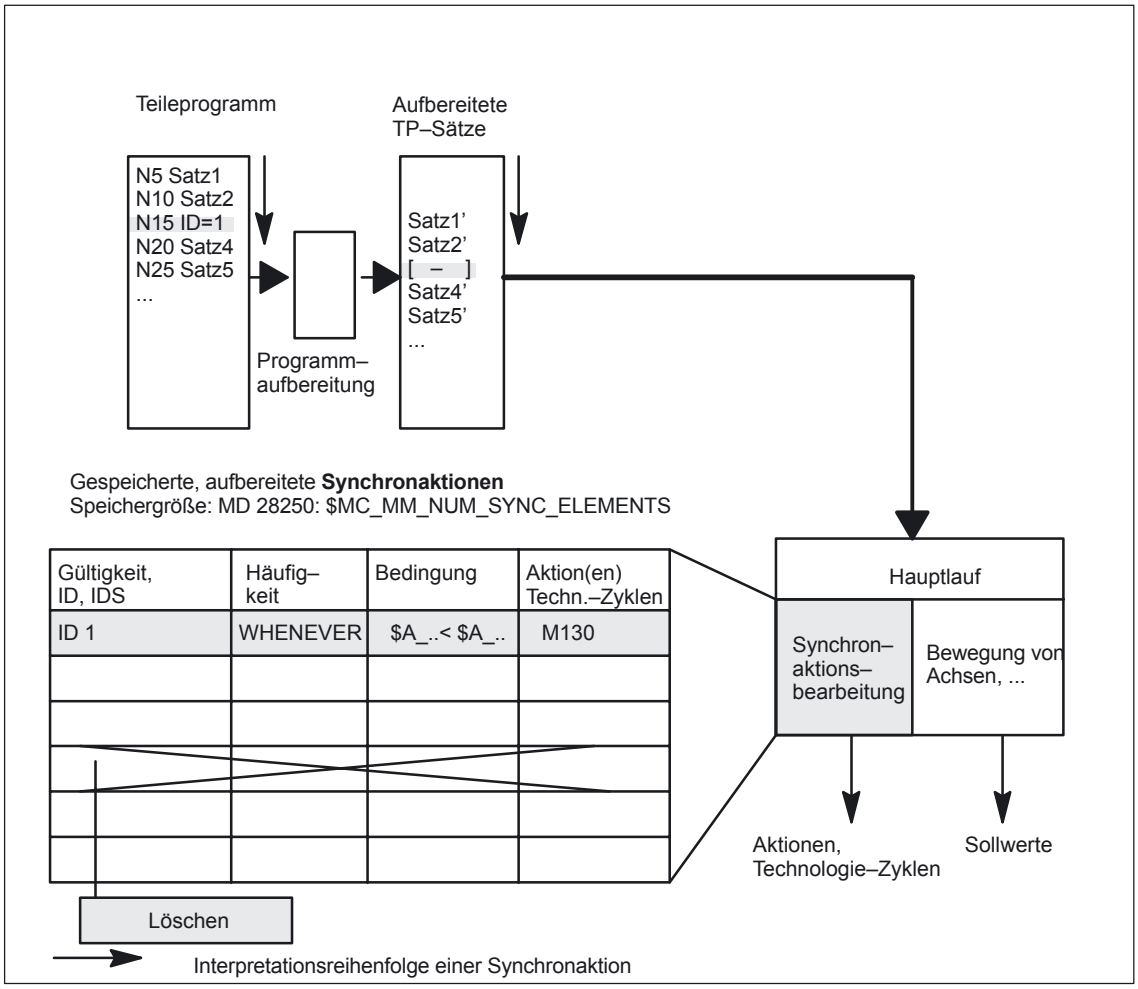


Bild 2-1 Bearbeitung von Synchronaktionen schematisch

Bearbeitung der Synchronaktionen

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 Interpolationstaktes werden
modal wirksame Synchronaktionensanweisungen 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.

2.1.1 Definition von Bewegungssynchronaktionen

Definierende Programme

Bewegungssynchronaktionen können definiert werden:

- im Teileprogramm
- Statische Synchronaktionen in einem asynchronen Unterprogramm ASUP, das durch die PLC aktiviert wird.

2.1.2 Ausführung der Aktionen

Ausführungsbedingungen

Die Aktionen in Bewegungssynchronaktionen werden ausgeführt, wenn:

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

  Kapitel 2.5.1 ”Koordinierungen zwischen Synchronaktionen, Technologiezyklen, Teileprogramm und PLC”.
- aufgrund des Häufigkeitsschlüsselwortes eine Auswertung fällig ist
- die Bedingung erfüllt ist

Weitere Details finden Sie in den folgenden Unterkapiteln.

2.1.3 Liste möglicher Aktionen

- Ausgabe von M–, S– und H–Hilfsfunktionen an die PLC
- Durch Setzen (Schreiben) von Hauptlaufvariablen ist möglich:
  - Überlagerte Bewegung ($AA_OFF).
  - Aufgeschaltete Werkzeuglängenkorrekturen ($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–Wertes 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
- 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 Momemt FOC (FOCON/FOCOF)
- Erweitertes Stillsetzen und Rückziehen (Funktionshandbuch (M3))
- Lesen und, wenn entsprechend gekennzeichnet, Schreiben von
  Systemvariablen siehe /PGA1/ Listenandbuch Systemvariablen.

In Kapitel 2.4 werden diese Aktionen im Detail beschrieben.

2.2 Auswertungen und Berechnungen in Echtzeit

Abgrenzung

Die Berechnungen, die in Echtzeit durchgeführt werden, sind eine Untermenge der in der NC–Sprache möglichen Berechnungen. Sie ist für die Datentypen REAL, INT, CHAR und BOOL möglich.

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

Anwendungsfeld

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 im /LHB/ Handbuch der Systemvariablen in der ”Liste der Systemvariablen” in einen hierfür vorgesehenen Feld gekennzeichnet.

Kennzeichen von Hauptlaufvariablen

Hauptlaufvariablen sind alle Variablen, die beginnen mit:
$A... ,aktuelle Hauptlaufvariable
$V... , Servo–Werte
$R... , R–Parameter.
NC–Maschinen– und Settingdaten werden im Hauptlauf mit
$$M...
$$S...
in Synchronaktionen interpretiert.

Hinweis

Maschinen– und Settingdaten–Werte zum Hauptlaufzeitpunkt lesen
Sie werden für einen Online–Zugriff mit $$S... oder $$M... adressiert und zum Hauptlaufzeitpunkt ausgewertet, während MD– oder 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

Innerhalb eines Ausdrucks in Synchronaktionen können nur Hauptlaufvariable **eines Datentyps** miteinander verknüpft werden. Um trotzdem Daten verschiedener Typen zu verarbeiten, können die bereitgestellten Konvertierungsroutinen zur Typangleichung benutzt werden. Im Gegensatz zum vollen Ausdruck in der NC–Sprache erfolgt die Berechnung im Datentyp der Hauptlaufvariablen.

```
... DO $R10 = $AC_PARAM[0]      ; Zulässig REAL, REAL
... DO $R10 = $AC_MARKER[0]     ; nicht erlaubt REAL, INT
```

Konvertierungs-routinen ab SW 5.3

Der Anwendern kann in der Synchronaktion impliziteTypwandlung von REAL nach INT und umgekehrt zwei Konvertierungsroutinen **RTOI( ) und ITOR( )** für die Typwandlung explizit aufrufen. Diese Funktionen ist für Softwarestäde gedacht , bei denen noch keine interne Typkonvertierung von Werten möglich war. Beide Konvertierungsroutinen sind

- im Teileprogramm und
- aus der Synchronaktion

aufrufbar.

ITOR

REAL ITOR(INT) – Konvertierung von Integer nach Real

Die Funktion wandelt den übergebenen Integer–Wert in einen Real–Wert um und gibt diesen zurück. Die übergebene Variable wird dabei nicht verändert.

Beispiel:

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

RTOI

INT RTOI(REAL) – Konvertierung von Real nach Integer

Die Funktion RTOI() wandelt den übergebenenen Real–Wert in eine gerundete INT–Zahl um und gibt diese zurück. Liegt der Übergabewert außerhalb des als Integer–Wert eindeutig darstellbaren Bereiches, so wird Alarm 20145 ”Bewegungssynchronaktion: Arithmetikfehler” ausgegeben, und die Konvertierung wird nicht durchgeführt. Die übergebene Variable wird dabei nicht verändert.

Hinweis

Die Funktion RTOI() ist nicht umkehrbar eindeutig, d.h. aus dem Rückgabewert, läßt sich der ursprüngliche REAL Wert nicht mehr ermitteln, da die Nachkommastellen bei der Konvertierung verlorengehen!

Beispiele RTOI:

```
$AC_PARAM[1] = 561.4378
ID=1 WHEN TRUE DO $AC_MARKER[1] = RTOI( $AC_PARAM[1] )
                                  ; Ergebnis: 561

...

$AC_PARAM[1] = –63.867
ID=1 WHEN TRUE DO $AC_MARKER[1] = RTOI( $AC_PARAM[1] )
                                  ; Ergebnis:–64

...

$AC_MARKER[1]= 10
$AC_PARAM[1] = –6386798797.29
ID=1 WHEN TRUE DO $AC_MARKER[1] = RTOI( $AC_PARAM[1] )
                  ;Ergebnis: Alarm 20145

                  ;$AC_MARKER[1] = 10 (unverändert wegen Alarm)
```

Implizite Typwandlung ab SW 6.4

In Synchronaktionen können ab SW 6.4 Variablen verschiedener Datentypen wie z.B. von REAL nach INT und umgekehrt **ohne** den Aufruf der Funktion RTOI und ITOR einander zugewiesen werden.

Liegen bei der Konvertierung von REAL nach INTEGER Werte außerhalb des Intervalls [INT_MIN, INT_MAX], dann wird der Alarm 20145 ”Bewegungssynchronaktion: Arithmetikfehler” ausgegeben, und die Konvertierung wird nicht durchgeführt.

Beispiele:
bisher

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

ab SW 6.4

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

bisher

```
$AC_PARAM[1] = 561.4378
ID=1 WHEN TRUE DO $AC_MARKER[1] = RTOI( $AC_PARAM[1] ) ; 561
```

ab SW 6.4

```
$AC_PARAM[1] = 561.4378
ID=1 WHEN TRUE DO $AC_MARKER[1] = $AC_PARAM[1]          : 561
```

interne Typkonvertierung von Werten ab SW 7.4

Ohne den Aufruf der Funktion RTOI und ITOR könnenTypkonvertierungen für Wertzuweisungen und Parameterübergaben mehrerer impliziter Typwandlungen ausgelöst werden. Möglich sind Typkonvertierungen der Datentypen von

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

Beispiele:

implizite Wandlung von INTEGER nach BOOLen

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

implizite Wandlung von REALnach BOOLen

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

implizite Wandlung von BOOLen nach INTEGER

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

implizite Wandlung von BOOLen nach REAL

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

Bei Typkonvertierungen von REAL nach INT wird bei gebrochem Wert $\geqq 0.5$ aufgerundet. Ansonsten wird anlog der Funktion ROUND abgerundet. Bei Werteüberschreitungen wird der Alarm 20145 ausgelöst.

Literatur:

/PGA/ Programmierhandbuch Arbeitsvorbereitung,
Bewegungssynchronaktionen, Kap. ” Hauptlaufvariablen für Synchronaktionen”

Grundrechenarten

Echtzeit–Variablen vom Typ REAL und INT können durch Grundrechenarten:

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

miteinander verknüpft werden.

Es können nur Variable gleichen Typs verknüpft werden.

Ausdrücke

Ausdrücke aus Grundrechenarten können geklammert und geschachtelt werden. Siehe Prioritäten von Operatoren auf der Folgeseite.

Vergleiche

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 Operatoren

Möglich sind die bitweisen Operatoren:

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

Operanden sind Variablen und Konstanten vom Typ INT.

Priorität von Operatoren

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

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

und gegebenenfalls runde Klammern zu verwenden.
Das Verknüpfungsergebnis von Bedingungen muss vom Typ BOOL sein.

Beispiel einer mehrgliedrigen Bedingung:
WHEN ($AA_IM[X] > WERT) AND ($AA_IM[Y] > WERT1) DO ...

Funktionen

Von einer Hauptlauf–Variable des Typs REAL kann auch der Funktionswert sin, cos etc. gebildet werden.
Möglich sind z.B. auch die Funkionen:

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

Beispiel:
... DO $AC_PARAM[3]=**COS**($AA_IM[X])

Die Erklärung der Bedeutungen der angegebenen Funktionen stehen in:

Literatur: /PG/ Programmierhandbuch Grundlagen
/PGA/ Programmierhandbuch Arbeitsvorbereitung

Indizierung

Der Index einer Hauptlauf–Feldvariablen kann wiederum eine Hauptlaufvariable sein.

Beispiel:
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 Echtzeit–Variablen 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]]**

2.3 Spezielle Hauptlaufvariablen für Synchronaktionen

Zur **vollständigen Liste** der ansprechbaren Systemvariablen siehe

Literatur:
/PGA1/ Listenhandbuch Systemvariablen.

Die in Synchronaktionen ansprechbaren Systemvariablen sind im Feld SA: ”Verwendung in Synchronaktionen möglich” gekennzeichnet. Eine im Hauptlauf aktualisierte Systemvariable wird auch im Hauplauf synchroniesiert, wenn dies im Feld: HL–Sync angekreuzt ist und damit bestätigt wird.

Im folgenden werden die Eigenschaften einiger spezieller Hauptlaufvariablen vorgestellt:

- Synchronaktions–Marker–Variable $AC_MARKER[n]
  - Kanalspezifische Merker
- Zeiten (Timer)
- Synchronaktionsparameter
- R–Parameter
- Maschinen– und Settingdaten
- FIFO–Variablen (Durchlaufspeicher)

2.3.1 Synchronaktions–Marker–Variable $AC_MARKER[n]

Kanalspezifische Marker

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

Datentyp: INTEGER
n: Feldindex des Markers: 0–n

Die Anzahl der Marker pro Kanal wird eingestellt über das Maschinendatum
MD28256 $MC_NUM_AC_MARKER.

Im MD28256 $MC_NUM_AC_MARKER kann
als Höchstwert 20000 angegeben werden.
Ein Element benötigt 4 Bytes Speicherplatz. Es muß darauf geachtet werden, daß der erforderliche Speicherplatz in der gewählten Speicherart verfügbar ist.

Die Marker sind unter gleichem Namen einmal pro Kanal vorhanden.

Der Speicherort für $AC_MARKER[n] kann mit dem Maschinendatum
MD28257 $MC_MM_BUFFERED_AC_MARKER mit
0: im dynamischen Speicher (aktiven Filesystem)
1: im statischer Speicher (passiven Filssystem)
ausgewählt werden.

Im statischen Speicher abgelegte Marker können in die Datensicherung einbezogen werden. Siehe 2.3.7

Bei Power On, NC–Reset und Programmende werden die Marker auf 0 gesetzt. Damit sind gleiche Startbedingungen für jeden Programmdurchlauf gegeben.

2.3.2 Timer–Variable $AC_TIMER[n] (Zeiten)

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

Definition

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

Die Anzahl der verfügbaren Timer–Variablen wird per Maschinendatum
MD28258 $MC_MM_NUM_AC_TIMER
festgelegt.

Timer setzen

Das Hochzählen einer Timer–Variable wird gestartet durch Wertzuweisung:
$AC_TIMER[n]=value
n: Nummer der Zeitvariable
value: Startwert (i.d.R. 0)

Timer anhalten

Das Hochzählen einer Timer–Variable wird gestoppt durch Zuweisung eines negativen Wertes:
$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 Istwertes über Analogausgang 500ms nach Erkennen eines digitalen Eingangs:
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

2.3.3 Synchronaktions–Parameter $AC_PARAM[n]

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

Definition

Datentyp: REAL
n: Nummer des Parameters 0 – n

Die Anzahl der verfügbaren AC–Parameter–Variablen pro Kanal wird über das Maschinendatum
MD28254 $MC_MM_NUM_AC_PARAM
festgelegt.

Die Parameter sind unter gleichem Namen einmal pro Kanal vorhanden. Die Parameter $AC_PARAM werden im dynamischen Speicher gehalten.

Im MD28255 $MC_NUM_AC_PARAM kann
als Höchstwert 20000 angegeben werden.
Ein Element benötigt 4 Bytes Speicherplatz. Es muss darauf geachtet werden, daß der erforderliche Speicherplatz in der gewählten Speicherart verfügbar ist.

Der Speicherort für $AC_PARAM[n] kann mit dem Maschinendatum
MD28255 $MC_MM_BUFFERED_AC_PARAM mit
0: im dynamischen Speicher (aktiven Filesystem, Default)
1: im statischer Speicher (passiven Filssystem)
ausgewählt werden.

Im statischen Speicher abgelegte Synchronaktionsparameter können in die Datensicherung einbezogen werden. Siehe 2.3.7.

Bei Power On, NC–Reset und Programmende werden die Parameter auf 0 gesetzt. Damit sind gleiche Startbedingungen für jeden Teileprogrammdurchlauf gegeben.

2.3.4 Rechenparameter $R[n]

Verwendung in Synchronaktionen

Mit dem Einleitungszeichen **$** vor dem R–Parameter können Rechenparameter auch in Synchronaktionen verwendet werden. Die Feld–Variablen

$R[n] oder **$Rn** dienenzur Berechgnungen in Synchronaktionen
R[n] oder **Rn** dienenzur Berechgnungen im Teileprogramm,

die im statischen Speicher batteriegepuffert gehalten werden. Deshalb behalten R–Parameter über Programmende, RESET, Power ON hinweg ihre Werte.

Beispiel:
Messwert im Rechenparameter übernehmen
WHEN $AC_MEA== 1 DO **$R10**= $AA_MM[Y]
; wenn gültige Messung vorliegt, Messwert in R–Parameter übernehmen.

Definition

Datentyp REAL
n: Nummer der Feld–Variable

Beispiel

Auswertung des R–Parameters

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

2.3.5 Maschinen– und Settingdaten

MD, SD lesen und schreiben

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

- MD, SD, die während der Bearbeitung unverändert bleiben
- MD, 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:
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

Ein vollständiges Beispiel für Pendeln mit Zustellung im Umkehrbereich finden Sie in Kapitel 6.2 und:

Literatur:
/FB2/ Funktionshandbuch Erweiterungsfunktionen; Pendeln (P5).

Zum Hauptlaufzeitpunkt lesen

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

Beispiel:
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:
/LIS1/ Listen (Buch 1); Kapitel ”MD–/SD–Listen”.

Adressierung:
Zu schreibende Maschinendaten und Settingdaten sind eingeletet mit **$$** zu adressieren.

Beispiel:

```
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
```

2.3.6 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 Bild 2-3 .

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 MD 28266: $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.

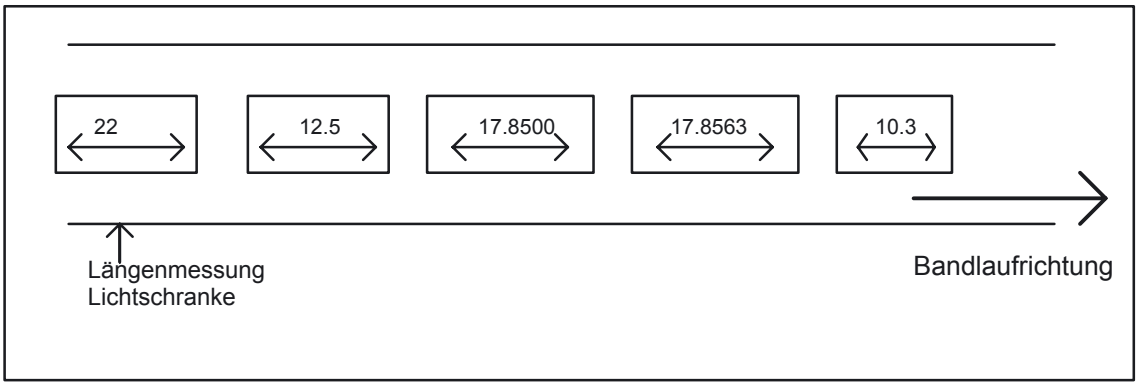


Bild 2-2 Produktlängen einer Teilefolge auf Band

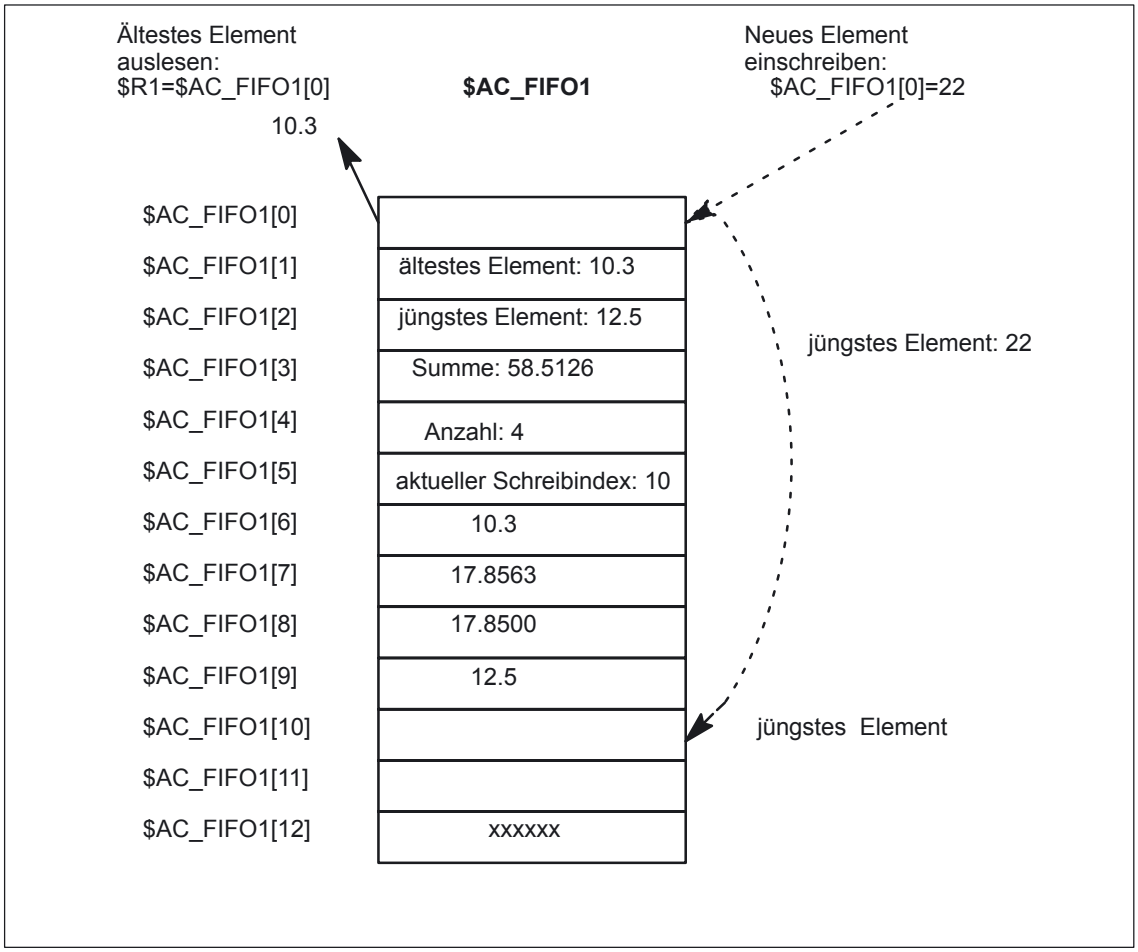


Bild 2-3 Beispiel FIFO–Variable

2.3.7 SRAM gespeicherte Systemvariablen (ab SW 6.3)

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: /IADC/ Inbetriebnahmehandbuch; NCU 840D und CCU 810D.

2.3.8 Bestimmung des Bahntangentenwinkels in Synchronaktionen

$AC_TANEB

Die in Synchronaktionen lesbare Systemvariable $AC_TANEB (**T**angent **AN**gel 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 Grad ausgegeben. Existiert kein Nachfolgesatz im Hautlauf, so wird der Winkel –180.0 Grad 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;
```

2.3.9 Bestimmung des aktuellen Override

Aktueller Override

Der aktuelle Override (NC–Anteil) kann mit den Systemvariablen:
$AA_OVR Achsial Override
$AC_OVR Bahnoverride
in Synchronaktionen gelesen und geschrieben werden.

PLC–Override

Der von der PLC vorgegebene Override wird für Synchronaktionen in den Systemvariablen:
$AA_PLC_OVR Achsial Override
$AC_PLC_OVR Bahnoverride
zum Lesen bereitgestellt.

Resultierender Override

Der resultierende Override wird für Synchronaktionen in den Systemvariablen:
$AA_TOTAL_OVR Achsial Override
$AC_TOTAL_OVR Bahnoverride
zum Lesen bereitgestellt.

Resultierender Override errechnet sich als:
$AA_OVR * $AA_PLC_OVR bzw.
$AC_OVR * $AC_PLC_OVR

2.3.10 Auslastungsauswertung über Zeitbedarf bei Synchronaktionen

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 das Maschinendatum MD11510 $MN_IPO_MAX_LOAD größer als 0 ist. Andernfalls geben die Variablen sowohl für SINUMERIK powerline als auch für solution line Systeme 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–Taktes.

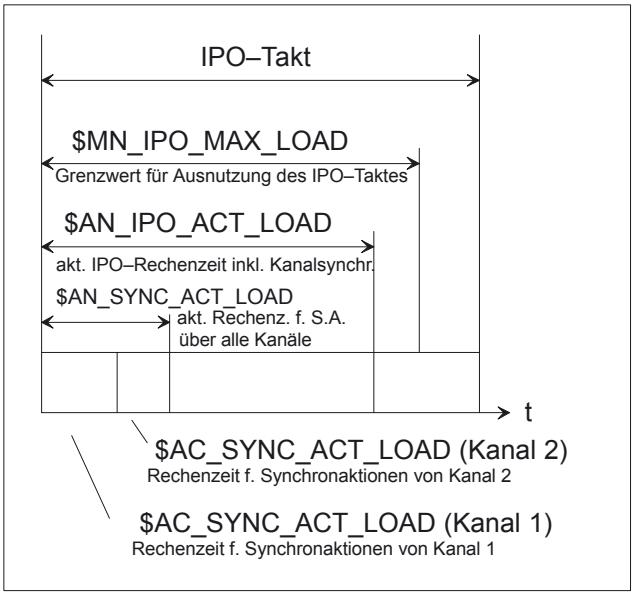


Bild 2-4 Zeitanteile der Synchronaktionen am IPO–Takt

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 (%).

Systemvariable	Bedeutung
$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.

Systemvariablen beschreibbar

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.

Programmier–beispiel

```
$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.

2.3.11 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”.

2.4 Aktionen in Synchronaktionen

Aktionen

Jede Synchronaktion enthält nach dem Aktionskennwort **DO ...**

– eine oder mehrere (max. 16) Aktionen oder einen Technologiezyklus (Als Oberbegriff wird im Weiteren **Aktionen** verwendet.)

die bei erfüllter Bedingung ausgeführt werden.

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 2-2 Aktionen in Synchronaktionen

... DO ...	**Bedeutung**	**Verweis**
Mxx Sxx Hxx	Hilfsfunktionsausgabe an PLC	2.4.1
SETAL(nr)	Alarm setzen, Reaktion auf Fehler	2.4.23
$V... = ... $A... = ... $AA_OFF = $AC_OVR = $AA_OVR = $AC_VC = $AA_VC = $$SN_SW_CAM_ ... $AC_FCT... $AA_TOFF =	Variable zuweisen (Servo–Werte) Variable zuweisen (Hauptlaufvariable) – Überlagerte Bewegung – Geschwindigkeitsbeeinflussung: Bahngeschwindigkeit Achsgeschwindigkeit add. Bahnvorschubkorrektur add. Korrekturwert der Achse Verändern von SW–Nockenpositionen (Settingdaten) und alle anderen SD Überschreiben von FCTDEF–Parametern – aufgeschaltete WZL–Korrekturen	2.4.2 2.4.3 2.4.4 2.4.8
 RDISABLE STOPREOF DELDTG FTOC SYNFCT ZYKL_T1 (z.B.)	Synchronaktionsprozeduren: Einlesesperre aktivieren Vorlaufstopp beenden Restweg löschen ohne Vorlaufstopp Online–Werkzeugkorrektur Polynomauswertung Aufruf von Technologiezyklen	 2.4.9 2.4.10 2.4.11 2.4.7 2.4.5 2.5
 $AA_OVR[x]= 0 ACHSE_X (z.B.) POS[u]= ... FA[u]= ... MOV[u]= >0 MOV[u] = <0 MOV[u] = =0 AXTOCHAN GET(Achse) RELEASE(Achse)	Steuerung von Positionierachsen: Sperren einer Achsbewegung Aufruf eines Achsprogrammes Positionieren Achsvorschub festlegen Kommando–Achsen kontinuierl. bewegen: – vorwärts – rückwärts – anhalten Achstausch aus einer Synchronaktion Achse für Achstausch anfordern Achse zum Achstausch freigeben	 2.4.12 2.4.13 2.4.13 2.4.14 2.4.15 ” ” ” 2.4.16 ” ”

Tabelle 2-2 Aktionen in Synchronaktionen

... DO ...	**Bedeutung**	**Verweis**
SPOS M3, M4, M5, S = $AA_OVR[S1]= 0	Spindeln: Positionieren Drehrichtung, Halt, Drehzahl Sperren der Spindelbewegung	2.4.18
PRESETON(,)	Istwertsetzen	2.4.19
	Mitschleppen und Kopplungen	2.4.20
LEADON LEADOF TRAILON TRAILOF	Kopplungen aktivieren/deaktivieren Folgeachse an Leitachse ankoppeln Kopplung aufheben Mitschleppen aktivieren/deaktivieren Asynchrones Mitschleppen ein Asynchrones Mitschleppen aus	2.4.20
Schlüsselwörter CPOF CPLON CPLOF CPLNUM CPLDEN CPLCID CPSETTYPE	Koppelmodule der Generische Kopplung Hauptlaufkopplungen aktivieren/deaktivieren: Aussschalten eines Koppelmoduls Einschalten einer Leitachse eines Koppelmoduls Ausschalten einer Leitachse eines Koppelmoduls Kopplungseigenschaften programmieren: Zähler des Koppelfaktors Nenner des Koppelfaktors Nummer der Kurventabelle Kopplungstyp der bestehenden Kopplungsarten	2.4.20
MEAWA MEAC	Messen ohne Restweglöschen zyklisches Messen	2.4.21
SETM CLEARM	Kanalsynchronisation: Setzen einer Wartemarke Löschen einer Wartemarke	2.4.22
LOCK UNLOCK RESET	Koordinierung zw. Synchronaktionen: – Synchronaktion / Technologiezyklus sperren – Synchronaktion / Technologiezyklus freigeben – Technologiezyklus rücksetzen	2.5.1

2.4.1 Ausgabe von M–, S– und H–Hilfsfunktionen an die PLC

Details zur Hilfsfunktionsausgabe im Allgemeinen finden Sie in:

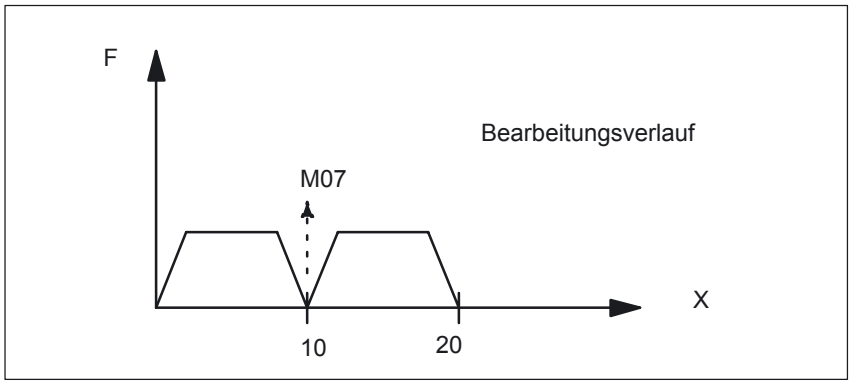
Literatur:
/FB1/ Funktionshandbuch Grundfunktionen; Hilfsfunktionsausgabe an PLC (H2)

Beispiele

Der Vorteile der Hilfsfunktionsausgabe aus Synchronaktionen wird am folgenden Beispiel deutlich: Kühlmittel an bestimmter Position einschalten

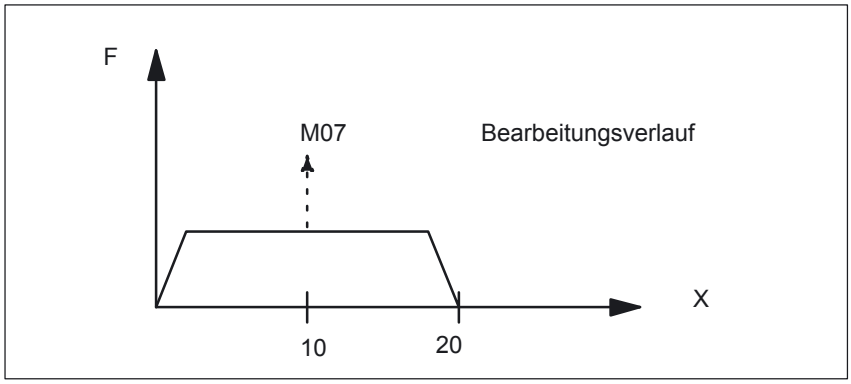
Lösung **ohne** Synchronaktion: 3 Sätze

```
N10 G1 X10 F150
N20 M07
N30 X20
```



Lösung **mit** Synchronaktion: 1 Satz

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



Hilfsfunktionsausgabe an die PLC

Als Synchronaktion 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.
Der im

MD11110 $MN_AUXFU_GROUP_SPEC (Hilfsfunktionsgruppen–spezifikation) bzw.
$MC_AUXFU_M_SYNC_TYPE (Ausgabezeitpunkt der M–Funktionen) /
$MC_AUXFU_S_SYNC_TYPE (Ausgabezeitpunkt der S–Funktionen) /
$MC_AUXFU_H_SYNC_TYPE (Ausgabezeitpunkt der H–Funktionen) /

ggf. definierte Ausgabezeitpunkt ist unwirksam.

Programmierung

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

Beispiel

```
WHEN $AA_IM[X] > 50 DO H15 S3000 M03
; wenn Istwert der X–Achse größer 50 wird, H15 ausgeben, neue Spindeldrehzahl, neue Drehrichtung einstellen
```

Einschränkung

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 Hilfsfunktionen je Synchronaktions–Satz oder Technologiezyklus–Satz:

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

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

```
WHEN ... DO M0                    ; Alarm
```

Erlaubt sind jedoch die Spindel–M–Funktionen: M3, M4, M5 und
M17 als Ende für einen Technologiezyklus.

Quittierung

Technologiezyklensätze (siehe Kap. 2.5) 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 Quittungsverhalten wurde auf weitere 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:–2 147 483 648 bis 2 147 483 647.

Literatur:
/FB1/ Funktionshandbuch Grundfunktionen; Hilfsfunktionsausgabe an PLC (H2)

2.4.2 Setzen (Schreiben) und Lesen von Hauptlaufvariablen

Schreiben

In Synchronaktionen können die Systemvariablen als Hauptlaufvariablen in Aktionen **geschrieben** werden, die in der Liste der Sytemvariablen in der 8. Zeile im Feld ”write:” mit X gekennzeichnet sind. Geschrieben werden auch im Hauptlauf

- 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 werden.

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: /LIS1/ Listen (Buch1).

Beispiele:
... DO **$$MN_MD_FILE_STYLE = 3** ; Maschinendatum setzen
... DO **$$SA_OSCILL_REVERSE_POS1 = 10** ; Settigdatum 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 im Hauptlauf

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

Hinweis

Maschinendaten und Settingdaten, die im Hauptlauf gelesen werden sollen, müssen mit **$$**.._... programmiert werden. Für Variablen, deren Inhalt sich nicht im Hauptlauf 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

2.4.3 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.

2.4.4 FCTDEF

Anwendung

Die in den folgenden Unterkapitel 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 2.4.7. Beispiele für lastabhängige Vorschübe und Abstandsregelung über Polynome finden Sie unter 2.4.5.

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. S. 2.4.2 .

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 sein können, werden per
MD28252 $MC_MM_NUM_FCTDEF_ELEMENTS
vorgegeben.

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 Systenvariablen geschrieben oder verändert werde. Der Gültigkeitsbereich des Polynoms wird durch die Grenzen $AC_FCTLL[n] und $AC_FCTUL[n] festgelegt.

Aufruf der Polynomaus–wertung

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).

2.4.5 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 Polynom finden Sie in 2.4.4.

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:

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

in den Bearbeitungsvorgang eingehen.

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 Bahnvorschubes

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.

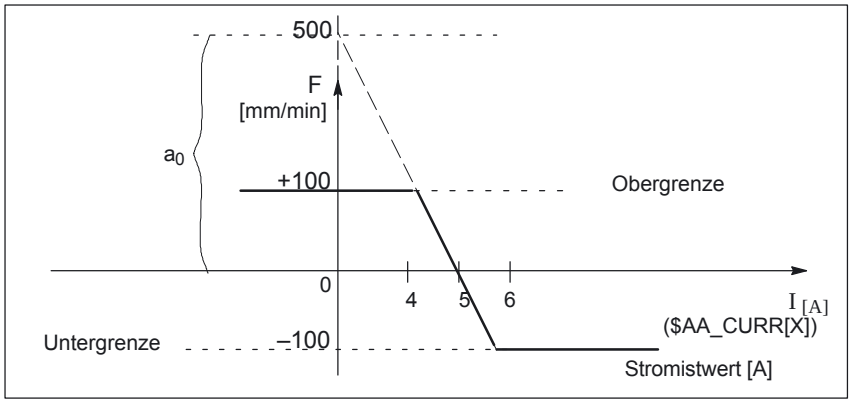


Bild 2-5 Beispiel additive Beeinflußung

Bestimmung der Koeffizienten s. auch 2.4.4:

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

$a_1 = -\frac{100\ mm}{1\ min \cdot A}$

$a_1 = -100 \Rightarrow$ Regelkonstante

$a_0 = -(-100) \cdot 5 = 500$

$a_2 = 0$ (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 Beispiel Bild 2-5 vollständig beschrieben.

Mit folgender Synchronaktion wird die *AC–Regelung* eingeschaltet:
ID = 1 DO **SYNFCT**(1, $AC_VC[x], $AA_LOAD[x])
; der additive Korrekturwert für den Vorschub der 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) **muliplizierte**. $F_{wirksam} = F_{programmiert} \cdot Faktor_{AC}$

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.

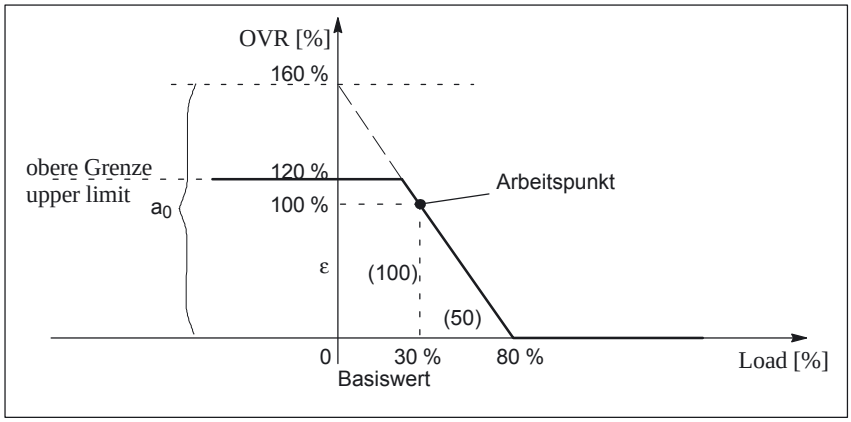


Bild 2-6 Beispiel multiplikative Beeinflußung

Bestimmung der Koeffizienten s. auch 2.4.4:

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

$a_1 = -\frac{100\ \%}{(80 - 30)\ \%} = -2$

$a_0 = 100 + (2 \cdot 30) = 160$
$a_2 = 0$ (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 Beispiel Bild 2-6 vollständig beschrieben.

Die dazugehörige Synchronaktion kann wie folgt lauten:
ID = 1 DO **SYNFCT**(2, $AC_OVR, $AA_LOAD[x])
; der Bahnoverride wird in jedem 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 durch:

MD36750 $MA_AA_OFF_MODE

festgelegt.

0: proportionale Bewertung
1: integrale Bewertung

Ab SW–Stand 4 ist es möglich, den absolut zu korrigierenden Wert (Hauptlaufvariable–Ausgang) auf den im Settingdatum

SD43350 $SA_AA_OFF_LIMIT

hinterlegten Wert zubegrenzen.
Ob die Begrenzung erreicht wird, kann durch Auswertung der achsspezifischen Systemvariablen

$AA_OFF_LIMIT[Achse]

in einer (weiteren) Synchronaktion abgefragt werden.

Wert –1: Limit des Korrekturwertes wurde in negativer Richtung erreicht.
Wert 1: Limit des Korrekturwertes wurde in positver Richtung erreicht.
Wert 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. S. u.

Beispiel

Aufgabe:
Abstandsregelung als Funktion eines Sensorsignales 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, s.2.4.12) oder Alarm setzen s. 2.4.23 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 Funktrion ”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

In 6.3.1 finden Sie ein Beispiel mit dynamischer Anpassung einer Grenze des Polynoms bei der AC–Regelung (Abstandsregelung). In 6.3.2 finden Sie ein Beispiel für AC–Regelung des Bahnvorschubes.

Abstandsregelung

Der Abstandswert wird über das MD36750 $MA_AA_OFF_MODE[V]=1 integrierend verrechnet. Es wirkt im Basiskoordinatensystem, d. h. vor der Transformation. Damit kann man es als Abstandsregelung in Orientierungsrichtung verwenden (nach Frameanwahl mit TOFRAME).

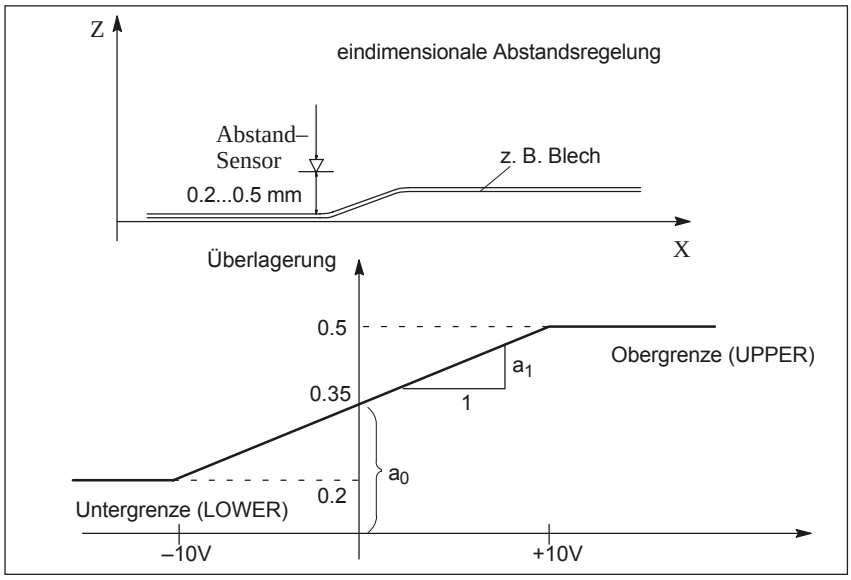


Bild 2-7 Abstandsregelung

```
%_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 $AA_OVR[X] = 0
                                  ; Bei Überschreitung des Grenzbereiches 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
                                  ; MD 36750 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
```

2.4.6 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.

Mit dem achsialen MD36750 $MA_AA_OFF_MODE wird die Art der Verrechnung wie folgt festgelegt:

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 MD 32070: $MA_CORR_VELO.

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 über das Maschinendatum MD36750 $MA_AA_OFF_MODE das Verhalten von RESET eingesetellt werden.

Bit1 = 0: $AA_OFF wird bei RESET abgewählt
Bit1 = 1: $AA_OFF bleibt über RESET hinaus erhalten

In der Betriebsart JOG kann eine überlagerte Bewegung stattfinden

Auch in der Betriebsart JOG kann bei einer Änderung von $AA_OFF eine Interpolation des Positionsoffset als überlagerte Bewegung über das Maschinendatum MD36750 $MA_AA_OFF_MODE eingesetellt werden.

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 errreicht ist.

Ab Software–Stand 3.2 erfolgt mit Einführung einer für Synchronaktionen zugelassenen $$–Hauptvariablen ein Vergleich der Synchronisations– bedingungen 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 DROF wird beim Satzsuchlauf mit aufgesammelt und in einem Aktionssatz ausgegeben. Dabei werden in den letzten vom Suchlauf behandelten Satz mit CORROF oder DROF alle abgwählten 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 den Satzsuchlauf via Programmtest ”SERUPRO” möglich. Weitere Details zu diesen Satzsuchlaufs sind beschrieben in:

/FB1/ Funktionshandbuch Grundfunktionen; BAG, Kanal, Programmmbet r. (K1)

- **DRF–Verschiebungen achsspezifischmit CORROF abwählen**
  Mit CORROF sind DRF–Verschiebungen für die einzelne 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.

2.4.7 Online–Werkzeugkorrektur FTOC

Online–Werkzeug–korrektur

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.

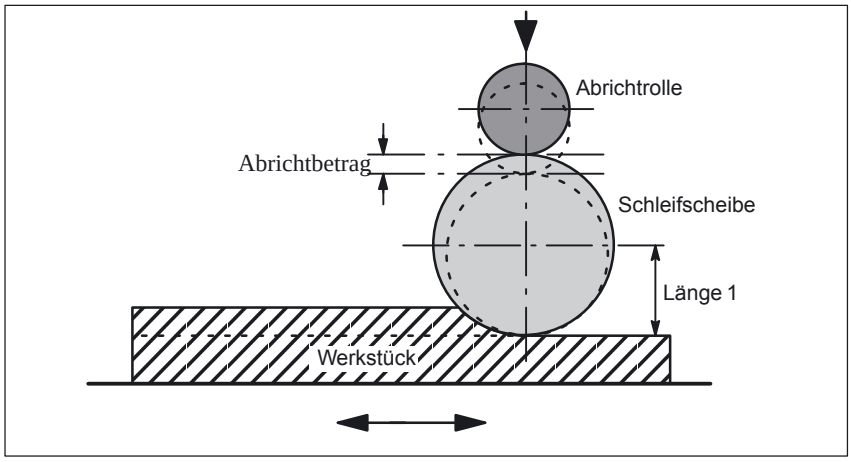


Bild 2-8 Abrichten während der Bearbeitung mit einer Abrichtrolle

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 (s. 2.4.4) 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–Hauptvariable_lesen,      ;Bezugswert
            Länge 1_2_3,
            Kanalnummer,
            Spindelnummer)
```

Parameter

Polynom–Nr:	Nummer der zuvor mit FCTDEF parametrierten Funktion.
Real–Hauptvariable_lesen:	Es sind alle bei 8.1.1 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.

2.4.8 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 über das Maschinendatum MD21194 $MC_TOFF_VELO eingestellt werden.

Dementsprechend kann über das Maschinendatum MD21196 $MC_TOFF_ACCEL die **Beschleunigung** eingestellt werden.

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 kann über das 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–Werrkzeuglängenkorrektur

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–Werrkzeuglängenkorrektur

```
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 Postion 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” (DB21, ... DBX318.3) auf 1 gesetzt. Nach Abwahl wird das NST ”TOFF aktiv” DB21, ... DBX318.2 auf 0 gesetzt.

Hinweis

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

Werkzeuglängen–offset bei RESET und POWER ON

Über das Maschinendatum MD21190 MC_TOFF_MODE kann das Verhalten bei RESET mit Bit 0 eingestellt werden. Der Werkzeuglängenoffset $AA_TOFF[ ] wird bei MD21190 MC_TOFF_MODE

- 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 über das MD20110 MC_RESET_MODE_MASK eine Transformation oder ein Orientierbarer Werkzeugträger **nach** RESET abgewählt werden. 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 eineTransformation 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 Ebenenenwechsel
  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.

2.4.9 RDISABLE

Programmierte Einlesesperre RDISABLE

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.
Anwendung: Damit kann z. B. abhängig von externen Eingängen das Programm im Interpolationstakt gestartet werden.

Beispiel RDISABLE

Programmierte Einlesesperre

```
WHENEVER $A_INA[2]<7000 DO RDISABLE
...
N10 G01 X10      ;Am Ende von N10 wirkt RDISABLE, wenn während seiner
                  Bearbeitung die Bedingung erfüllt ist
N20 Y20
```

Wenn die Spannung 7V am Eingang 2 unterschreitet, wird die Programmfortsetzung angehalten (Annahme: Wert 1000 entspricht 1V).
Anwendung dieser Lösung z.B: Einlesesperre bis Hindernis aus dem Weg geräumt ist.

2.4.10 STOPREOF

Beendigung des Vorlaufstopp STOPREOF

Eine Bewegungssynchonaktion mit einem STOPREOF–Befehl hebt den bestehenden Vorlaufstopp 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 Vorlaufstopp. Wenn die Spannung 5V am Eingang 7 überschreitet, springe vorwärts bis zum Label 1 (Annahme: Wert 1000 entspricht 1V).

2.4.11 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 achsialen Restweg.

(Annahme: Wert 1000 entspricht 1V).

2.4.12 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 digitialen Eingang
```

Hinweis

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

2.4.13 Starten von Kommandoachsen

Einführung

Achsen können auch vollkommen asynchron zum Teileprogramm aus Synchronaktionen positioniert, gestartet gestoppt werden. Diese Art der Programmierung empfiehlt sich für zyklische Abläufe oder Abläufe, die sehr stark ereignisgesteuert sind. Achsen, die aus Synchronaktionen gestartet werden, heissen **Kommandoachsen.**

Kontrolle vom PLC

Autarke Einzelachsvorgänge

Eine vom Hauptlauf interpolierende Kommandoachse (gestartet über statische Synchronaktionen) reagiert unabhängig vom NC–Programm bei NC–STOP, Alarmbehandlung, Programmende, Programmbeeinflussungen und RESET, wenn die Kontrolle der Kommandoachse vom PLC übernommen wurde.

Die Kontrolle über die Kommandoachse erfolgt über die axiale VDI–Nahtstelle (PLC→NCK) mit den NST ”PLC kontrolliert Achse” (DB31, ... DBX28.7) == 1

Weitere Informationen über den genauen Ablauf der verschiedenen Schritte die Kontrolle der Kommandoachse an die PLC zu übertragen siehe bitte:
Literatur:
/FB2/ Funktionshandbuch Erweiterungsfunktionen; Positionierachsen (P2).

Randbedingung

Eine Achse kann nicht gleichzeitig aus dem Teileprogramm und aus Synchronaktionen bewegt werden. Zeitlich nacheinander ist dies möglich.
Wartezeiten können auftreten, wenn eine Achse nach einer Bewegung aus Synchronaktionen wieder im Teileprogramm programmiert wird.

Hinweis

Das MD30450 $MA_IS_CONCURRENT_POS_AX gibt darüber Auskunft, ob die Achse hauptsächlich als Kommandoachse oder für Programmierung durch das Teileprogramm vorgesehen ist:
0: keine konkurrierende Achse
1: konkurrierende Achse (Kommando–Achse)

Beispiel 1

```
...
ID=1 EVERY $A_IN[1]==1 DO POS[X]=100
...
```

Beispiel 2

Eine Achsbewegung kann als Technologiezyklus ausgelöst werden.(S. 2.5)
Hauptprogramm:

```
...
ID=2 EVERY $A_IN[1]==1 DO ACHSE_X
...
```

Achsprogramm:

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

Programmierung

Positionierachsbewegungen werden in der Synchronaktion wie aus dem Teileprogramm programmiert:

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

Die programmierte Position wird inch oder metrisch bewertet entsprechend der im aktuell bearbeiteten Teileprogrammsatz gültigen Einstellung G70 oder G71.

G70/G71 sowie G700/G710 können auch in den Synchronaktionen direkt programmiert werden.

Damit kann die Inch–, Metrisch–Bewertung einer Kommandoachs–Bewegung unabhängig von der Programmierung im Teileprogramm festgelegt werden.

ID = 1 WHENEVER $A_OUT[1] ==1 DO **G710** POS[X]=10

ID = 2 EVERY **G710** $AA_IM[Z] >100 DO **G700** POS[Z2]=10

Hinweis

Es sind nur **G70, G71, G700, G710** in Synchronaktionen programmierbar! Vergl. Kapitel 2.1.

G–Funktionen, die im Synchronaktionssatz programmiert werden, wirken nur auf die Synchronaktion oder im Technologiezyklus. Sie haben keine Auswirkung auf die folgenden Sätze im Teileprogramm.

Literatur: /PG/ Programmierhandbuch Grundlagen; Kap. ”Wegangaben”

Endposition absolut / inkrementell

Die Endposition kann absolut oder inkrementell programmiert werden. Je nachdem, ob im gerade aktiven Satz des Hauptprogramms G90 oder G91 aktiv ist, wird die Position absolut oder inkrementell verfahren.
Es kann auch bei der Programmierung explizit bestimmt werden, ob der Wert absolut oder inkrementell programmiert wird:

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

Codierte Positionen sind in Maschinendaten hinterlegte Werte.

Beispiel 1 fester Wert

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

; Wenn Ereignis eintritt, U–Achse um 10 weiterpositionieren

Beispiel 2 aktueller Wert

Der zu verfahrende Weg wird in Echtzeit aus einer Hauptlaufvariablen gebildet:

ID = 1 EVERY $AA_IM[B] > 75 DO **POS[U]=$AA_MW[V]–$AA_IM[W] + 13.5**

Axiale Frames

Das Verhalten von Synchronaktionen und axialen Frames erläutern die folgenden Abschnitte:

Wirkung

Bei Positionierbewegungen aus Synchronaktionen wirken die achsialen Verschiebungen, Skalierungen und Spiegelungen des programmierbaren und einstellbaren Frames (G54 usw.) sowie Werkzeuglängenkorrekturen.

Es wirkt jeweils der im aktiven Satz wirksame Frame. Ist im aktiven Satz eine Drehung aktiv, so wird das Auslösen einer Positionierbewegung aus der Synchronaktion mit Alarm abgelehnt.

Beispiel:

```
TRANS X20
IDS= 1 EVERY $A_IN==1 DO POS[X]=40
G1 Y100          ;Wenn der Eingang gesetzt wird, wird X auf 60 positioniert
...
TRANS X–10
G1 Y10           ; Wenn der Eingang gesetzt wird, wird X auf 30 positioniert
```

Unterdrückung

Die Wirksamkeit von Frames und Werkzeuglängen kann unterdrückt werden mit

MD32074 $MA_FRAME_OR_CORRPOS_NOTALLOWED

axiale Frames unterdücken

Auf eine Kommandoachse sind axiale Frames nicht wirksam, die inkrementell auf Teilungspositionen verfahren. Deshalb wird im

MD32074 $MA_FRAME_OR_CORRPOS_NOTALLOWED[AX4] das Bit 9 =1 gesetzt und die Kommandoachse mit JOG positioniert.

Beispiel:

```
RANS A=0,001
POS[A]=CAC(2)                    ; Achse fährt auf Position 180.001 Grad

; Für die Kommandoachse ist das axiale Frame nicht wirksam
; MD32074 $MA_FRAME_OR_CORRPOS_NOTALLOWED[AX4] = 'H0020'

WHEN TRUE DO POS[A]=CIC(–1) ; Achse fährt auf Position 180.000 Grad.
```

Hinweis

Wird einen Kommandoachse inkrementell auf Teilungspositionen verfahren, dann sind axiale Frames generell für diese Kommandoachse **nicht** wirksam.

2.4.14 Axialer Vorschub aus Synchronaktionen

Vorschübe

Zusätzlich zur Endposition kann ein axsialer 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 über das axiale
MD32060 $MA_POS_AX_VELO
vorgegeben.

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.
Das Settingdatum kann per Bedienung oder von PLC, sowie aus dem Teileprogramm verändert werden. Synchron zumTeileprogrammkontext 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 axsiale Vorschub aus Bewegungssynchronaktionen wird nicht als Hilfsfunktion an die PLC ausgegeben. Parallele axsiale Technologiezyklen würden sich sonst gegenseitig blockieren.

2.4.15 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**. (siehe oben). Ist kein axialer Vorschub programmiert, so ergibt sich der Wert aus einer evtl. bereits aus Synchronaktionen aktivierten Achs–Bewegung oder aus der über das axiale MD32060 $MA_POS_AX_VELO eingestellten Achsgeschwindigkeit.

Beispiel

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

2.4.16 Achstausch aus Synchronaktionen

Anwendung

Für einen Werkzeugwechsel können die betreffenden Kommandoachsen als Aktion einer Synchronaktionen 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 **GET[Achse]**

Achse freigeben **RELEASE[Achse]**

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 dieser Achse besitzt, und vom eigentlichen Zustand für den zulässigen Achstausch, ergibt sich aus der Synchronaktion ein unterschiedlicher Ablauf.

Aus einer Sychronaktion kann eine Achse zum Aktivierungszeitpunkt mit **GET[Achse]** angefordert werden, wenn

- ein anderer Kanal das Schreib– bzw. Interpolationsrecht für die Achse hat.
- ein anderer Kanal das Schreibrecht für die Achse hat und die Achse bereits als neutrale Achse angefordert wurde.
- 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

Randbedingungen:

Eine ausschließlich von der PLC kontrollierte Achse kann nicht dem NC–Programm zugeordnet werden. Ebenso eine fest zugeordnete PLC–Achse.

Aus einer Sychronaktion 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 Schreib– bzw. Interpolationsrecht dieser Achse hat.

Achse aus anderen 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 neutrale 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 kontroliert

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 abhängig vom Bit 0 im Maschinendatum MD 10722: AXCHANGE_MASK für einen automatischen Achstausch zwischen Kanälen gesperrt (Bit 0 == 0).

Dies entspricht $AA_AXCHANGE_STAT[Achse] == 1.

Achse ist als Kommandoachse aktiv / konkurrierende Positionierachse

Ist die Achse **als Kommandoachse** bzw. Pendelachse 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 abhängig vom Bit 0 im Maschinendatum MD 10722: AXCHANGE_MASK für einen automatischen Achstausch zwischen Kanälen gesperrt (Bit 0 == 0).

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 Achs–tausch 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 Achs–tausch in einen anderen Kanal freigegeben.

Dies hat **einen** Reorganisationsvorgang zur Folge, falls die Achse noch dem NC–Programm zugeordnet ist.

Freizugebene Achse ist bereits neutrale Achse:
Ist die Achse bereits im Zustand neutrale Achse $AA_AXCHANGE_TYP[<Achse>] == 3 oder als Kommando– bzw. Pendelachse oder konkorrierende Positionierachse (PLC–Achse), $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 z.B. aktive Achskopplung oder Transformation, aktives Schnellabheben, JOG–Anforderung, rotierendes Frame mit möglicher PLC– / Kommando– / Pendelachsbewegung der Achse an den Kanal vorliegt.

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 weitergeschalten, 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

Die 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
N110 G4 F0.1
    WHEN TRUE DO GET(Z)        ; Z–Achse wird wieder NC–Programm–Achse
    WHENEVER $AA_TYP[Z] <> 1 DO RDISABLE       ; Einlesesperre bis
                                               ; 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

Für eine Achse bei automatisch erzeugtem GET ein GETD abgegeben wenn MD30552 AUTO_GET_TYPE = 2. Die Achse U ist im 1. und 2. Kanal bekannt und aktuell hat der Kanal 1 das Interpolationrecht, 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.

2.4.17 Achse einem anderen Kanal übergeben (AXTOCHAN)

Funktion

Aus einer Sychronaktion 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. Es ist somit möglich, eine Achse einem anderen Kanal zu übergeben.

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

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

Randbedingungen AXTOCHAN

Eine konkurrierende Positionierachse kann den Kanal nicht wechseln. Dies ist auch nicht über die Vorgabe von der VDI–Nahtstelle möglich.

Literatur: /FB2/ Funktionshandbuch Erweiterungsfunktionen;
Positionierachsen (P2)
BAGs, Kanäle, Achstausch (K5).

Hinweis

Eine ausschließlich von der PLC kontrollierte Achse kann nicht dem NC–Programm zugeordnet werden. Soll diese Achse über statische Synchronaktionen verfahren werden, so ist hierfür Synchronaktion Stufe 2 notwendig.

2.4.18 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 digitialen 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 ($A_IN[1]==0)  DO  SPOS=0
                                                 ; Spindel positionieren
```

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 abhängig vom Settingdatum: SD43400 $SA_WORKAREA_PLUS_ENABLE berücksichtigt.

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

Beschleunigung korrigieren

Mit ACC[Achse]=0..200 kann die im MD32300 $MA_MAX_AX_ACCEL vorgegebene Beschleunigung in einem Bereich von 0% bis 200% verändert werden:

ACC[Achse]=<Wert> (wirkt im Teileprogramm und in Synchronaktionen).

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 digitialer 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 entgegengesetzte 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

Die mit x gekennzeichneten Übergänge sind zulässig:

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		
drehzahlgeregelte Spindel				x	x	x		
Achse in Bewegung								
Achsbetrieb	x	x	x				x	x
lagegeregelte Spindel								
drehzahlgeregelte Spindel				x	x	x		

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 Achs–kopplungen

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

– siehe Kapitel 2.4.20 ”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.

2.4.19 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.
z.B:

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 Kapitel 6.7.3 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.

2.4.20 Mitschleppen und Kopplungen aktivieren, deaktivieren

Einführung

In der Funktionsbeschreibung:

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

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
  Von den für Teileprogramme möglichen Leitwertkopplungen:
  - Achsleitwertkopplung
  - Bahnleitwertkopplung

    stehen für die Nutzung in Synchronaktionen nur Achsleitwertkopplungen zur Verfügung.
- Elektronsches Getriebe
  Mit Hilfe der Funktion ”Elektronisches Getriebe” kann die Bewegung einer Folgeachse FA abhängig von bis zu fünf Leitachsen LA interpoliert werden. Ein Getriebeverband kann aus dem Teileprogramm:
  - definiert,
  - eingeschaltet,
  - ausgeschaltet,
  - gelöscht werden.
- Generische Kopplung
  Mit Hilfe eines Koppelmoduls kann die Bewegung einer Achse (→ FA Folgeachse) abhängig von anderen Achsen (→ Leitachsen) interpoliert werden.

  Koppelmodule können im Teileprogrammm und Synchronaktionen implizit 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
  - Koppelfaktor oder
  - Kurventabelle definiert.

Über Schlüsselwörter kann jede Kopplungseigenschaft der Generischen Kopplung programmiert werden.
In Synchronktionen 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 imTeileprogramm 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 optionale Parameter für Skalierung: – Folgeachse FA, – Leitachse LA)

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 optionaler Parameter für Skalierung: – Folgeachse FA, – Leitachse LA)

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

LEADON

Achsleitwertkopplung aus Synchronktionen

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 Achsleitwertkopplung 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

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 6.7.3 ”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 dem Befehl

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 bei dieser Art von Kopplung somit nicht möglich.

Weitere Erläuterungen zum Achstausch in Synchronaktionen siehe Kapitel 2.4.16 Achstausch aus Synchronaktionen; GET/RELEASE[Achse].

Kopplung aktivieren/deaktivieren

Die CP–Schlüsserwörter 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 Schlüsselwort

CPOF=<FAx>

werden alle Leitachsen zur Folgeachse in Synchronaktionen deaktiviert.

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 mit:

CPLCTID[FAx, LAx]= <Wert>

Beispiele

Programmierung mit Schlüsselwörtern in Synchronaktionen:

Beispiel 1:
Definition einer Achskopplung mit einer Leitachse
DO CPDEF=Y CPLDEF[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]=2 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 WHEN TRUE DO CPLON[X]=Y CPLNUM[X,Y]
N15 Y=100 F100
N20 WHEN TRUE DO CPOF=X CPLON[X]=Y CPLNUM[X, Y]=3

In diesem Beispiel wird das in N10 aktive Koppelmodul X in N20 zuerst ausgeschaltet und gelöscht. Mit CPLON wird das 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 CPOF=X MOV[X]=1

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

bisherige Kopplungstypen verwenden

Bestehende Koppkungsarten TRAIL, LEAD, EG und COUP

Wird eine Voreinstellung der bisher bestehenden Kopplungsarten wie Mitschleppen, Leitwertkopllung, 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 (Standardwert)
- ”TRAIL” Kopplungstyp ”Mitschleppen”
- ”LEAD” Kopplungstyp ”Leitwertkopplung”
- ”EG” Kopplungstyp ”Elektronisches Getriebe”
- ”COUP” Kopplungstyp ”Synchronspindel”

2.4.21 Messen aus Synchronaktionen

Einführung

Von den für Teileprogramme verfügbaren Meßfunktionen:
MEAS, MEAW, MEASA, MEAWA, MEAC

Literatur:
/PGA/ Programmierhandbuch Arbeitsvorbereitung
/FB2/ Funktionshandbuch Erweiterungsfunktionen; Messen (M5)

stehen in Synchronaktionen nur folgende zur Verfügung:

- MEAWA achsiales Messen ohne Restweglöschen
- MEAC achsiales, 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)
; Achsiales Messen ohne Restweglöschen einschalten

MEAC[Achse]=(Modus, Meßspeicher,Triggerereignis_1, Triggerereignis_2,
Triggerereignis_3, Triggerereignis_4)
; achsiales, kontinuierliches Messen einschalten

Achse: Achse, für die gemessen wird

Tabelle 2-3 Modus–Bedeutungen:

Zehnerdekade	Einerdekade	Bedeutung
	0	Meßauftrag abbrechen
	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]=(, , , ,) ;Achsiales Messen ohne Restweglöschen

Restweglöschen kann bei Bedarf in der Synchronaktion explizit aufgerufen werden. S. 2.4.11 und Beispiel unten.

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

Programmierung:
Die Programmierung entspricht der 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 (S. 2.3.6.) 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
N1    MEAC[X]=( 1, 1, 1) POS[X]=100
                                   ; Modus = 1, gleichzeitig
                                   ; 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
      DO $R[$AC_MARKER[1]]= $AC_FIFO1[0] $AC_MARKER[1]=$AC_MARKER[1]+1
                                   ;Synchronaktion als Prüfung:
                                   ; wenn 1 oder mehr Meßwerte in FIFO–Variable 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
      DO  MEAC[X]=(0) DELDTG(X)             ; Schlußbedingung als Synchronaktion:
                                            ; 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 Trigger–ereignisse 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 folgende 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

Zustand	Verhalten
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.

2.4.22 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).

2.4.23 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 in Kapitel 2.4 ”Aktionen in Synchronaktionen” aufgeführte Aktionen

Beispiel Alarm setzen

```
ID=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/ Programmieranleitung Zyklen.

2.4.24 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 Funktionsbeschreibung Grundmaschine: Nahtstellensignale von Achse/Spindel.

Verfügbarkeit

Die Werte zur Maschinenwartung stehen zur Verfügung, wenn das globale NCK–Maschhinendatum MD18860 $MN_MM_MAINTENANCE_MON gesetzt ist und je interessierede Achse mit achsspezifischen Maschinendaten angegeben wurde, welche Daten bereitgestellt werden sollen.

Das MD18860 $MN_MM_MAINTENANCE_MON sorgt für die Aktivierung der Funktion und die Bereitstellung des Speichers für die in den achsspezifischen Maschinendatum angegebenen Werte. Änderungen von MD18860 $MN_MM_MAINTENANCE_MON wirken mit Power On.

Achsspezifische Werte:

Im MD33060 $MA_MAINTENANCE_DATA kann bitcodiert angegeben werden:

Bit 0:	Gesamtverfahrweg, Gesamtverfahrzeit und die Anzahl der Verfahrtvorgänge der Achse
Bit 1:	Gesamtverfahrweg, Gesamtverfahrzeit und die Anzahl der Verfahrtvorgänge 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

MD33060 $MA_MAINTENANCE_DATA[0]=1
MD33060 $MA_MAINTENANCE_DATA[1]=1
MD33060 $MA_MAINTENANCE_DATA[2]=1

aktiviert die Systemvariablen für Gesamtverfahrweg, Gesamtverfahrzeit und die Anzahl der Verfahrtvorgänge für die ersten 3 Achsen

Systemvariablen

Die Systemvariablen

$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

können aus dem Teileprogramm und aus Synchronaktionen gelesen werden.

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
```

2.5 Aufruf von Technologiezyklen

Definition

Ein Technologiezyklus ist eine Folge von Aktionen, die sequentiell im Interpolationstakt abgearbeitet werden. Die in 2.4 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 muß der Anwender aus den folgenden Möglichkeiten die geeignetere wählen:

- 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, S. 2.4.13) dauern **mehrere** Interpolationstakte. Ist die Funktion beendet (z.B. Genauhalt bei Positionieren einer Achse), so wird im darauffolgenden 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. Bild 2-9 kennzeichnet beispielhaft, welche Aktionen eintaktig und welche mehrtaktig sind.

Anwendung

Mit Technologiezyklen ist es beispielweise 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.

Das Programmende wird mit M02 / M17 / M30 / RET programmiert.

Suchpfad

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

Beispiel:

```
...
ID=1 EVERY $AA_IM[Y]>=10  DO AX_X                ; AX_X Unterprogramm–
                                                 ; name für Achsprogramm für X–Achse
```

```
AX_X:                ; Achsprogramm
POS[X]=$R[7] FA[X]=377
$A_OUT[1]=1
POS[X]=R10
POS[X]=–90
M30
```

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.

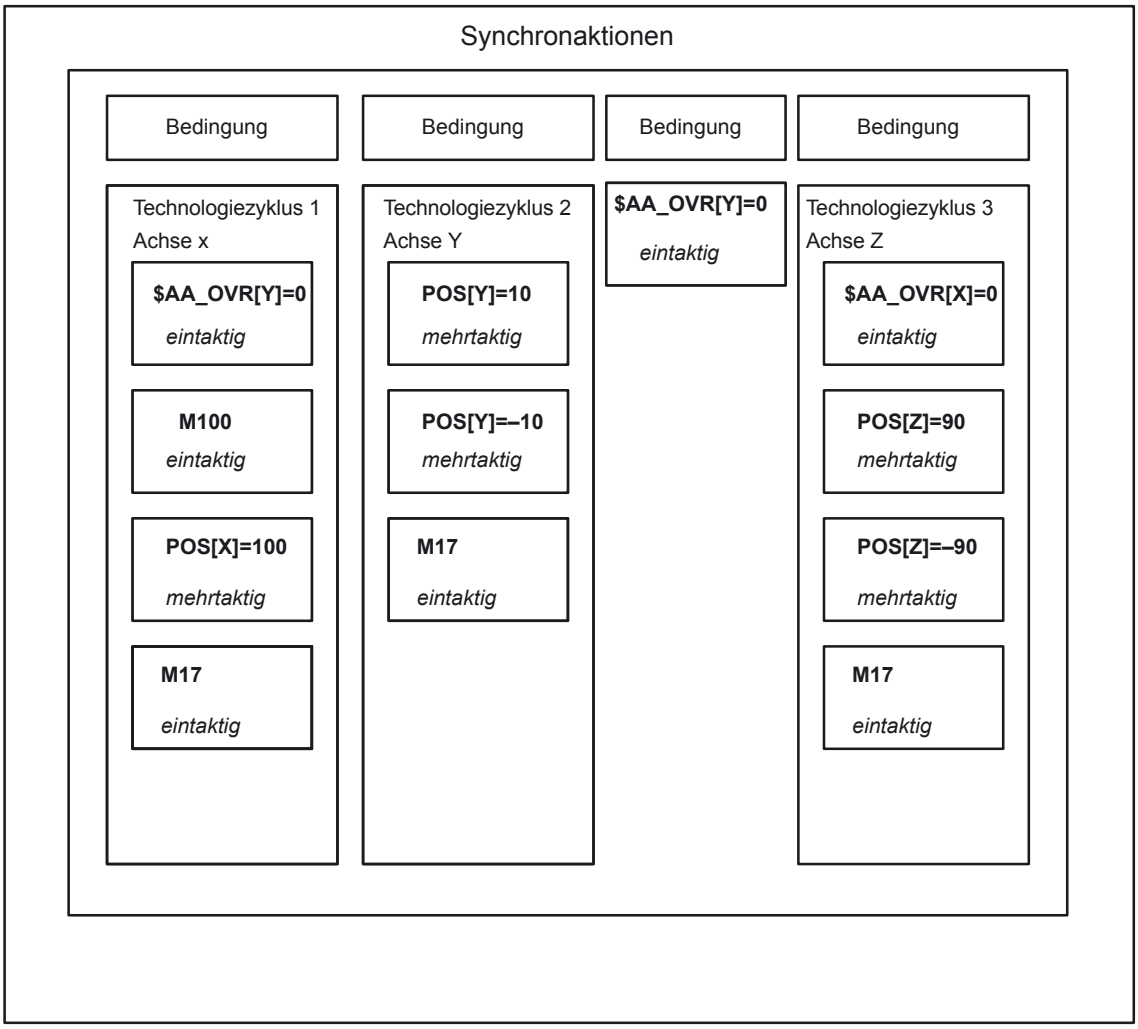


Bild 2-9 Mehrere Technologiezyklen

Beispiel (2) für koordinierte Achsbewegungen:
Durch Setzen digitaler NC–Eingänge werden verschiedene Achs–Programme gestartet.

Hauptprogramm:

```
...
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 AA_OVR[Y]=0
ID=4 WHEN $A_IN[4]==1 DO ACHSE_Z
M30
```

Achsprogramme:

ACHSE_X:

```
$AA_OVR[Y]=0
M100
POS[X]=100
M17
```

ACHSE_Y:

```
POS[Y]=10
POS[Y]=–10
M17
```

ACHSE_Z:

```
$AA_OVR[X]=0
POS[Z]=90
POS[Z]=–90
M17
```

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

Beeinflussung von Technologiezyklen

Die Technologiezyklen / Synchronaktionen werden über die Identifikationsnummer der Synchronaktionen beeinflußt, in denen sie als Aktion angegeben sind:

Mittel zur Koordinierung

Schlüsselwort	Bedeutung	TP	SA
	Aufruf zulässig im Teileprogramm Aufruf zulässig in Synchr. Aktion / Technologiezyklus	+	 +
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.	+	

- LOCK(ID), UNLOCK(ID) durch PLC siehe Kapitel 2.6.1 ”Beeinflussung von PLC”.

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.

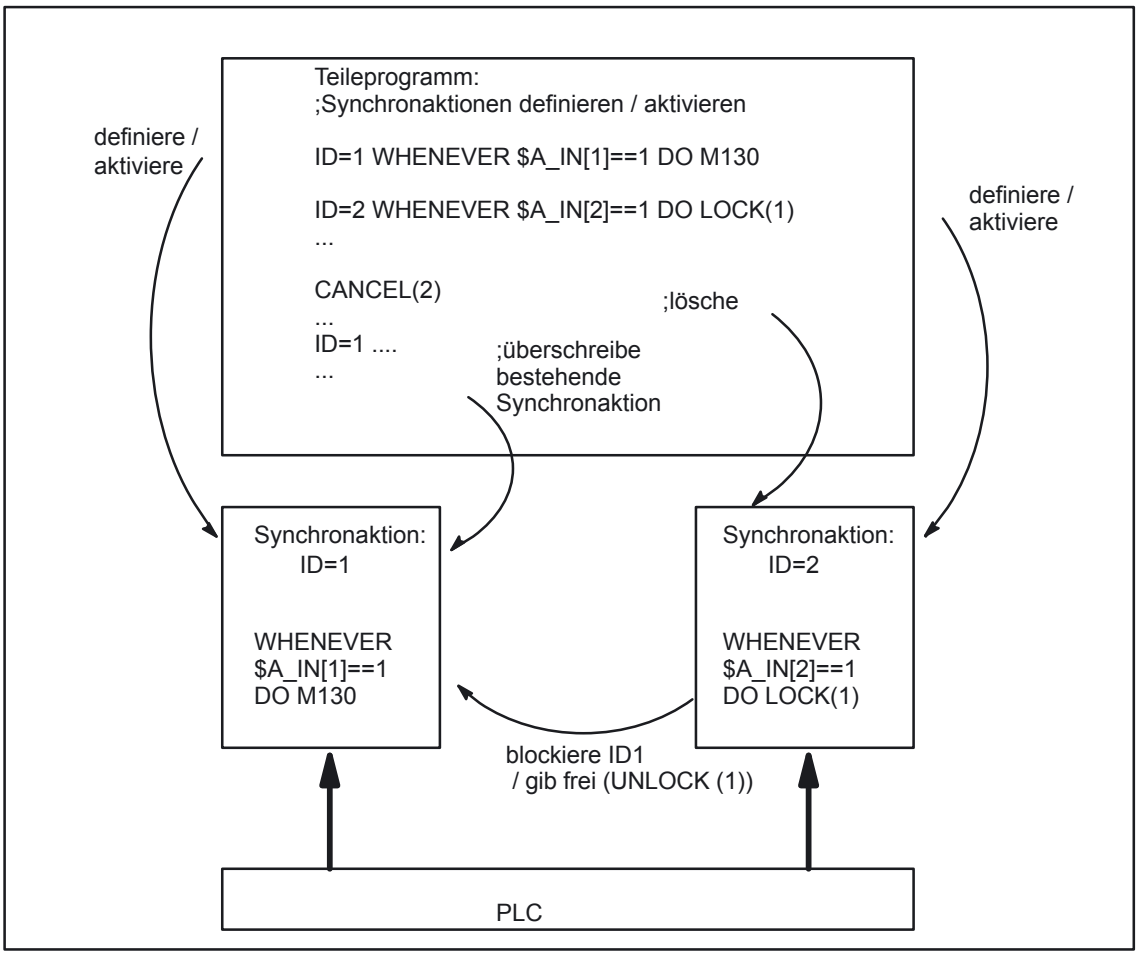


Bild 2-10 Anlegen / Verriegeln modaler Synchronaktionen / Löschen

2.6 Beeinflussung und Schutz von Synchronaktionen

2.6.1 Beeinflussung von 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 Nahtstelle:
DB21–30, DBB308–315
durch die NC mit 1 gekennzeichnet. Geschützte Synchronaktionen sind nie als sperrbar gekennzeichnet. SieheKapitel 2.6.2. ”Geschützte Synchronaktionen”.

Alle Synchronaktionen sperren

Durch Setzen von DB 21–30, DBB1 Bit 2 durch das PLC–Anwendungsprogramm können alle modalen Synchronaktionen, die in der NC bereits definiert und gespeichert sind, von der Aktivierung ausgeschlossen werden. Eine Ausnahme bilden geschützte Synchronaktionen siehe 2.6.2.
Durch Setzen von DB 21–30, DBB1 Bit 2 auf 0 wird die pauschale Sperrung durch PLC wieder aufgehoben.

Benutzung gezielte Sperren

Für die ersten 64 IDs (1–64) ist in der PLC–Nahtstelle je ein Bit reserviert. (DB 21–30, DBB 300 Bit 0 bis DB21–30 DBB 307 Bit 7).

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

Aufhebung gezielter Sperren

Durch Setzen des der ID–, IDS–Nummer entsprechenden Bits auf 0 in
DB 21–30, DBB 300 Bit 0 bis DB 21–30, DBB 307 Bit 7
wird eine zuvor gesperrte Synchronaktion wieder von PLC freigegeben.

Aktualisieren der gezielten Sperren

Wenn das PLC–Anwenderprogramm im Bereich von DB 21–30, DBB 300 Bit 0 bis DB 21–30, DBB 307 Bit 7 Änderungen vorgenommen hat, muss es diese mit DB 21–30 DBX280.1 aktivieren.

Rückmeldung gezieltes Sperren

Wenn von NCK die gezielten Sperren aktiviert wurden, wird dies in DB 21–30 DBX.281.1 signalisiert.

Literatur: /LIS2/ Listen (Buch2); Nahtstelle solution line bzw. power line.

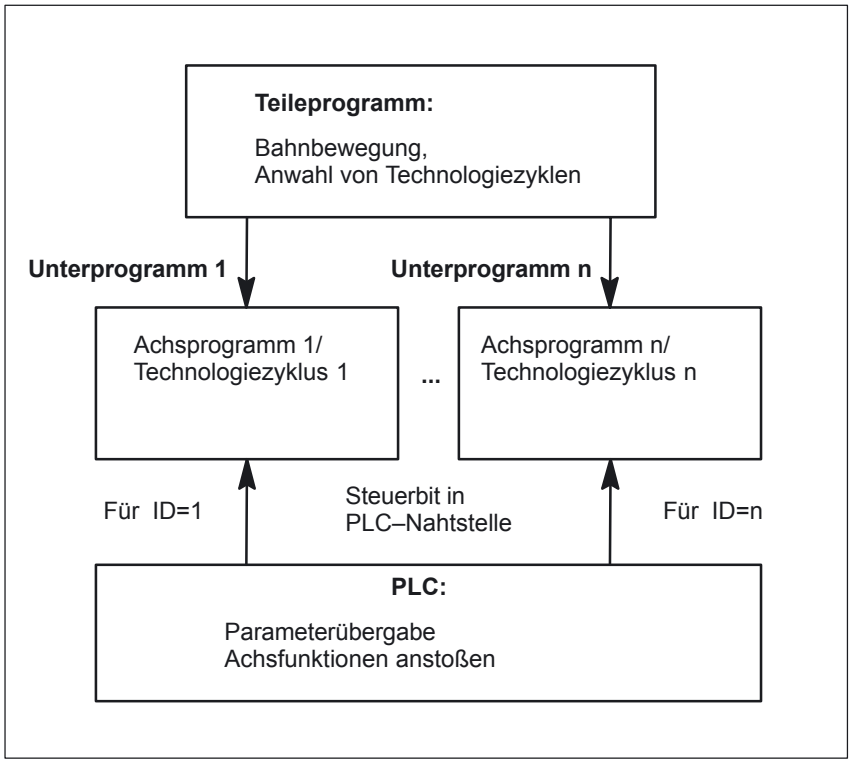


Bild 2-11 Achsprogramme/Technologiezyklen

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:
/PGA1/ Listenhandbuch Systemvariablen.

2.6.2 Geschützte Synchronaktionen

Globaler Schutz

Funktion

Über das Maschinendatum
MD11500 $MN_PREVENT_SYNACT_LOCK
kann ein Bereich von schreibgeschützten Synchronaktionen festgelegt werden. Synchronaktionen mit ID–Nummern, die im geschützten Bereich liegen, können **nicht** mehr:

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

werden, wenn sie einmal definiert sind. Geschützte Synchronaktionen können auch durch PLC nicht gesperrt werden. Sie werden der PLC an der Nahtstelle als nicht sperrbar angezeigt. Vergleiche 2.6.1. ”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 MD 11500

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

i und j können auch vertauscht angegeben werden.
Mit i = 0 und j = 0 gibt es keinen geschützten Bereich.

Kanalspezifischer Schutz

Funktion

Über das kanalspezifische Maschinendatum
MD21240 $MC_PREVENT_SYNACT_LOCK_CHAN
kann ein Bereich von schreibgeschützten Synchronaktionen für den Kanal festgelegt werden. Synchronaktionen mit ID–Nummern, die im geschützten Bereich liegen, können **nicht** mehr:

– überschrieben,
– gelöscht (CANCEL) oder
– gesperrt (LOCK)

werden, wenn sie einmal definiert sind. Geschützte Synchronaktionen können auch durch PLC nicht gesperrt werden. Sie werden der PLC an der Nahtstelle als nicht sperrbar angezeigt. Vergleiche Kapitel 2.6.1.

Anwendung

s.o.

Notation des MD 21240

```
CHANDATA(C)    ; mit C Kanalnummer
$MC_PREVENT_SYNACT_LOCK_CHAN[0]= k
                ; k Nummer der 1. für den Kanal zu sperrenden ID
$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 mit MD11500 $MN_PREVENT_SYNACT_LOCK festgelegte globale Bereich von geschützten Synchronaktionen gelten soll.

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 als:
globale Sperren oder als
kanalspezifische Sperren
angegeben wurden.

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.

```
$MN_PREVENT_SYNACT_LOCK[0] = 25          ; globale Angabe

$MN_PREVENT_SYNACT_LOCK[1] = 35          ; globale Angabe

CHANDATA(1)

$MC_PREVENT_SYNACT_LOCK_CHAN[0] = 20
; im 1. Kanal wirkt nur das kanalspez. MD(1. zu schützende ID–Nummer)

$MC_PREVENT_SYNACT_LOCK_CHAN1] = 30
; im 1. Kanal wirkt nur das kanalspez. MD(letzte zu schützende ID–Nummer)

CHANDATA(2)

$MC_PREVENT_SYNACT_LOCK_CHAN[0] = –1
                  ; im 2. Kanal wirkt das globale Maschinendatum
                  ; $MN_PREVENT_SYNACT_LOCK!

$MC_PREVENT_SYNACT_LOCK_CHAN[1] = –1
...
```

2.7 Steuerungsverhalten für Synchronaktionen in bestimmten Betriebszuständen

2.7.1 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 (P3),
/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

2.7.2 RESET

Bei Positionier–achsbewegungen

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

2.7.3 NC–STOP

Bewegungs Start aus statischen

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 mit NST ”PLC kontrolliert Achse” (DB31, ... DBX28.7). Eine solche Achse wird durch NC–STOP **nicht** mehr angehalten, sondern nur durch einen axialen STOP.

Bewegungs Start aus satzweisen und modalen

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

2.7.4 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.

2.7.5 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.

2.7.6 Verhalten der aktiven Aktionen bei Programmende und Betriebsartenwechsel

Siehe die Kapitel 2.7.4 ”Betriebsartenwechsel” und 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: $MA_SPIND_ACTIVE_AFTER_RESET== TRUE: Spindel bleibt aktiv $MA_SPIND_ACTIVE_AFTER_RESET==FALSE: Spindel stoppt Bei Betriebsartenwechsel bleibt Spindel aktiv	Spindel bleibt aktiv
Leitwertkopplung	$MC_RESET_MODE_MASK, Bit13 == 1: Leitwertkopplung bleibt aktiv $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

2.7.7 Satzsuchlauf

Allgemein

Die im Satzsuchlauf interpretierten Synchronaktionen des Programmes 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.

Polynom–koeffizienten

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

2.7.8 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.

2.7.9 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.

2.7.10 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, so wird diese Aktion im nächsten Interpolationstaktakt nicht weiter bearbeitet. Der Alarm wird also nur einmal abgesetzt. Alarme die Interpreterstopp als Alarmreaktion haben, wirken erst nach Abarbeiten der vordekodierten Sätze.

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

2.8 Projektierung

2.8.1 Projektierbarkeit

Anzahl Synchron–aktionselemente

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

MD28250 $MC_MM_NUM_SYNC_ELEMENTS

festgelegt.

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** Synchronaktionslement 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 von MD28250 $MC_MM_NUM_SYNC_ELEMENTS ist so eingestellt, dass die bis SW–Stand 3 fest vorgegebene Anzahl von max. 16 Synchronaktionen aktiviert werden kann.

Hinweis

Programmiert der Anwender keine Synchronaktionen, so kann er den Wert im MD28250 $MC_MM_NUM_SYNC_ELEMENTS auf 0 setzen, um so ca. 16 KByte an DRAM Speicher einzusparen.

Anzeige

Mit der Statusanzeige für Synchronaktionen (siehe Kap. 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 Maschinendatum

MD28252 $MC_MM_NUM_FCTDEF_ELEMENTS

projektiert.

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

Literatur: /LIS1/ Listen (Buch 1); ”MD–/SD–Listen”.

Interpolationstakt

Bei großer Anzahl von Synchronaktionen erhöht sich der Zeitbedarf für die Interpolationsebene. Der Interpolationstakt muß ggf. durch den Inbetriebnehmer 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**

2.9 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:

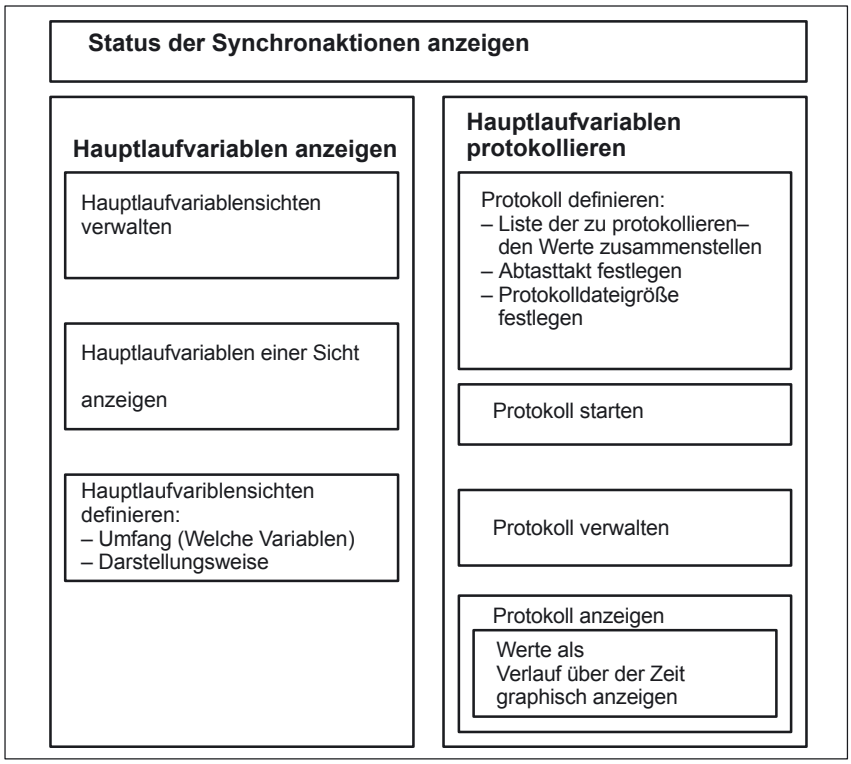


Bild 2-12 Funktionalität der Testmittel für Synchronaktionen

Die Beschreibung der Bedienung dieser Funktionen finden Sie in:

Literatur: /BAD/ Bedienungsanleitung HMI Advanced.

2.9.1 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.

2.9.2 Hauptlaufvariablen anzeigen

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.

2.9.3 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.

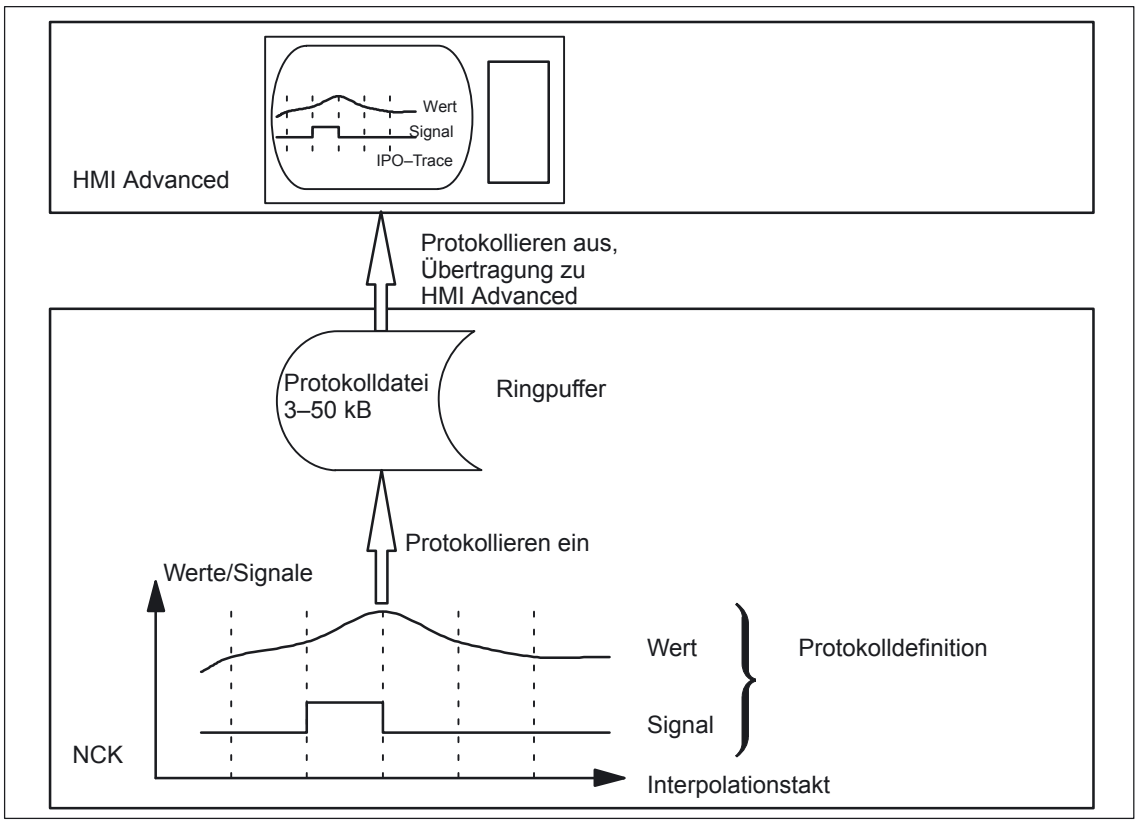


Bild 2-13 Schematischer Ablauf Hautlaufvariablen protokollieren

Bedienung

Die Hinweise zur Bedienung der Protokollierfunktion finden Sie in:

Literatur: /BAD/ Bedienungsanleitung 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 kB kann gewählt werden. Eine Protokolldefinition muß initialisiert werden, damit sie auf NCK aktiviert werden kann zum Erfassen der gewünschten Werte.

Protokolldatei-größe

Als Größe der Protokollierdatei können Werte von minimal 3 kB bis maximal 50 kB 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.

■

3 Randbedingungen

Verfügbarkeit / Leistungsumfang

Die möglichen Leistungen des Funktionspaketes 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: /BU/ Bestellunterlage, **Katalog NC 60 und NC 61** und in /LIS/ Listen (Buch 1 und Buch 2)

Darüberhinaus hängen die Funktionen der Synchronaktionen von der Liste der aus Synchronaktionen lesbaren/änderbaren Systemvariablen einschließlich Maschinen– und Settingdaten (sind auch abhängig vom SW–Stand) ab. Die für einen bestimmten SW–Stand nutzbaren Systemvariablen sind beschrieben in:

Literatur: /PGA1/ Listenandbuch Systemvariablen (gültig für den jeweiligen SW–Stand)

Erweiterungen im SW–Stand 4

Folgende Erweiterungen wurden mit SW–Stand 4 eingeführt:

- Diagnosemöglichkeiten für Synchronaktionen
- Verfügbarkeit zusätzlicher Echtzeitvariablen
- Komplexe Bedingungen in Synchronaktionen
  - Grundrechenarten
  - Funktionen
  - Indizierung mit Echtzeitvariablen
  - 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

Erweiterungen ab SW–Stand 5

Folgende Leistungen werden ab SW 5 bis 7 zusätzlich erbracht:

- Für PLC gekennzeichnete sperrbare Synchronaktionen
- Verfügbarkeit zusätzlicher Echtzeitvariablen
- 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 (ab SW 6).
- Achstausch über Synchronaktionen und im Technologiezyklus (ab SW 7).
- Anlegen von Koppelmodulen für die Generische Kopplung (ab SW 7.4).

■

4 Datenbeschreibungen (MD, SD)

4.1 Allgemeine Maschinendaten

11500 MD–Nummer	**PREVENT_SYNACT_LOCK** Geschützte Synchronaktionen		
Standardvorbesetzung: 0, 0	min. Eingabegrenze: 0	max. Eingabegrenze: 255	
Änderung gültig nach Power On	Schutzstufe: 2 / 7	Einheit: –	
Datentype: DWORD	gültig ab SW–Stand: 4.1		
Bedeutung:	Erste und letzte ID eines geschützten Synchronaktions–Bereichs. Synchronaktionen mit IDs in diesem Bereich können nicht überschrieben oder im Programm gesperrt (NC: CANCEL, LOCK) werden. Über PLC können die geschützten Synchronaktionen auch nicht gesperrt (LOCK) werden. Typische Anwendung: Der Maschinenhersteller definiert eine Sicherheitslogik in einem asynchronen Unterprogramm. Dieses wird bei Power On von PLC gestartet. Der Bereich der verwendeten IDs wird über das Maschinendatum gesperrt. Damit hat der Endkunde keine Möglichkeit, die vom Maschinerhersteller definierte Logik zu verändern oder außer Kraft zu setzen. Hinweis: Während der Erstellung der zu schützenden Synchronaktionen sollte der Schutz aufgehoben werden, da sonst bei jeder Änderung Power On notwendig ist, um die Logik neu definieren zu können. Mit 0,0 gibt es keinen Bereich von geschützten Synchronaktionen. Die Funktion ist ausgeschaltet. Die Werte werden als Absolutwerte gelesen und Ober– und Unterwert können in beliebiger Reihenfolge angegeben werden. Die Projektierung kann durch das kanalspezifische MD 21240: PREVENT_SYNACT_LOCK_CHAN ggf. umprojektiert werden.		
korrespondierend mit	MD 21240: PREVENT_SYNACT_LOCK_CHAN		

11510 MD–Nummer	**IPO_MAX_LOAD** Maximale erlaubte IPO–Last		
Standardvorbesetzung: 0	min. Eingabegrenze: 0	max. Eingabegrenze: 100	
Änderung gültig nach Power On	Schutzstufe: 2 / 7	Einheit: %	
Datentype:	gültig ab SW–Stand:		
Bedeutung:	Auslastungsauswertung über Synchronaktionen aktivieren. Über dieses MD wird eingestellt, ab welcher IPO–Rechenzeit (in % vom IPO–Takt) die Systemvariable $AN_IPO_LOAD_LIMIT auf TRUE gesetzt werden soll. Wird der Wert nach Überschreitung wieder unterschritten, so wird die Variable wieder auf FALSE gesetzt. Ist das Maschinendatum 0, so wird diese Diagnosefunktion deaktiviert.		

18860 MD–Nummer	**MM_MAINTENANCE_MON** Aktivierung der Aufzeichnung von Wartungsdaten.		
Standardvorbesetzung: 0	min. Eingabegrenze: 0	max. Eingabegrenze: 1	
Änderung gültig nach Power On	Schutzstufe: 2 / 7	Einheit: –	
Datentype: BOOLEAN	gültig ab SW–Stand: 6.5, 7.1		
Bedeutung:	Mit diesem Maschinendatum wird die Aufzeichnung von Daten zur Wartung einer Maschine aktiviert. Die Daten, die aufgezeichnet werden sollen, müssen mit dem achsspezifischen MD ... : MAINTENANCE_DATA eingestellt werden.		

4.2 Kanalspezifische Maschinendaten

21240 MD–Nummer	**PREVENT_SYNACT_LOCK_CHAN** Geschützte Synchronaktionen des Kanals		
Standardvorbesetzung: –1, –1	min. Eingabegrenze: –1	max. Eingabegrenze: 255	
Änderung gültig nach Power On	Schutzstufe: 2 / 7	Einheit: –	
Datentype: DWORD	gültig ab SW–Stand: 6.4		
Bedeutung:	Erste und letzte ID eines geschützten Synchronaktions–Bereichs. Synchronaktionen mit IDs in diesem Bereich können nicht überschrieben oder im Programm gesperrt (NC: CANCEL, LOCK) werden. Über PLC können die geschützten Synchronaktionen auch nicht gesperrt (LOCK) werden. Der Bereich der verwendeten IDs wird über das Maschinendatum gesperrt. Damit hat der Endkunde keine Möglichkeit, die vom Maschinerhersteller definierte Logik zu verändern oder außer Kraft zu setzen. Hinweis: Während der Erstellung der zu schützenden Synchronaktionen sollte der Schutz aufgehoben werden, da sonst bei jeder Änderung Power On notwendig ist, um die Logik neu definieren zu können. Mit 0,0 gibt es keinen Bereich von geschützten Synchronaktionen. Die Funktion ist ausgeschaltet. Die Werte werden als Absolutwerte gelesen und Ober– und Unterwert können in beliebiger Reihenfolge angegeben werden. Mit –1, –1 wird angegeben, daß für den Kanal die mit MD 11500: PREVENT_SYNACT_LOCK festgelegten ID–Nummern gelten sollen.		
korrespondierend mit	MD 11500: PREVENT_SYNACT_LOCK		

28250 MD–Nummer	**MM_NUM_SYNC_ELEMENTS** Anzahl Elemente für Ausdrücke der Synchronaktionen		
Standardvorbesetzung: 159	min. Eingabegrenze: 0	max. Eingabegrenze: 2000	
Änderung gültig nach Power On	Schutzstufe: 2 / 7	Einheit: –	
Datentype: DWORD	gültig ab SW–Stand: 4.1		
Bedeutung:	Die Teile der Bewegungssynchronaktionen werden für die Abspeicherung in der Steuerung in Speicherelementen abgelegt. Eine Bewegungssynchronaktion belegt minimal 4 Elemente. Es belegen: – jeder Operand in der Bedingung 1 Element – jede Aktion >= 1 Element – jede Zuweisung 2 Elemente – jeder weitere Operand in komplexen Ausdrücken 1 Element. Ein Element belegt ca 64 Bytes.		
weiterführende Literatur	Programmieranleitung Arbeitsvorbereitung		

28252 MD–Nummer	**MM_NUM_FCTDEF_ELEMENTS** Anzahl der FCTDEF–Elemente		
Standardvorbesetzung: 3	min. Eingabegrenze: 0	max. Eingabegrenze: 100	
Änderung gültig nach Power On	Schutzstufe: 2 / 7	Einheit: –	
Datentype: DWORD	gültig ab SW–Stand: 4.1		
Bedeutung:	Für die Abspeicherung von Funktionen in der Steuerung für die Verwendung in Synchronaktionen werden Speicherelemente benötigt. Das MD legt die Anzahl dieser Elemente fest.		

28254 MD–Nummer	**MM_NUM_AC_PARAM** Parameteranzahl $AC_PARAM		
Standardvorbesetzung: 50	min. Eingabegrenze: 0	max. Eingabegrenze: 10000, ab SW 6.3: 20000	
Änderung gültig nach Power On	Schutzstufe: 2 / 7	Einheit: –	
Datentype: DWORD	gültig ab SW–Stand: 4.1		
Bedeutung:	Anzahl kanalspezifischer Parameter $AC_PARAM für Bewegungssynchronaktionen		

28255 MD–Nummer	**MM_BUFFERED_AC_PARAM** Speicherort für $AC_PARAM		
Standardvorbesetzung: 0	min. Eingabegrenze: 0	max. Eingabegrenze: 1	
Änderung gültig nach Power On	Schutzstufe: 2 / 7	Einheit: –	
Datentype: DWORD	gültig ab SW–Stand: 6.3		
Bedeutung:	Die Systemvariablen $AC_PARAM können wahlweise abgespeichert werden: 0: im dynamischen Speicher DRAM, Vorbesetzung 1: im statischen Speicher SRAM Im SRAM gespeicherte Systemvariablen behalten über RESET und Power On hinweg ihre aktuellen Werte. Sie können in die Datensicherung einbezogen werden.		
korrespondierend mit	MM_NUM_AC_PARAM		
weiterführende Literatur	/IAD/, Inbetriebnahmeanleitung		

28256 MD–Nummer	**MM_NUM_AC_MARKER** Merkeranzahl $AC_MARKER		
Standardvorbesetzung: 8	min. Eingabegrenze: 0	max. Eingabegrenze: 10000, ab SW 6.3: 20000	
Änderung gültig nach POwer On	Schutzstufe: 2 / 7	Einheit: –	
Datentype: DWORD	gültig ab SW–Stand: 4.1		
Bedeutung:	Anzahl kanalspezifischer Merker $AC_MARKER für Bewegungssynchronaktionen		

28257 MD–Nummer	**MM_BUFFERED_AC_MARKER** Speicherort für $AC_MARKER		
Standardvorbesetzung: 0	min. Eingabegrenze: 0	max. Eingabegrenze: 1	
Änderung gültig nach Power On	Schutzstufe: 2 / 7	Einheit: –	
Datentype:	gültig ab SW–Stand: 6.3		
Bedeutung:	Die Systemvariablen $AC_MARKER können wahlweise abgespeichert werden: 0: im dynamischen Speicher DRAM, Vorbesetzung 1: im statischen Speicher SRAM Im SRAM gespeicherte Systemvariablen behalten über RESET und Power On hinweg ihre aktuellen Werte. Sie können in die Datensicherung einbezogen werden.		
korrespondierend mit	MM_NUM_MARKER		
weiterführende Literatur	/IAD(, Inbetriebnahmeanleitung		

28258 MD–Nummer	**MM_NUM_AC_TIMER** Anzahl Zeitvariablen $AC_TIMER		
Standardvorbesetzung: 0	min. Eingabegrenze: 0	max. Eingabegrenze: 10000	
Änderung gültig nach Power On	Schutzstufe: 2 / 7	Einheit: –	
Datentype: DWORD	gültig ab SW–Stand: 4.1		
Bedeutung:	Anzahl kanalspezifischer Zeitvariablen $AC_TIMER für Bewegungssynchronaktionen		

28260 MD–Nummer	**NUM_AC_FIFO** Anzahl Variablen $AC_FIFO1, $AC_FIFO2, ...		
Standardvorbesetzung: 0	min. Eingabegrenze: 0	max. Eingabegrenze: 10	
Änderung gültig nach Power On	Schutzstufe: 2 / /	Einheit: –	
Datentype: DWORD	gültig ab SW–Stand: 4.1		
Bedeutung:	Anzahl FIFO–Variablen $AC_FIFO1 bis $AC_FIFO10 für Bewegungssynchronaktionen.		
Anwendungsbeispiel(e)	FIFO–Variable dienen z.B. zur Produktverfolgung: In jeder FIFO–Variable kann für jedes Teil auf einem Band eine Information (z.B. die Produktlänge) zwischengespeichert werden.		
korrespondierend mit	MD 28262: START_AC_FIFO		

28262 MD–Nummer	**START_AC_FIFO** FIFO–Variablen speichern ab R–Parameter		
Standardvorbesetzung: 0	min. Eingabegrenze: 0	max. Eingabegrenze: 10000	
Änderung gültig nach Power On	Schutzstufe: 2 / 7	Einheit: –	
Datentype: DWORD	gültig ab SW–Stand: 4.1		
Bedeutung:	Nummer des R–Parameters, ab dem FIFO–Variablen gespeichert werden. Alle R–Parameter mit niedrigeren Nummern können beliebig im Teileprogramm verwendet werden. R–Parameter oberhalb des FIFO–Bereichs können aus dem Teileprogramm nicht beschrieben werden. Die Anzahl der R–Parameter muß über das Maschinendatum MD 28050: $MC_MM_NUM_R_PARAM so eingestellt werden, daß ab dem Start R–Parameter alle FIFO–Variable untergebracht werden können: $MC_MM_NUM_R_PARAM=$MC_START_FIFO + $MC_NUM_AC_FIFO*($MC_LEN_AC_FIFO+6) Die FIFO–Variable tragen die Namen $AC_FIFO1 bis $AC_FIFOn. Sie sind als Felder angelegt. Die Indizes 0 – 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 zuerst eingelesene Element n=2: Zugriff auf das zuletzt eingelesene Element n=3: Summe aller FIFO–Elemente n=4: Anzahl der im FIFO verfügbaren Elemente n=5: aktueller Schreibindex relativ zum FIFO–Anfang		
korrespondierend mit	MD 28260: NUM_AC_FIFO		

28264 MD–Nummer	**LEN_AC_FIFO** Länge der FIFO–Variablen $AC_FIFO ...		
Standardvorbesetzung: 0	min. Eingabegrenze: 0	max. Eingabegrenze: 10000	
Änderung gültig nach Power On	Schutzstufe: 2 / 7	Einheit: –	
Datentype: DWORD	gültig ab SW–Stand: 4.1		
Bedeutung:	Länge der FIFO–Variablen $AC_FIFO1 bis $AC_FIFO10. Alle FIFO–Variablen eines Kanals haben gleiche Länge.		
korrespondierend mit	MD 28262, MD 28260		

28266 MD–Nummer	**MODE_AC_FIFO** Modus der FIFO–Bearbeitung		
Standardvorbesetzung: 0	min. Eingabegrenze: 0	max. Eingabegrenze: ***	
Änderung gültig nach Power On	Schutzstufe: 2 / 7	Einheit: –	
Datentype: BYTE	gültig ab SW–Stand: 4.1		
Bedeutung:	Modus der FIFO–Bearbeitung: Bit 0 = 1: Die Summe aller FIFO–Inhalte wird bei jedem Schreibzugriff aktuell gebildet. Bit 0 = 0: Keine Summenbildung		
korrespondierend mit	MD 28260: NUM_AC_FIFO		

4.3 Achs–/Spindelspezifische Maschinendaten

18860 MD–Nummer	**MAINTENANCE_DATA** Einstellung der zur Wartung aufzuzeichnenden Daten		
Standardvorbesetzung: 1	min. Eingabegrenze: 0	max. Eingabegrenze: 3	
Änderung gültig nach RESET	Schutzstufe: 2 / 7	Einheit: –	
Datentype:	gültig ab SW–Stand:		
Bedeutung:	Mit diesem Maschinendatum wird die Aufzeichnung von Daten zur Wartung einer Maschine eingestellt. Es kann damit für jede Achse ausgewählt werden, welche Daten aufgezeichnet werden sollen. Es gibt folgende Möglichkeiten: Bit 0: Gesamtverfahrweg, Gesamtverfahrzeit und Anzahl der Verfahrvorgänge BIt 1: Gesamtverfahrweg, Gesamtverfahrzeit und Anzahl der Verfahrvorgänge bei großen Geschwindigkeiten der Achse Bit 2: Gesamtsumme des Rucks, Verfahrzeit und Anzahl der Verfahrvorgänge mit Ruck Bit 3 – 15 reserviert.		

30450 MD–Nummer	**IS_CONCURRENT_POS_AX** Konkurrierende Positionierachse		
Standardvorbesetzung: 0	min. Eingabegrenze: 0	max. Eingabegrenze: 1	
Änderung gültig nach Power On	Schutzstufe: 2 / 7	Einheit: 1	
Datentype: Boolean	gültig ab SW–Stand: 1		
Bedeutung:	Bei der Achse handelt es sich um eine konkurrierende Positionierachse. AB SW4.3 (nicht FM–NC): Wenn FALSE: Bei RESET wird eine neutrale Achse wieder Kanalachse. Wenn TRUE: Bei RESET bleibt eine neutrale Achse im Zustand neutrale Achse, und eine Kanalachse wird neutrale Achse		
weiterführende Literatur	Starten von kommandoachsen S. 2.4.13		

32070 MD–Nummer	**CORR_VELO** Achsgeschwindigkeit für Handrad, ext. NPV, cont. Dressing, Abstandsregelung		
Standardvorbesetzung: 100	min. Eingabegrenze: 0	max. Eingabegrenze: plus	
Änderung gültig nach Power On	Schutzstufe: 2 / 7	Einheit: %	
Datentype: DWORD	gültig ab SW–Stand: 3.2		
Bedeutung:	Begrenzung der Achsgeschwindigkeit für Handradüberlagerung, externe Nullpunktverschiebung, Continuous Dressing, Abstandsregelung $AA_OFF über Synchronaktionen bezogen auf die JOG–Geschwindigkeit MD: JOG_VELO, MD: JOG_VELO_RAPID, MD: JOG_REV_VELO, MD: JOG_REV_VELO_RAPID. Die maximal zulässige Geschwindigkeit ist die maximale Geschwindigkeit im MD: MAX_AX_VELO. Auf diesen Wert wird begrenzt. Eine Überschreitung dieses Werts wird durch Alarm angezeigt. Die Umrechnung nach Linear– oder Rundachsgeschwindigkeit erfolgt entsprechend MD: IS_ROT_AX.		
Anwendungsbeispiel(e)	Begrenzung der Geschwindigkeit beim Verfahren überlagerter Bewegungen.		

32074 MD–Nummer	**FRAME_OR_CORRPOS_NOTALLOWED** Wirksamkeit der Frames und Werkzeuglängenkorrektur		
Standardvorbesetzung: 0	min. Eingabegrenze: 0	max. Eingabegrenze: 0xFF	
Änderung gültig nach Power On	Schutzstufe: 2 / 7	Einheit: –	
Datentype: DWORD	gültig ab SW–Stand: 4.2		
Bedeutung:	Über dieses Maschinendatum wird die Wirksamkeit der Frames und Werkzeuglängenkorrekturen für Teilungsachsen, PLC–Achsen und aus Synchronaktionen gestartete Kommandoachsen festgelegt. Bit == 0: Frame bzw. Korrekturwerte sind erlaubt Bitbelegung: Bit 0 == 1: programmierbare Nullpunktverschiebung (TRANS) für Teilungsachse verboten. Bit 1 == 1: Maßstabsänderung (SCALE) für Teilungsachse verboten Bit 2 == 1: Richtungsumkehr (MIRROR) für Teilungsachse verboten Bit 3 == 1: DRF–Verschiebung für Achse verboten Bit 4 == 1: Externe Nullpunktverschiebung für Achse verboten Bit 5 == 1: Online–Werkzeugkorrektur für Achse verboten Bit 6 == 1: Synchronaktions–Offset für Achse verboten Bit 7 == 1: Compilezyklen–Offset für Achse verboten Bit 8 == 1: axiale Frames werden für PLC–Achsen berücksichtigt Bit 8 == 0: axiale Frames werden für PLC–Achsen NICHT berücksichtigt (Bitauswertung so aus Kompatibilitaetsgründen) Bit 9 == 1: axiale Frames werden für Kommandoachsen NICHT berücksichtigt Bit 9 == 0: axiale Frames werden für Kommandoachsen berücksichtigt		

32920 MD–Nummer	**AC_FILTER_TIME** Filter–Glättungskonstante für Adaptive–Control		
Standardvorbesetzung: 0.0	min. Eingabegrenze: 0.0	max. Eingabegrenze: plus	
Änderung gültig nach Power On	Schutzstufe: 2/7	Einheit: s	
Datentype: DOUBLE	gültig ab SW–Stand: 2.1		
Bedeutung:	Mit den Hauptlaufvariablen $AA_LOAD, $AA_POWER, $AA_TORQUE und $AA_CURR können die folgenden Antriebs–Istwerte erfasst werden: – Antriebsauslastung – Antriebswirkleistung – Antriebsmomentensollwert – Stromistwert der Achse oder Spindel Um Spitzen auszugleichen, können die gemessenen Werte durch ein PT1–Filter geglättet werden. Die Filterzeitkonstante wird mit dem MD: AC_FILTER_TIME (Filter–Glättungszeitkonstante für Adaptive–Control) definiert. Bei Erfassung des Antriebsmomentensollwerts oder Stromistwerts wirkt das Filter zusätzlich zu den im 611–D vorhandenen Filtern. Beide Filter werden hintereinander geschaltet, wenn im System sowohl stark wie auch schwach geglättete Werte benötigt werden. Durch Vorgabe der Glättungszeit 0 Sekunden wird das Filter ausgeschaltet.		
MD irrelevant bei	FM–NC mit 611A		
Anwendungsbeispiel(e)	Glättung des Stromistwertes bei AC–Regelung.		

36750 MD–Nummer	**AA_OFF_MODE** Wirkung der Wertzuweisung für axiale Überlagerung bei Synchronaktionen		
Standardvorbesetzung: 0	min. Eingabegrenze: 0	max. Eingabegrenze: 7	
Änderung gültig nach Power On	Schutzstufe: 2/7	Einheit: –	
Datentype: BYTE	gültig ab SW–Stand: 3.2 (ab SW 6 Bit 1 und 2)		
Bedeutung:	Mit der Hauptlaufvariablen $AA_OFF kann eine überlagerte Bewegung für die programmierte Achse innerhalb einer Synchronaktion realisiert werden. Über das achsiale MD: AA_OFF_MODE wird die Art der Verrechnung wie folgt definiert: Bit0: Wirkung der Werkzeugzuweisung innerhalb einer Synchronvariablen: ab SW 3.2 Bit0 = 0: absoluter Wert Bit0 = 1: inkrementeller Wert (Integrator) Bit1: Verhalten von $AA_OFF bei RESET Bit1 = 0: $AA_OFF wird bei RESET abgewählt Bit1 = 1: $AA_OFF bleibt über RESET hinaus erhalten (ab SW 6) Bit2: $AA_OFF in der Betriebsart JOG Bit2 = 0: keine Überlagerte Bewegung aufgrund von $AA_OFF Bit2 = 1: Überlagerte Bewegung wird aufgrund von $AA_OFF interpoliert (ab SW 6)		
Anwendungsbeispiel(e)	• Abstandsregelung für Laserbearbeitung (integrierend) • Joystick gesteuertes Achsverfahren (proportional)		

4.4 Settingdaten

43350 MD–Nummer	**AA_OFF_LIMIT** Obergrenze des Korrekturwertes für $AA_OFF Abstandsregelung		
Standardvorbesetzung: 1.0 Ex+8	min. Eingabegrenze: 0	max. Eingabegrenze: ***	
Änderung gültig nach SOFORT	Schutzstufe: 2 / 7	Einheit: mm/ Grad	
Datentype: DOUBLE	gültig ab SW–Stand: 4.2		
Bedeutung:	Obergrenze des Korrekturwertes, der über Synchronaktionen über die Variable $AA_OFF vorgegeben werden kann. Der Grenzwert wirkt auf den absolut wirksamen Korrekturbetrag. Anwendung für die Abstandsregelung bei Laserbearbeitung: Der Korrekturwert wird begrenzt, damit sich der Laser–Kopf nicht in Blechausschnitten verhaken kann. Über die Systemvariable $AA_OFF_LIMIT kann abgefragt werden, ob sich der Korrekturwert im Grenzbereich befindet.		

■

Platz für Notizen

5 Signalbeschreibungen

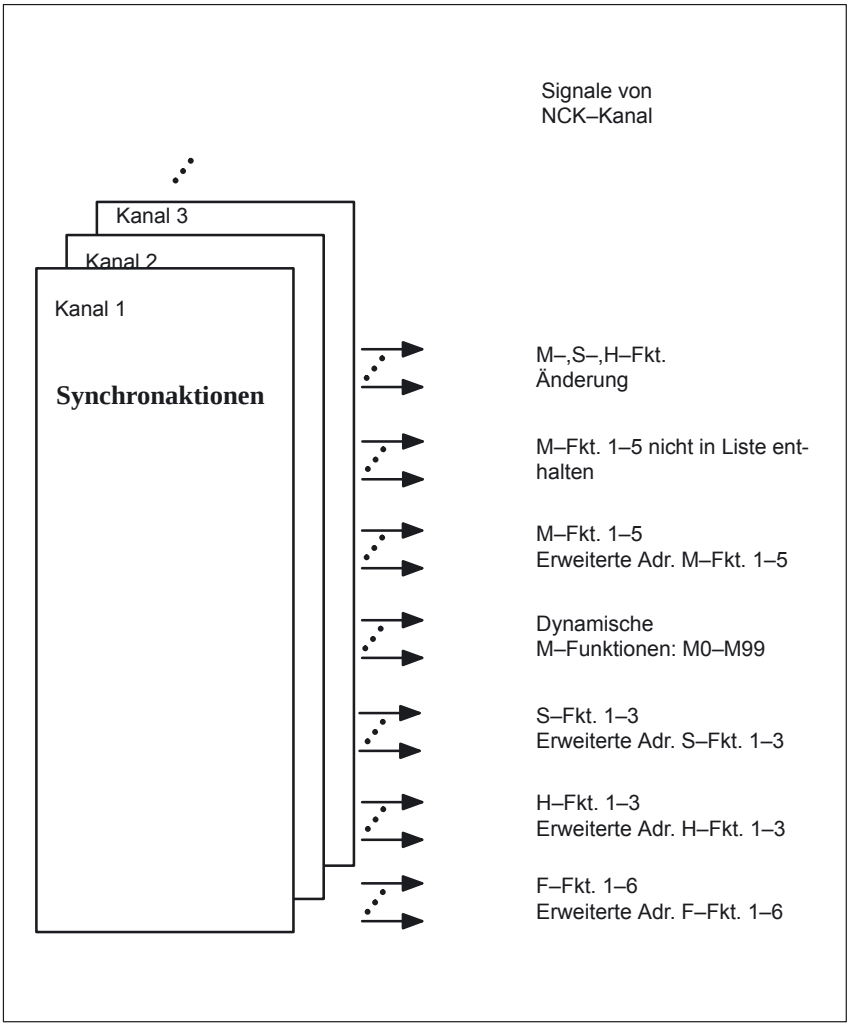


Bild 5-1 PLC–Nahtstellensignale für Synchronaktionen

Die durch Hilfsfunktionenausgabe aus Synchronaktionen erzeugten Signale entsprechen denen in

Literatur:
/FB1/ Funktionshandbuch Grundfunktionen; Hilfsfunktionsausgabe an PLC (H2) beschriebenen Signalen.

Signale an Kanal

Mit den Signalen DB 21 – 30 DBB 300 Bit 0 bis
DB 21 – 30 DBB 307 Bit 7
fordert das PLC–Anwendungsprogramm die Sperrung der zugeordneten Synchronaktionen an. Dabei entspricht:
DBB 300 Bit 0 der ersten modalen Synchronaktion (ID=1/IDS=1) und
DBB 307 Bit 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.

Signale von Kanal

Mit den Signalen DB 21 – 30 DBB 308 Bit 0 bis
DB 21 – 30 DBB 315 Bit 7
zeigt der Kanal dem PLC–Anwendungsprogramm an, welche Synchronaktionen durch PLC gesperrt werden dürfen. Dabei entspricht:
DBB 308 Bit 0 der ersten modalen Synchronaktion (ID=1/IDS=1) und
DBB 315 Bit 7 der 64. modalen Synchronaktion (ID=64/IDS=64).

Alle Synchronaktionen sperren

Das globale Signal

DB21–30 DBB1 Bit 2

sperrt **alle** modalen/statischen Synchronaktionen, soweit sie nicht geschützt sind.

Markierte Synchronaktionen sperren

DB 21 – 30 DBX280.1
Von den in DB 21 – 30 DBB 308 Bit 0 bis DB 21 – 30 DBB 315 Bit 7 als sperrbar markierten Synchronaktionen sollen die in DB 21 – 30 DBB 300 Bit 0 bis DB 21 – 30 DBB 307 Bit 7 durch gesetzte Bit als zu sperren gekennzeichneten Synchronaktionen gesperrt werden.

Synchronaktionen gesperrt

DB 21 – 30 DBX281.1
Die angeforderten Synchronaktionen wurden von NCK als gesperrt bestätigt.

■

Beispiele

6

6.1 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 ...

Schrittweise Positionieren

Jedesmal, 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 gesetz werden.
WHENEVER $A_IN[1]==0 DO $AC_OVR= 0

Weitere Möglichkeiten

Die Liste der in Synchronaktionen lesbaren Systemvariablen in

Literatur: /PGA1/ Listenhandbuch Systemvariablen

erschließt die volle Menge der in Bedingungen von Synchronaktionen auswertbaren Größen.

6.2 Schreiben und Lesen von SD/MD aus Synchronaktionen

Zustellung und Pendeln beim Schleifen

Settingdaten, deren Wert 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!)

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
                                ;          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
```

6.3 Beispiele zur AC–Regelung

Allgemeines Vorgehen

Die folgenden Beispiele benutzen die Polynomauswertefunktion SYNFCT().

1. Darstellung des Zusammenhangs zwischen Eingangswert und Ausgangswert (jeweils Echtzeitvariablen)
2. Definition dieses Zusammenhanges als Polynom mit Begrenzungen
3. bei Positionsoffset: Setzen der MD und SD
   – MD 36750: $AA_OFF_MODE
   – SA 43350: $SA_AA_OFF_LIMIT (optional)
4. Aktivierung der Regelung in einer Synchronaktion

6.3.1 Abstandsregelung mit variabler Obergrenze

Beispiel für Polynom mit dyn. 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:

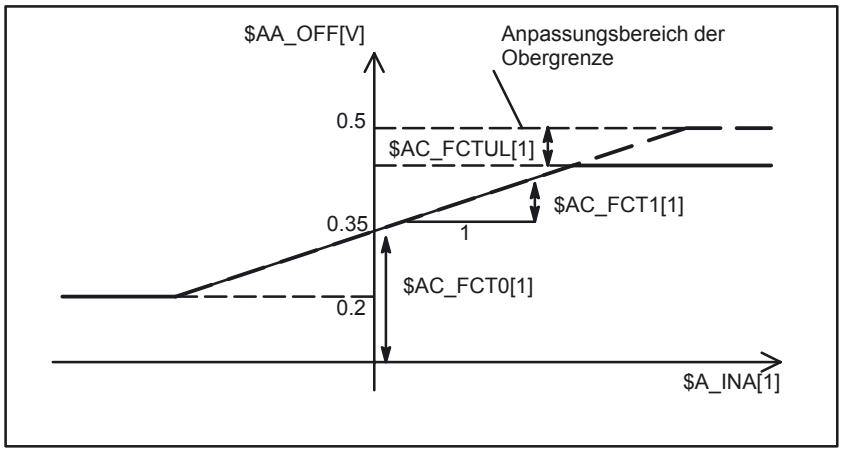


Bild 6-1 Abstandsregelung mit variabler Obergrenze

```
$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
...
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 Programmierungvon STOPRE für satzsynchrones Schreiben gesorgt werden. Gleichwertig zur obigen Notation zur Polynomdefinition ist:
FCTDEF(1,0.2, 0.5, 0.35, 1.5EX–5).

6.3.2 Regelung des Vorschubs

Beispiel für AC–Regelung mit einer analogen Eingangs–Spannung

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.

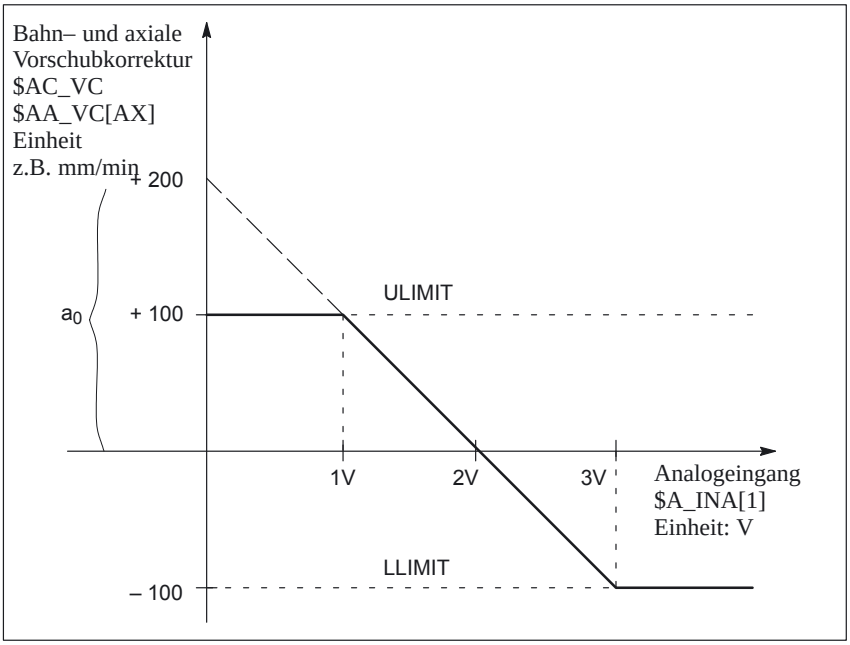


Bild 6-2 Diagramm für AC–Regelung

Bestimmung der Koeffizienten:

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

$a_1 = -\frac{100\ mm}{1\ min \cdot 1\ V}$

$a_1 = -100 \Rightarrow$ Regelkonstante, Steigung

$a_0 = -(-100) \cdot 2 = 200$

$a_2 = 0$ (kein quadratisches Glied)

$a_3 = 0$ (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])

6.3.3 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.

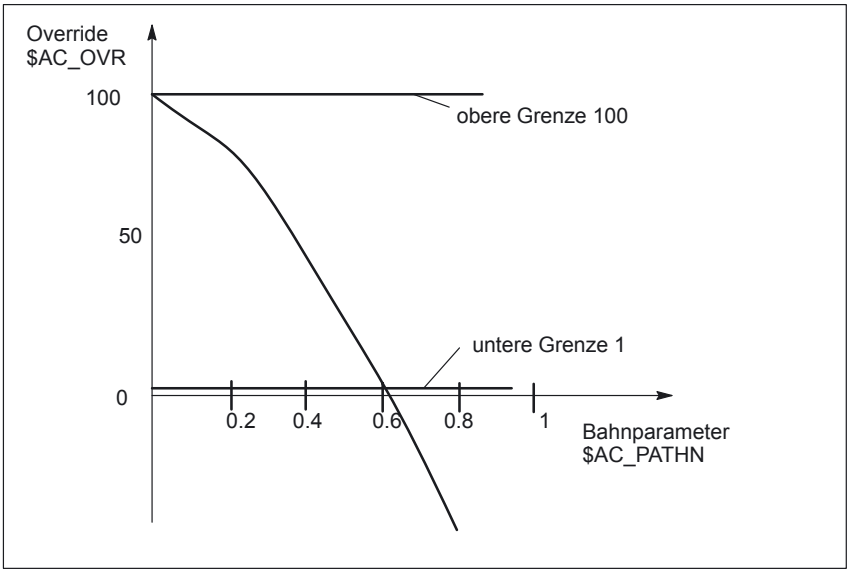


Bild 6-3 Geschwindigkeit kontinuierlich regeln

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
```

6.4 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

6.5 Ausführungszeiten in R–Parameter ablegen

Aufgabe

Lege ab R–Parameter 10 die Ausführungszeit für die Teileprogrammsätze ab.

Programm **Kommentar**

```
                                        ; Ohne symbolische Programmierung sieht das
                                        ; Beispiel so aus:
IDS=1 EVERY $AC_TIMEC==0 DO $AC_MARKER[0] = $AC_MARKER[0] + 1
                                        ; bei Satzwechsel R–Parameter–Zeiger weiterstellen

IDS=2 DO $R[10+$AC_MARKER[0]] = $AC_TIME
                                        ; Schreibe jeweils die aktuelle Zeit vom Satzanfang in
                                        ; R–Parameter

                                        ; Mit symbolischer Programmierung sieht das
                                        ; Beispiel so aus:
DEFINE INDEX AS $AC_MARKER[0]           ; Vereinbarungen für symbolische Programmierung
IDS=1 EVERY $AC_TIMEC==0 DO INDEX = INDEX + 1   ; bei Satzwechsel R–Parameter–Zeiger weiterstellen
IDS=2 DO $R[10+INDEX] = $AC_TIME
                                        ; Schreibe jeweils die aktuelle Zeit vom Satzanfang in
                                        ; R–Parameter
```

6.6 ”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.

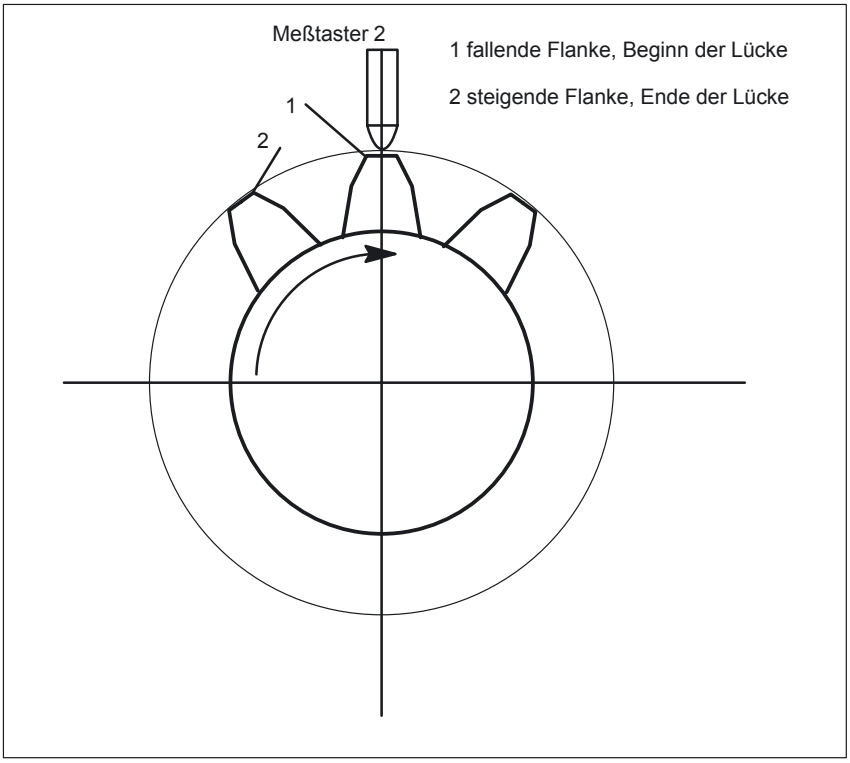


Bild 6-4 Schemabild zum Messen der Zahnradlücken

```
%_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                                 ; Lückenpositon 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  FA[BACH]=1000          ; einschalten
ID=4 EVERY $R6==0 und ($AA_STAT[BACH]==1) DO MOV[BACH]=0  ; ausschalten

; Messen nacheinader, 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 MEAC[BACH]=(0)   ; Messung abbrechen
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 Zaehnen 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 Meßpostion auf Grad umgerechnet
N530 R3=R3 MOD 360 ;ersten Meßpunkt modulo
N540 R5=(R3–R14)+(R4/2) ; Lückenpositon rechnen
M00
stopre
R6=0                                ; Achsdrehung von BACH ausschalten
gotob anfang
M30
```

6.7 Achskopplungen über Synchronaktionen

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

N265 WAITP(BACH)

N270 ID=3 EVERY $R2==1 DO MOV[BACH]=1  FA[BACH]=R5   ; Leitachse mit Vorschub in R5 endlos drehen
N275 ID=4 EVERY $R2==0 DO MOV[BACH]=0                ; Leitachse anhalten

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
```

6.7.2 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 Werkstükkes 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 wirk:
– 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.

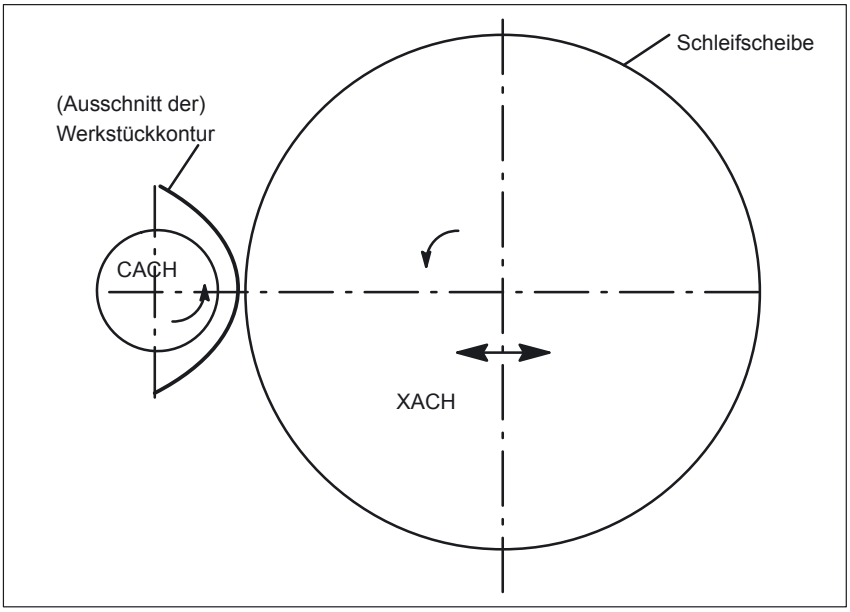


Bild 6-5 Schema Schleifen einer Unrund–Kontur

```
%_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 Linieare 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
; Kopplelverbund 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 anwaehlen,  (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                     ; Unterprogamm 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 LEADON(CASW,CACH,1)      ; CTAB Kopplung  EIN Leitachse CACH
N2400 ID=6 EVERY $R3==0 DO LEADOF(CASW,CACH)        ; CTAB Kopplung AUS Leitachse CACH


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

N2900 WAITP(CACH)

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

N3200 STOPRE
N3300 R90=$AA_COUP_ACT[CASW]                        ; Zustand der Kopplung für CASW zum Prüfen
N3400 MSG(”Overridetabelle CASW eingeschalten mit LEADON ”<<R90<<”, weiter ENDE mit NC–START”)
```

```
N3500 M00                ; ********** NC HALT **************
N3600 MSG()
N3700 STOPRE             ; 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 folgende 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. 2.4.7.

6.7.3 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 wird. 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
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
```

6.8 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. Kapitel 2.4.13 ”Starten von Kommandoachsen,
Kapitel 2.6.1. ”Beeinflussung von PLC”.

Koordinierung

Die Koordinierung zwischen PLC und NCK erfolgt über die ab SW–Stand 4 verfügbaren gemeinsamen Daten
(siehe: /LIS2/ Listen Buch 2 Nahtstelle solution, Nahtstelle line powerline).

- $A_DBB[0] 1 Grundposition einnehmen
- $A_DBB[1] 1 Zielposition einnehmen
- $A_DBW[1] zu positionierender Wert + / – , PLC berechnet den kürzesten Weg.

Synchronaktionen %_N_MAIN_MPF

```
...
IDS=1 EVERY $A_DBB[0]==1 DO  NULL_POS        ; wenn $A_DBB[0] von PLC gesetzt, Grundposition einnehmen
IDS=2 EVERY $A_DBB[1]==1 DO  ZIEL_POS        ; wenn $A_DBB[1] von PLC gesetzt, Spindel auf den in
                                             ; $A_DBW[1] hinterlegten Wert positionieren
...
```

Technologiezyklus NULL_POS %_N_NULL_POS_SPF

```
PROC NULL_POS
SPOS=0                                       ; Antrieb für den Werkzeugwechsel in Grundposition bringen
$A_DBB[0]=0                                  ; Grundposition in NCK ausgeführt
```

Technologiezyklus ZIEL_POS %_N_ZIEL_POS_SPF

```
PROC ZIEL_POS
SPOS=IC($A_DBW[1])                           ; Spindel auf den Wert positionieren, der in $A_DBW[1]
                                             ; von PLC hinterlegt wurde, Kettenmaß
$A_DBB[1]=0                                  ; Zielpositionieren in NCK ausgeführt
```

6.9 Synchronaktionen im Bereich WZW/BAZ

Einführung

Das folgende Bild zeigt den schematischen Ablauf Werkzeugwechselzyklus.

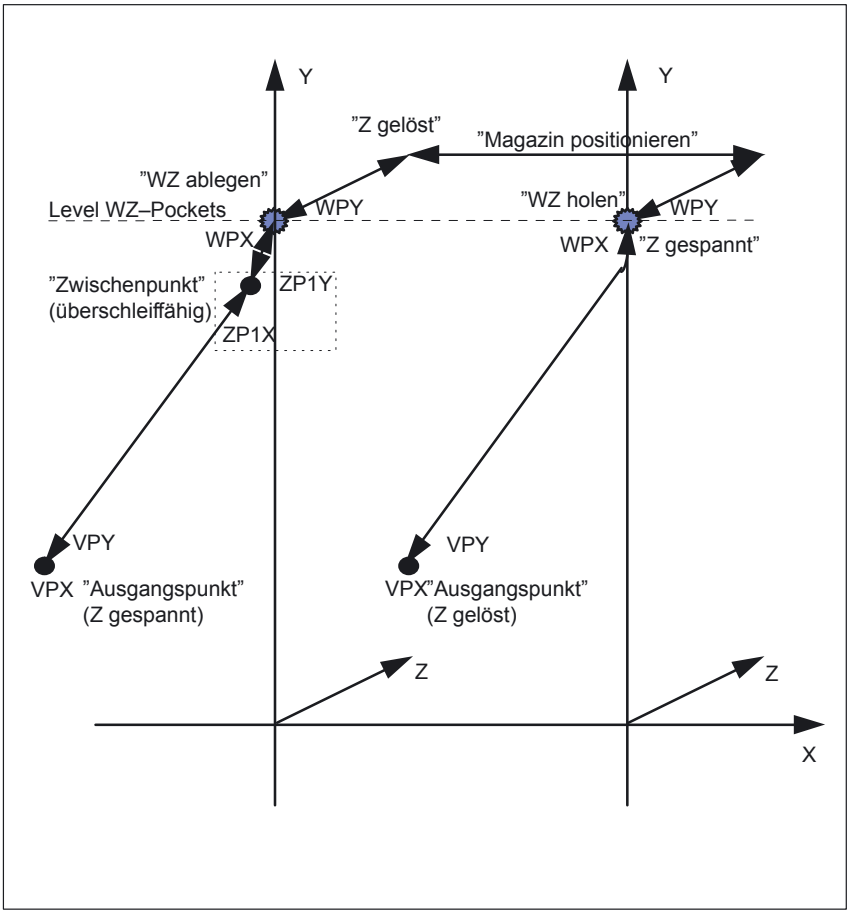


Bild 6-6 Schematischer Ablauf Werkzeugwechselzyklus

Ablaufdiagramm

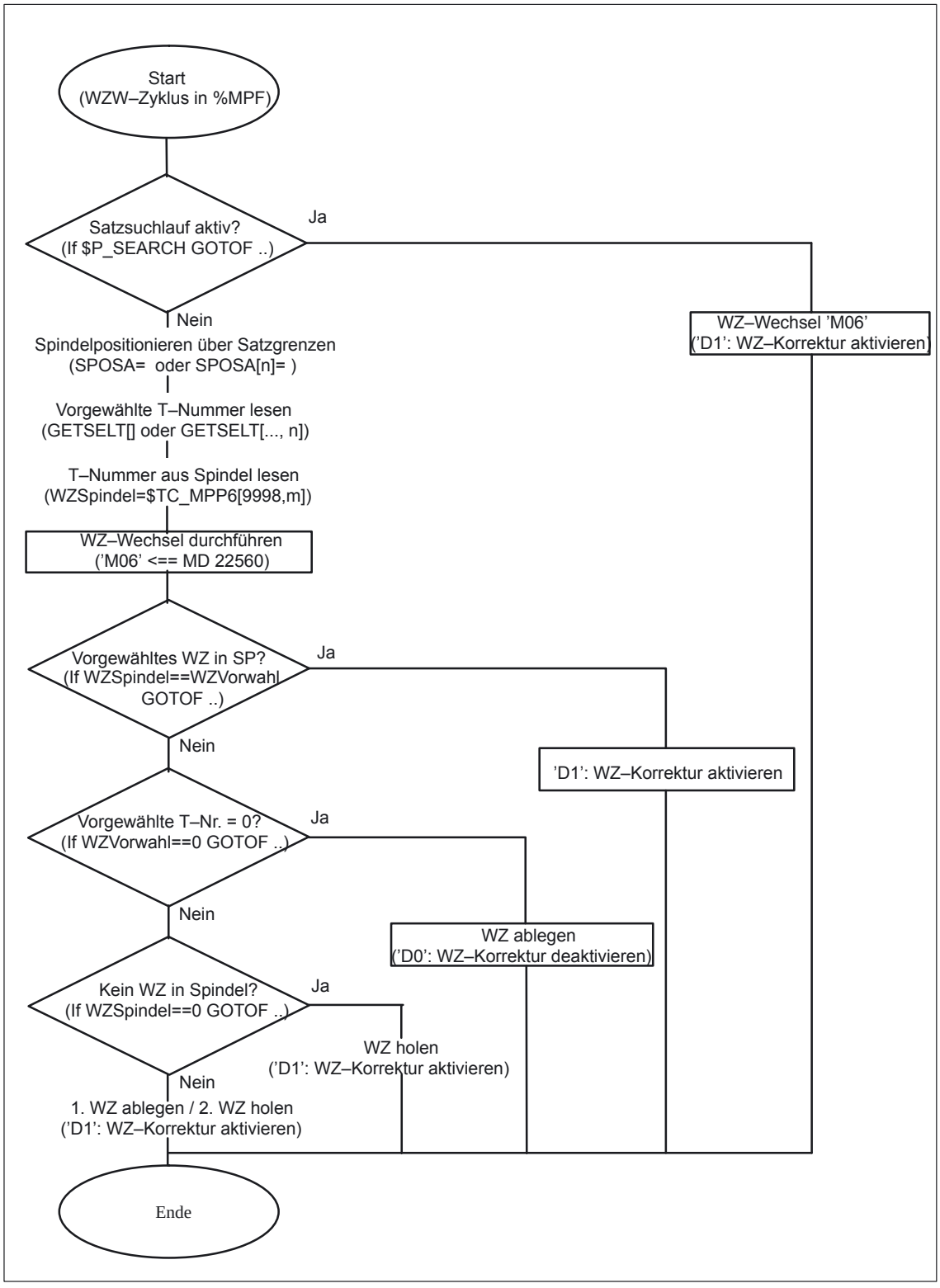


Bild 6-7 Ablaufdiagramm Werkzeugwechselzyklus

NC–Programm **Kommentar**

%_N_WZW_SPF

```
;$PATH=/_N_SPF_DIR
N10 DEF INT WZVorwahl,WZSpindel
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          ; Marker auf = 1 wenn MagAchse gefahren
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 GO-
TOF 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

;*****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
```

■

Platz für Notizen

7 Datenfelder, Listen

7.1 Systemvariable

Übersicht

In den folgenden Kapiteln sind diejenigen Systemvariablen aufgelistet, die aus Synchronaktionen heraus gelesen und/oder geschrieben werden können.

Legende:

r	Lesen
w	Schreiben
R	Lesen mit implizitem Vorlaufstop
W	Schreiben mit implizitem Vorlaufstop
TP	Teileprogramm
SA	Synchronaktion
SW	SW–Stand s. Hinweis

Hinweis

Regeln zur Bennenung von Systemvariablen für Synchronaktionen:

- ”ACT” (z.B. $AA_V**ACT**M) kennzeichnet **Soll**werte, die im Interpolator berechnet werden und als Eingangsgrößen der Achsregelung verwendet werden.
- ”$VA_...” kennzeichnet echte Istwerte, die über die Auswertung der Encoder–Information tatsächliche Werte einer Maschinenachse wiedergeben.

Literatur: /PGA1/ Listenandbuch Systemvariablen
(gültig für den jeweiligen SW–Stand)

7.2 Nahtstellensignale

DB–Nummer	Bit , Byte	Name	Verweis
kanalspezifisch			
21–30	280.1	Modale Synchronaktionen gem. DBX 300.0–307.7 sperren	
21–30	300.0 –	Modale Synchronaktionen gem. DBX 300.0–307.7 gesperrt, Quittung von NCK	
21–30	300.0 –	Modale Synchronaktionen ID oder IDS 1 –	
21–30	307.7	64 sperren. Anforderung an Kanal der NCK	
21–30	308.0 –	Modale Synchronaktionen ID oder IDS 1 –	
21–30	315.7	64 sperrbar. Mitteilung von NCK.	

7.3 Maschinendaten

Nummer	Bezeichner	Name	Verweis
allgemein ($MN_ ...)			
11110	AUXFU_GROUP_SPEC	Hilfsfunktionsgruppenspezifikation	H2
11500	PREVENT_SYNACT_LOCK	Geschützte Synchronaktionen	
11510	IPO_MAX_LOAD	Maximale erlaubte IPO_Last	
18860	MM_MAINTENANCE_MON	Aktivierung der Aufzeichnung von Wartungsdaten	
kanalspezifisch ($MC_ ...)			
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 (ab SW 6.3)	
28256	MM_NUM_AC_MARKER	Merkeranzahl $AC_MARKER	
28257	MM_BUFFERED_AC_MARKER	Speicherort für $AC_MARKER (ab SW 6.3)	
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	

achsspezifisch ($MA_ ...)			
30450	IS_CONCURRENT_POS_AX	Konkurrierende Positionierachse	P2
32060	POS_AX_VELO	Löschstellung für Positionierachsgeschwindigkeit	P2
32070	CORR_VELO	Achsgeschwindigkeit für Handrad, ext. NPV, cont. Dressing, Abstandsregelung (ab SW3)	H1
32074	FRAME_OR_CORRPOS_NOTALLOWED	Wirksamkeit der Frames und Werkzeuglängenkorrektur	
32920	AC_FILTER_TIME	Filter–Glättungszeitkonstante für Adaptive Control (ab SW2)	
33060	MAINTENANCE_DATA	Konfiguration der Aufzeichnung von Wartungdaten	
36750	AA_OFF_MODE	Wirkung der Wertzuweisung für axiale Überlagerung bei Synchronaktionen (ab SW3)	
37200	COUPLE_POS_TOL_COARSE	Schwellwert für ”Synchronlauf grob”	S3
37210	COUPLE_POS_TOL_FINE	Schwellwert für ”Synchronlauf fein”	S3
Settingdaten ($SA_ ...)			
43300	ASSIGN_FEED_PER_REV_SOURCE	Umdrehungsvorschub für Positionierachsen/Spindeln	V1
43350	AA_OFF_LIMIT	Obergrenze des Lorrekturwertes für $AA_OFF Abstandsregelung	
43400	WORKAREA_PLUS_ENABLE	Arbeitsfeldbegrenzung in pos. Richtung	A3

7.4 Alarme

Ausführliche Erläuterungen zu den auftretenden Alarmen können der
Literatur: /DA/ Diagnosehandbuch
bzw. bei Systemen mit HMI Embedded bzw. HMI Advanced der Online–Hilfe entnommen werden.

■

Platz für Notizen

Zeichen

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

Platz für Notizen

An
SIEMENS AG
A&D MC MS
Postfach 3180

D–91050 Erlangen

Tel. +49 (0) 180 5050–222 [Hotline]
Fax +49 (0) 09131 98–63315 [Dokumentation]
mailto:motioncontrol.docu@siemens.com

Vorschläge

Korrekturen

für Druckschrift:

SINUMERIK 840D sl
SINUMERIK 840D/840Di/810D
Funktionsbeschreibung
Synchronaktionen

Hersteller–/Service–Dokumentation

Absender

Name

Anschrift Ihrer Firma/Dienststelle

Straße

PLZ: Ort:

Telefon: /

Telefax: /

Funktionsbeschreibung

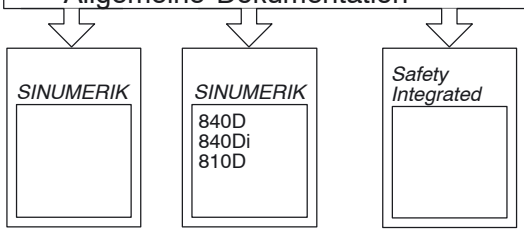
Bestell–Nr.: 6FC5397-5BP10-1AA0
Ausgabe: 03/2006

Sollten Sie beim Lesen dieser Unterlage auf Druckfehler gestoßen sein, bitten wir Sie, uns diese mit diesem Vordruck mitzuteilen. Ebenso dankbar sind wir für Anregungen und Verbesserungen.

Vorschläge und/oder Korrekturen

Dokumentationsübersicht SINUMERIK 840D/840Di/810D (03/2006)

Allgemeine Dokumentation

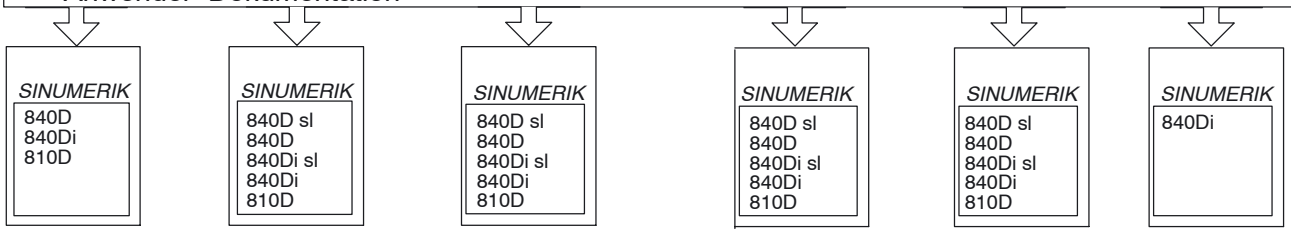


Werbeschrift

Katalog NC 60 ***)**

Safety Integrated
Applikationshandbuch

Anwender–Dokumentation



Bedienhandbuch
– **HMI Embedded *)**
– **ShopMill**
– **ShopTurn**
– HT6

Bedienhandbuch
– **HMI Advanced *)**
– Bedienung Kompakt

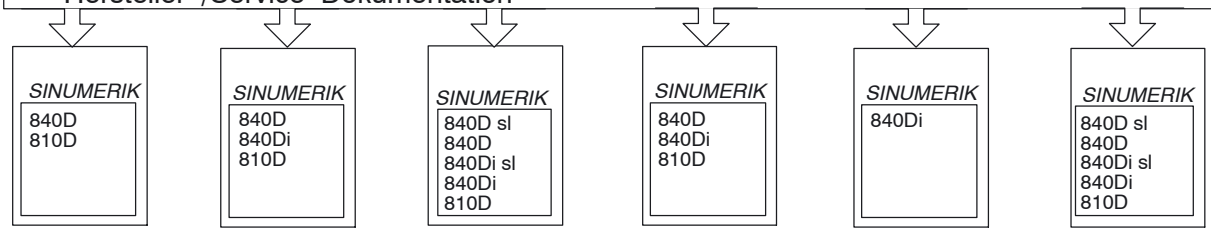
Programmierhandbuch
– Grundlagen ***)**
– Arbeitsvorbereitung ***)**
– Programmierung Kompakt
– Listen Systemvariablen
– ISO Turning/Milling

Programmierhandbuch
– **Zyklen**
– **Messzyklen**

Diagnose–handbuch *)

Systemüberblick

Hersteller–/Service–Dokumentation



Projektierung (HW) ***)**
– 840D
– 810D

Gerätehandbuch
Bedien–komponenten *)

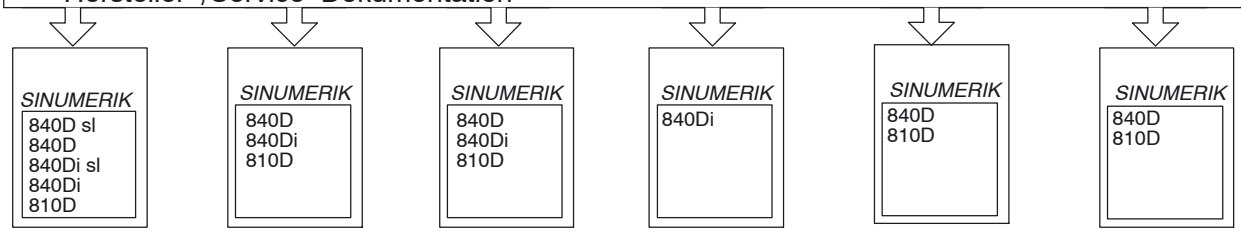
Inbetriebnahme–handbuch *)
– 840D
– 810D
– HMI

Funktionshanbuch
– **ShopMill**
– **ShopTurn**

Inbetriebnahme–handbuch

Listen–handbuch *)
– Teil 1
– Teil 2

Hersteller–/Service–Dokumentation



Funktionshandbuch
– Grundlagen ***)**
– Erweiterungsfkt.
– Sonderfunktionen
– **Synchronaktionen**
– **ISO–Dialekte**
– **EMV–Richtlinien**

Funktionshandbuch
– **Antriebsfkt.**
– **Werkzeugverw.**
– **Hydraulikmodul**
– **Analogmodul**

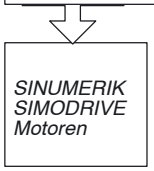
Funktionshandbuch
– **Ferndiagnose**
– **@Event**

Funktionshandbuch
Safety Integrated

Projektierung
– **Bedienoberfläche OP030**
– **HMI Embedded**

MCIS
– Rechnerkopplung
– Werkzeugbedarf
– NC–Datenverw.
– NC–Datenübertr.
– Tool Data Commun.

Elektronische Dokumentation

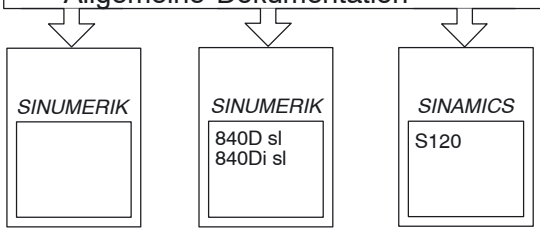


DOCONCD ***)**
DOCONWEB

***)** Empfohlener Minimalumfang der Dokumentation

Dokumentationsübersicht SINUMERIK 840D sl/840Di sl (03/2006)

Allgemeine Dokumentation

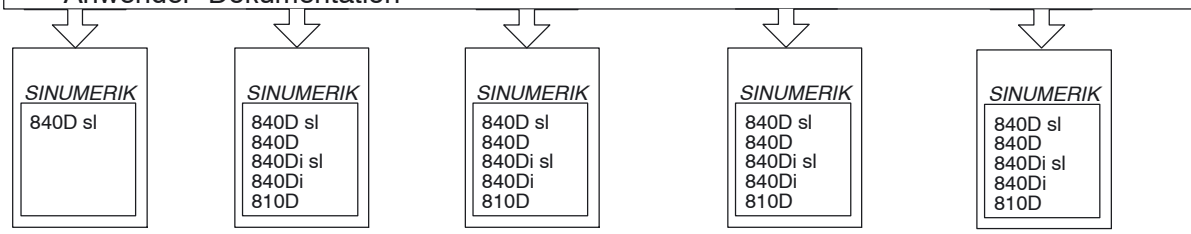


Werbeschrift

Katalog NC 61 ***)**

Katalog
D21.2 Servo Control ***)**

Anwender–Dokumentation



Bedienhandbuch
– **HMI Embedded *)**
– **ShopMill**
– **ShopTurn**

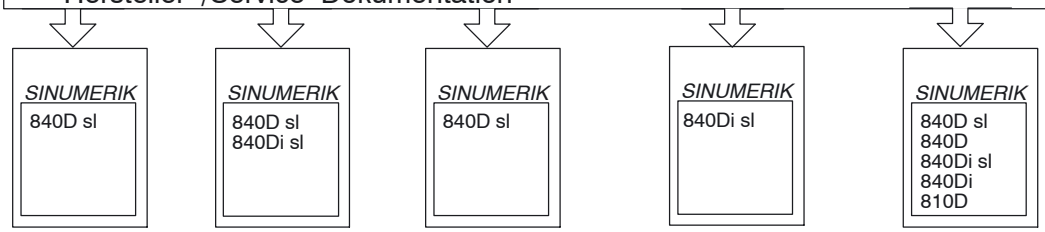
Bedienhandbuch
– **HMI Advanced *)**
– Bedienung Kompakt

Programmierhandbuch
– Grundlagen ***)**
– Arbeitsvorbereitung ***)**
– Programmierung Kompakt
– Listen Systemvariablen
– ISO Turning/Milling

Programmierhandbuch
– **Zyklen**
– **Messzyklen**

Diagnosehandbuch *)

Hersteller–/Service–Dokumentation



Gerätehandbuch
NCU *)

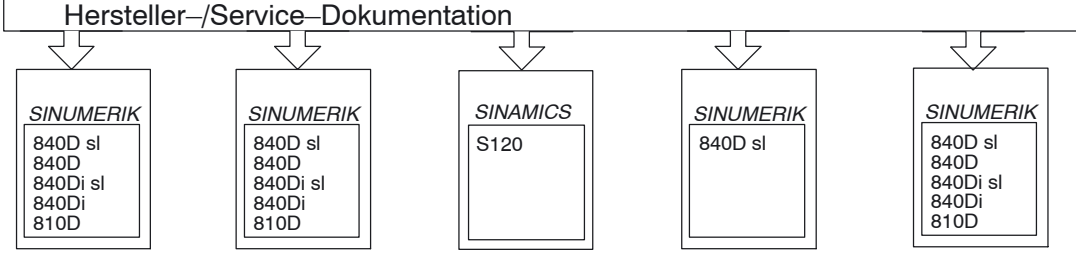
Gerätehandbuch
Bedien–komponenten *)

Inbetriebnahme–handbuch CNC ***)**
– Teil 1 NCK, PLC, Antrieb
– Teil 2 HMI
– Teil 3 ShopMill
– Teil 4 ShopTurn
– Teil 5 Basesoftware

Inbetriebnahme-handbuch

Listen–handbuch *)
– Teil 1
– Teil 2

Hersteller–/Service–Dokumentation



Funktionshandbuch
– Grundlagen ***)**
– Erweiterungsfkt.
– Sonderfunktionen

Funktionshandbuch
– **Synchronaktionen**
– **ISO–Dialekte**

Funktionshandbuch
Antriebsfunktionen

Funktionshandbuch
Safety Integrated

EMV–Richtlinien

Elektronische Dokumentation

DOCONCD ***)**
DOCONWEB

***)** Empfohlener Minimalumfang der Dokumentation