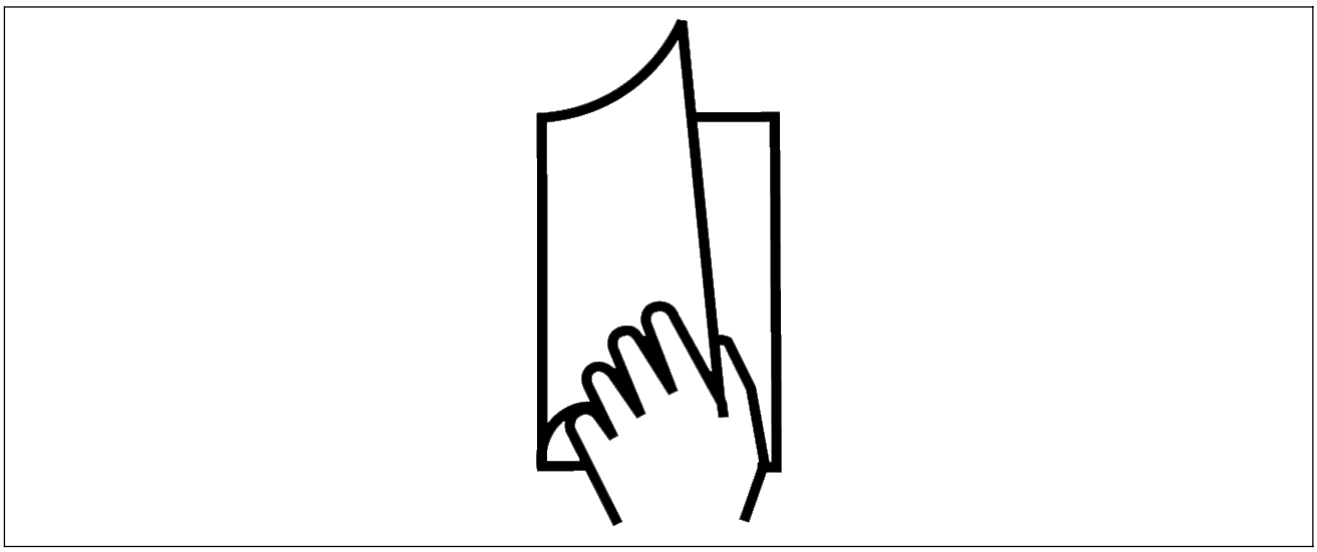
Inhaltsverzeichnis

Projektierungshandbuch lesen



□ Projektierungshandbuch lesen

Dieses Projektierungshandbuch führt Sie Schritt für Schritt durch die Anwendung der Motion Control Option MCO 305. Bitte lesen Sie auch das Produkthandbuch, um sicher und professionell mit dem System zu arbeiten und beachten Sie vor allem auch die Sicherheitshinweise und allgemeinen Warnungen.

Das Kapitel **Projektierungshandbuch lesen** führt in das Projektierungshandbuch ein und informiert über die Symbole, Abkürzungen und Definitionen, die in diesem Handbuch benutzt werden.

Seitenteiler für „Projektierungshandbuch lesen".

Das Kapitel **Einführung zu MCO 305** informiert über Funktionsweise und Eigenschaften der MCO 305, gibt einen Systemüberblick anhand von Konfigurationsbeispielen und erklärt einige grundlegende Themen wie Drehgeber und Programmausführung.

Seitenteiler für das Kapitel „Einführung".

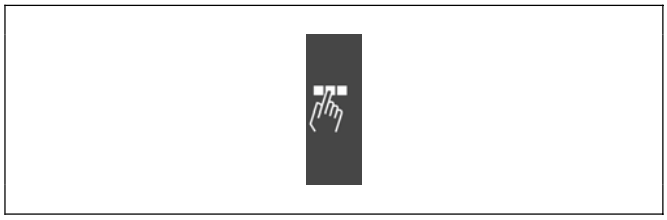
Das Kapitel **Funktionen und Beispiele** führt Sie durch Anwendungsbeispiele von der einfachen Positionierung über verschiedene Synchronisationen bis hin zu Kurvenscheibensteuerungen. Mit diesen Beispielen können Sie im Detail nachvollziehen wie die Parameter gesetzt, die Steuerungen programmiert und die Kurven editiert werden.

Seitenteiler für „Funktionen und Beispiele".

__ Projektierungshandbuch lesen __

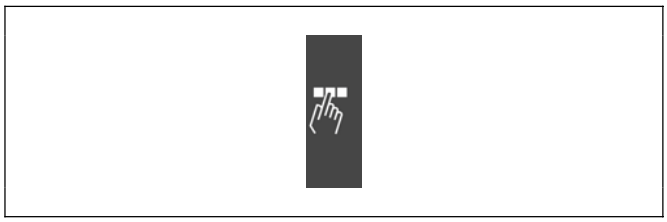
Das Kapitel **PC Software Benutzeroberfläche** informiert über die APOSS-spezifischen Menüs und Funktionen, vor allem über den CAM-Editor zum Erzeugen der Kurvenprofile.

Für mehr Details klicken Sie bitte auf → *Hilfe* in der APOSS Menüleiste.



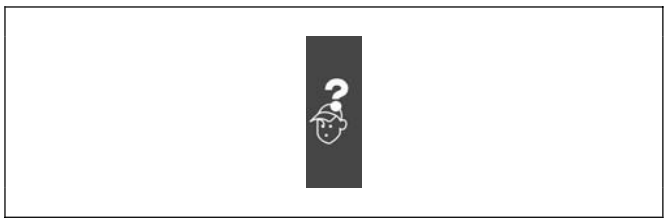
Seitenteiler für „PC Software Benutzeroberfläche".

Das Kapitel **Programmieren** zeigt wie man Steuerungen für den Frequenzumrichter mit MCO 305 programmiert. Dieses Kapitel bietet eine detaillierte Beschreibung aller Befehle und aller Parameter in den beiden Abschnitten Software- und Parameter-Referenz.



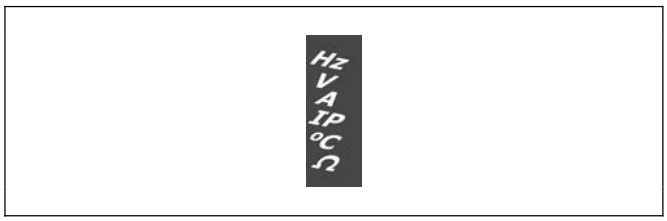
Seitenteiler für „Programmieren".

Das Kapitel **Fehlersuche und -behebung** hilft, die Ursachen von Problemen, die beim Arbeiten mit dem Frequenzumrichter mit MCO 305 auftreten können, zu finden und zu beheben. Der nächste Abschnitt erklärt die wichtigsten Meldungen von der PC-Bedienoberfläche.



Seitenteiler für „Fehlersuche und -behebung".

Das Kapitel **Anhang** zeigt im Überblick, was sich bei den MCO-Parametern im Vergleich zu den SYNCPOS-Parametern verändert hat. Erfahrene Anwender finden ausführliche Informationen in der Technischen Referenz zum Beispiel „Array Structure of CAM Profiles". Das Handbuch schließt mit einem Stichwortverzeichnis.



Seitenteiler für „Anhang".

In der Online-Hilfe finden Sie im Kapitel **Programmbeispiele** etwa 50 kurze Beispiele, die Sie benutzen können, um sich mit dem Programm vertraut zu machen oder direkt in Ihr Programm kopieren können.

□ Verfügbare Literatur für FC 300, MCO 305 und MCT 10 Motion Control Tool

- Das MCO 305 Produkthandbuch liefert die erforderlichen Informationen zum Einbau und für die Inbetriebnahme des MCO 305 sowie für die Optimierung der Steuerung.
- Das VLT® AutomationDrive FC 300 Produkthandbuch liefert die erforderlichen Informationen für die Inbetriebnahme und den Betrieb des Frequenzumrichters.
- Das VLT® AutomationDrive FC 300 Projektierungshandbuch enthält alle technischen Informationen zum Frequenzumrichter sowie Informationen zur kundenspezifischen Anpassung und Anwendung.
- Das VLT® AutomationDrive FC 300 MCT 10 Produkthandbuch bietet Informationen für die Installation und den Gebrauch der Software auf einem PC.

Die technische Literatur von Danfoss Drives ist auch online unter www.danfoss.com/drives verfügbar.

□ Symbole und Konventionen

In diesem Handbuch verwendete Symbole:

ACHTUNG!:
Kennzeichnet einen wichtigen Hinweis.

Kennzeichnet eine allgemeine Warnung.

Kennzeichnet eine Warnung vor gefährlicher elektrischer Spannung.

***** Markiert in der Auswahl die Werkseinstellung.

Konventionen

Die Informationen in diesem Handbuch sind weitestgehend systematisiert und typografisch folgendermaßen beschrieben:

Menüs und Funktionen, Befehle und Parameter

Menüs und Funktionen sind kursiv geschrieben, zum Beispiel *Steuerung* → *Parameter*.

Befehle und Parameternamen sind in Großbuchstaben geschrieben, zum Beispiel: AXEND und KPROP; Parameter sind kursiv geschrieben, zum Beispiel: *Proportionalfaktor.*

Parameter-Einstellungen

Werte, die für Parameter-Einstellungen ausgewählt werden können, stehen in eckigen Klammern, z. B. [3].

Tasten

Die Namen der Tasten und Funktionstaten stehen ebenfalls in eckigen Klammern, zum Beispiel die Steuerungstaste [Strg]-Taste oder nur [Strg], die [Esc]-Taste oder die [F1]-Taste.

□ Abkürzungen

Ampere, Milliampere	A, mA
Automatische Motor Anpassung	AMA
Benutzereinheiten	BE
Gleichstrom	DC
Digitaler Signal-Prozessor	DSP
Frequenzumrichter	FU
Hauptistwert	HIW
Hauptsollwert	HSW
LCP Bedieneinheit	LCP
Bit mit dem niedrigsten Stellenwert	LSB
Motion Control Option	MCO
Motion Control Tool	MCT
Minute	Min
Maschinennullpunkt	MN
Höchstwertiges Bit	MSB
Master Unit	MU

Schalter normalerweise geschlossen	NC
Schalter normalerweise offen	NO
Nach plus schaltender digitaler Ausgang	NPN
Parameter	Par.
PID Regelung	PID
Nach minus schaltender digitaler Ausgang	PNP
Pulse pro Umdrehung [PPR]	Pulse/U
Quadcounts	qc
Sekunde, Millisekunde	s, ms
Abtastzeit (Sample time)	st
Steuerwort	STW
Umdrehungen pro Minute	U/Min
Volt	V
Zustandswort	ZSW

□ Definitionen

□ MLONG

Eine untere oder obere Grenze für viele Parameter ist:

-MLONG = -1.073.741.824

MLONG = 1.073.741.823

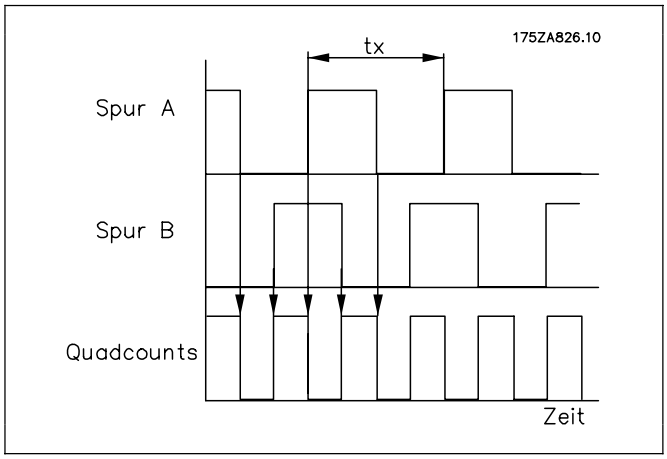
□ Online / Offline Parameter

Änderungen der Online-Parameter werden sofort nach Änderung des Datenwertes aktiviert. Änderungen der Offline-Parameter werden erst dann aktiviert, wenn am LCP [OK] gedrückt wurde.

□ Quadcounts

Inkrementalgeber: 4 Quadcounts entsprechen einer Drehgeber-Umdrehung.
Absolutgeber: 1:1 (1 qc entspricht einer Drehgeber-Umdrehung).

Aus den beiden Spuren (A/B) der Inkrementalgeber wird durch Flankenauswertung eine Vervierfachung der Inkremente erzeugt. Dies verbessert die Auflösung.



Ableitung der Quadcounts.

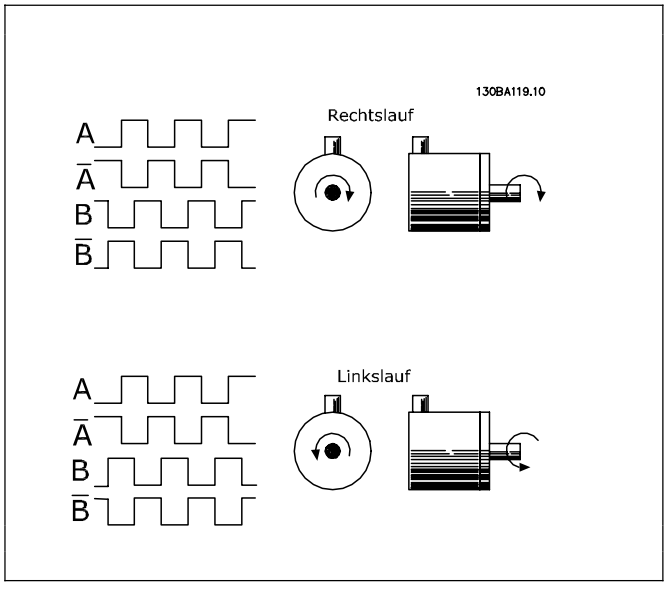
□ Drehgeber-Drehrichtung

Die Drehrichtung eines Drehgebers wird dadurch bestimmt, wie die Pulse in den Antrieb einfließen:

Rechtsdrehend heißt, dass Kanal A 90° (elektrische Grad) vor Kanal B liegt.

Linksdrehend heißt, dass Kanal B 90° (elektrische Grad) vor Kanal A liegt.

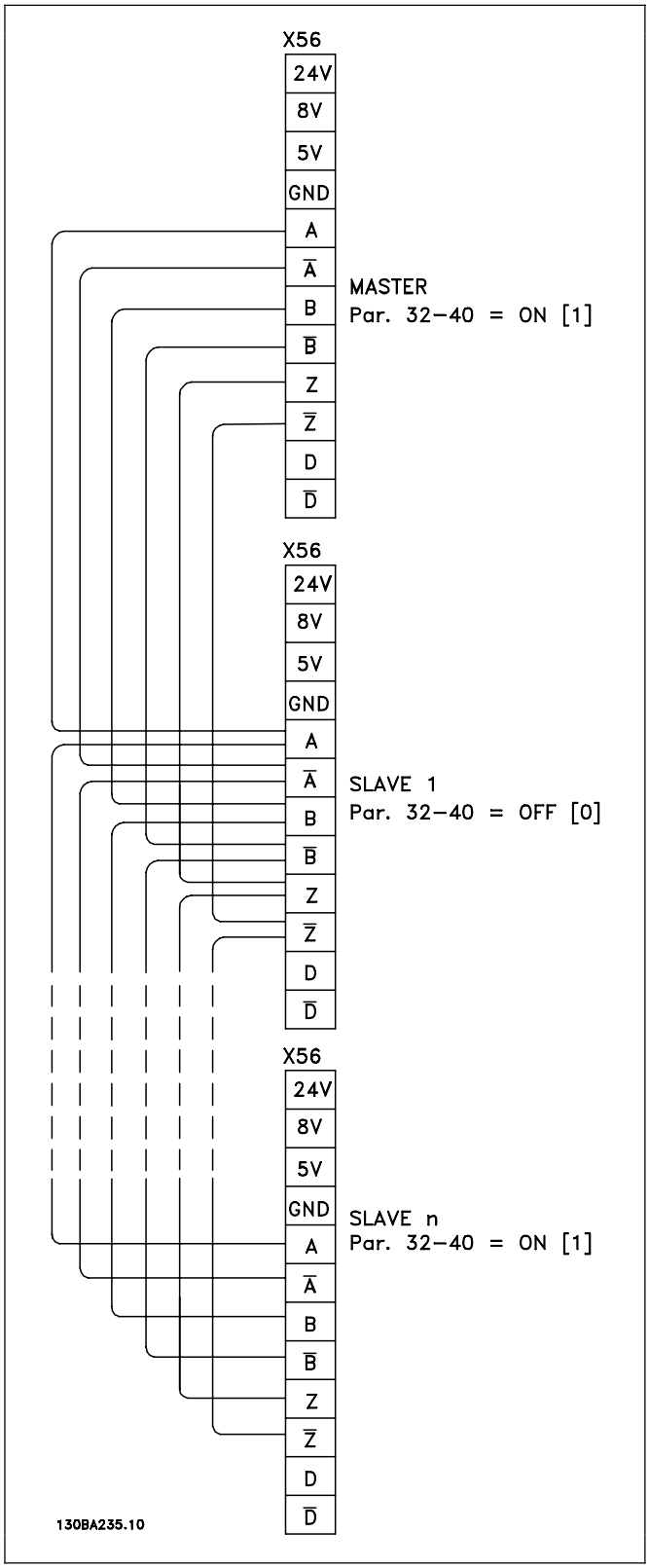
Die Drehrichtung erkennt man, wenn man auf das Wellenende schaut.



__ Projektierungshandbuch lesen __

☐ Virtueller Master

Ein virtueller Master ist eine Drehgeber-Simulation, die ein gewöhnliches Master-Signal für eine Synchronisation für bis zu 32 Achsen unterstützt.



□ Benutzereinheiten

Die Einheiten für den Antrieb oder den Slave und den Master können in beliebiger Weise definiert werden, so dass der Anwender mit sinnvollen Werten arbeiten kann.

Benutzereinheiten [BE]

Wegangaben in Fahrbefehlen erfolgen immer in Benutzereinheiten und werden intern in Quadcounts umgerechnet. Diese wirken sich auf alle Befehle für das Positionieren aus: z.B. APOS, POS.

Auch für die Kurvenscheibensteuerung kann der Anwender sinnvolle Einheiten wählen, um die Kurve für den Master und den Slave zu beschreiben. Zum Beispiel 1/100 mm oder bei Anwendungen, bei denen eine Umdrehung betrachtet wird 1/10 Grad.

Bei der Kurvenscheibensteuerung wird der maximale Fahrabstand des Slaves bzw. die Zykluslänge des Slaves in Benutzereinheiten BE [qc] angegeben.

Sie normieren die Einheit mit einem Faktor. Dieser ist ein Bruch, der sich aus Zähler und Nenner zusammensetzt:

$$1\ \text{Benutzereinheit [BE]} = \frac{\text{Par. 32 - 12 Benutzerfaktor Zähler}}{\text{Par. 32 - 11 Benutzerfaktor Nenner}}$$

Par. 32-12 *Benutzerfaktor Zähler* POSFACT_Z
Par. 32-11 *Benutzerfaktor Nenner* POSFACT_N

Die Normierung bestimmt, wie viele Quadcounts eine Benutzereinheit ergeben: Wenn der Faktor zum Beispiel 50375/1000 beträgt, entspricht eine BE genau 50,375 qc.

ACHTUNG!:
Wenn die Benutzereinheiten in qc umgerechnet werden, wird der Integer-Wert benutzt. Wenn qc in Benutzereinheiten umgerechnet werden, wird gerundet.

Master Units [MU]

Die Kurvenlänge bzw. Master-Zykluslänge und andere Angaben (zum Beispiel der Markerabstand) für die Kurvenscheibensteuerung werden in Master-Units MU angeben.

$$1\ \text{Master Unit [MU]} = \frac{\text{Par. } 33-10\ \text{ Synchronisationsfaktor Master}}{\text{Par. } 33-11\ \text{ Synchronisationsfaktor Slave}}$$

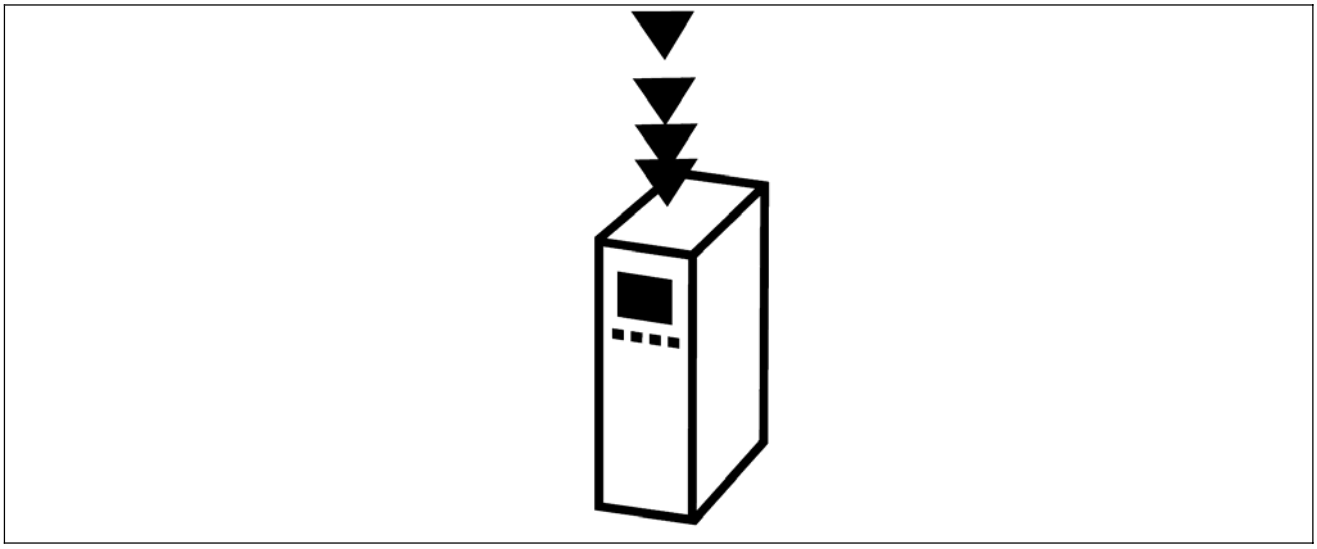
Par. 33-10 *Synchronisationsfaktor Master* SYNCFACTM
Par. 33-11 *Synchronisationsfaktor Slave* SYNCFACTS

□ Offener Regelkreis vs. geschlossenen Regelkreis (Open-Loop / Closed-Loop)

Unter „Open-Loop" (offener Regelkreis) versteht man eine Steuerung ohne Rückführung. „Closed-Loop"-Steuerungen (geschlossener Regelkreis) vergleichen die zurückgelieferte Geschwindigkeit oder Position mit der Sollgeschwindigkeit bzw. mit der Sollposition und erzeugen einen modifizierten Befehl um den Fehler zu verringern. Der Fehler ist die Differenz zwischen der erforderlichen Drehzahl und der Ist-Drehzahl.

Open-Loop kann in Systemen benutzt werden, wo weder die Motorgeschwindigkeit kritisch ist, noch eine exakte Positionierung erforderlich ist. Gebläse- oder Pumpensteuerungen und andere einfache Anwendungen sind Beispiele dafür.

Einführung in VLT Motion Control Option MCO 305



□ Was ist eine VLT Motion Control Option MCO 305?

MCO 305 ist eine integrierte programmierbare Steuerung für die beiden VLT Automation Drives FC 301 und FC 302; sie ergänzt die schon sehr umfassenden Standardfunktionen dieser Antriebe mit weiterer Funktionalität und hoher Flexibilität.

FC 301 und FC 302 mit MCO 305 sind intelligente Antriebe, die hohe Genauigkeit und Dynamik für Steuerungsaufgaben sowie für die Synchronisation (elektronische Welle), die Positionierung und die elektronische Kurvenscheibensteuerung (CAM) bieten. Zusätzlich zur Programmierbarkeit bietet MCO 305 eine Vielfalt von Anwendungsfunktionen wie Monitoring und eine ausgefeilte Fehlerbehandlung.

Die Entwicklungs- und Anwendungsprogramme für die MCO 305 sowie die Konfiguration und Inbetriebnahme werden mittels einer einfach zu benutzenden PC-Software erstellt, die im VLT Motion Control Tool MCT 10 integriert ist. Die PC Software enthält einen Editor zum Programmieren mit Programmbeispielen und einen Editor zum Erstellen der Kurvenprofile sowie „Testfahrt"- und „Scope"-Funktionen zum Optimieren der Steuerung. MCO 305 basiert auf eine ereigniskontrollierte Programmierung, die eine strukturierte Makro-Programmiersprache benutzt, die eigens für die Anwendung entwickelt und optimiert wurde.

FC 301 und FC 302 können als „all-in-one"-Antrieb mit einem vorinstallierten MCO 305 Modul geliefert werden oder eine MCO 305 wird als Option für die Installation im Feld geliefert.

Basisfunktionen und Spezifikationen:

- HOME Funktion.
- Absolute und relative Positionierung.
- Software- und Hardware-Begrenzung.
- Geschwindigkeits-, Positions- und Marker-Synchronisation.
- Kurvenscheibensteuerung (CAM).
- Virtuelle Masterfunktion zum Synchronisieren von mehreren Slaves.
- Online einstellbare Getriebeübersetzungen.
- Online einstellbarer Offset.
- Definition der Anwendungsparameter über das FC 300 Kontrollpanel.
- Lese/Schreib-Zugang zu allen FC 300 Parametern.
- Daten senden und empfangen über das Feldbus-Interface (erfordert die Feldbus-Option).
- Interrupt-Steuerung durch verschiedene Ereignisse: Digitaler Eingang, Position, Feldbus Daten, Parameter- oder Status-Änderung und Zeit.
- Operatoren, Vergleichsoperationen, Bitoperationen und logische Verknüpfungen.
- Bedingte und unbedingte Sprungbefehle.
- Grafische PID-Optimierung.
- Debugging-Funktionen.
- Unterstützte Drehgebertypen: 5V Inkremental RS422 und SSI absolut Single- und Multiturn, Gray Code, einstellbare Taktfrequenz und Datenlänge.
- 3 Versorgungsspannungen: 5 V, 8 V und 24 V.

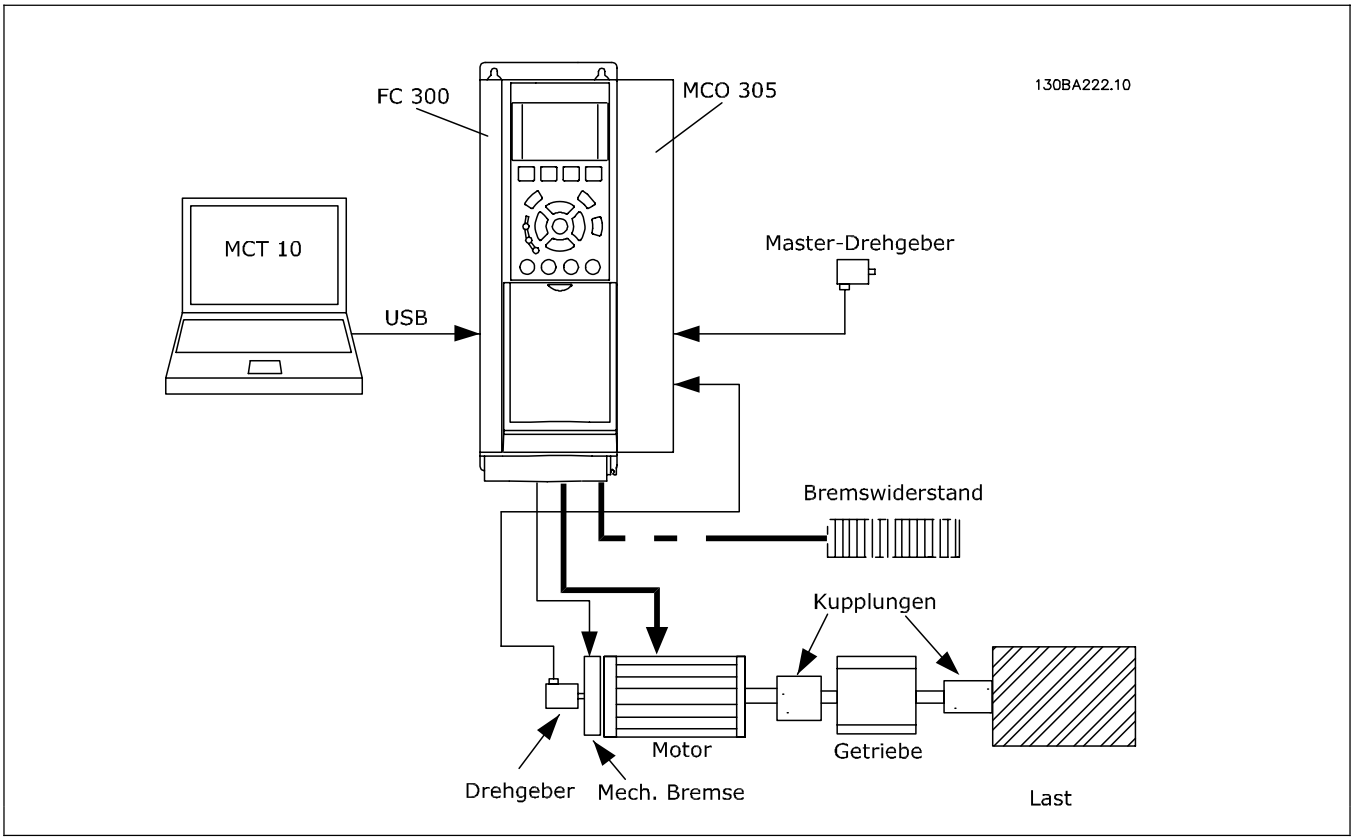
□ Systemüberblick

Das MCO 305 System enthält mindestens folgende Elemente:

- FC 300.
- MCO 305 Modul.
- Motor/Getriebemotor.
- Drehgeber mit Rückführung. Der Drehgeber muss auf der Motorwelle montiert sein, wenn der FC 300 mit Fluxvektor mit Rückführung benutzt wird. Der Drehgeber mit Rückführung zum Positionieren und Synchronisieren kann überall in der Anwendung montiert werden. Sehen Sie auch „Konfigurationsbeispiele".
- Master-Drehgeber (nur zum Synchronisieren).
- PC mit MCT 10 zum Programmieren.

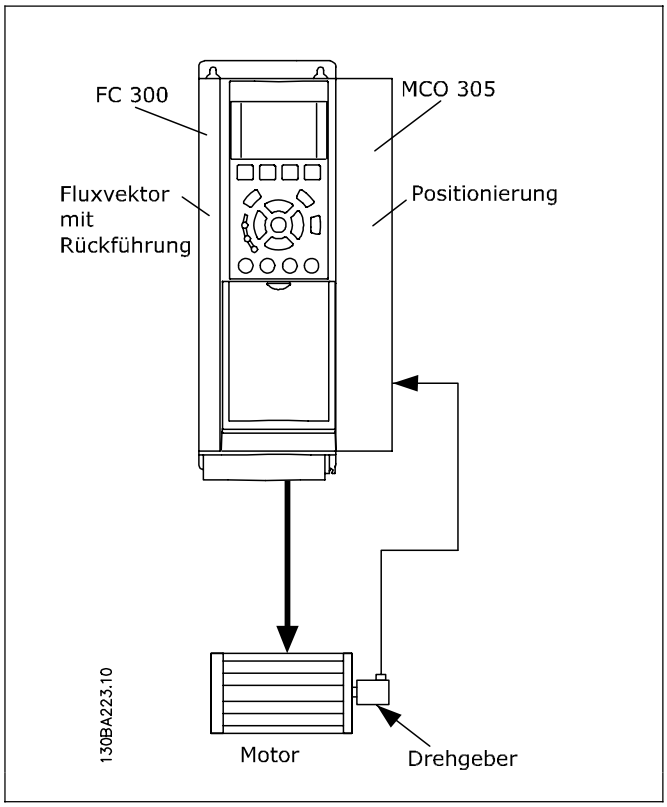
Folgendes kann auch erforderlich sein:

- Bremswiderstand für elektrische Bremsung
- Mechanische Bremse.

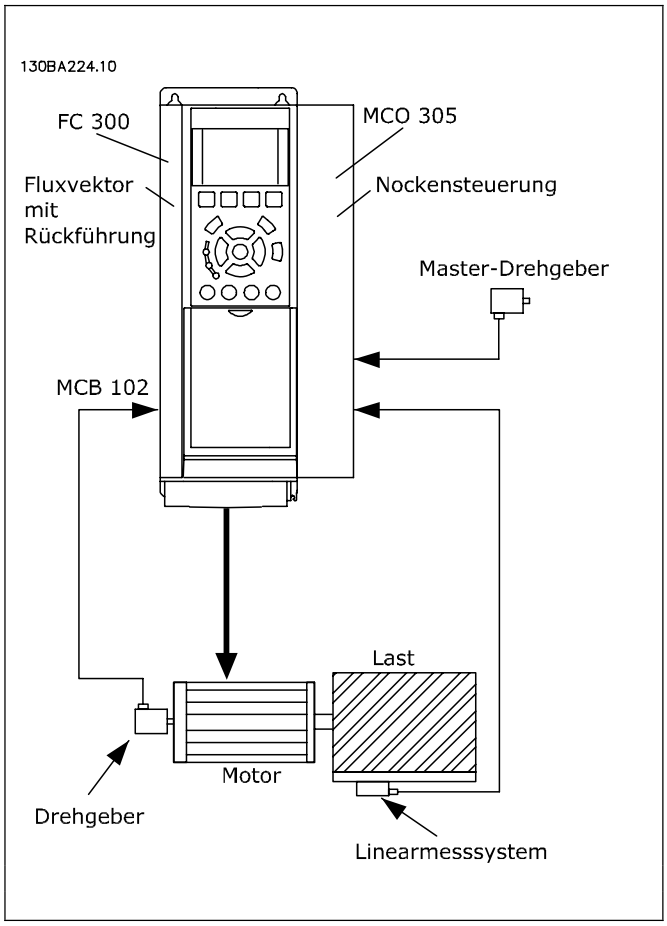


□ Konfigurationsbeispiele

Ein Drehgeber wird sowohl als Motor-Rückmeldung für Fluxvektor-Regelung als auch für die Positions-Rückmeldung verwendet.



Ein Drehgeber wird als Motor-Rückmeldung für die Fluxvektor-Regelung mit Rückführung verwendet (über die Drehgeber-Option MCB 102 angeschlossen), ein Linear-Drehgeber wird zur Slave-Positions-Rückmeldung benutzt und ein dritter Drehgeber als Master.



□ Schnittstellen zwischen MCO 305, FC 300 und anderen Options-Modulen

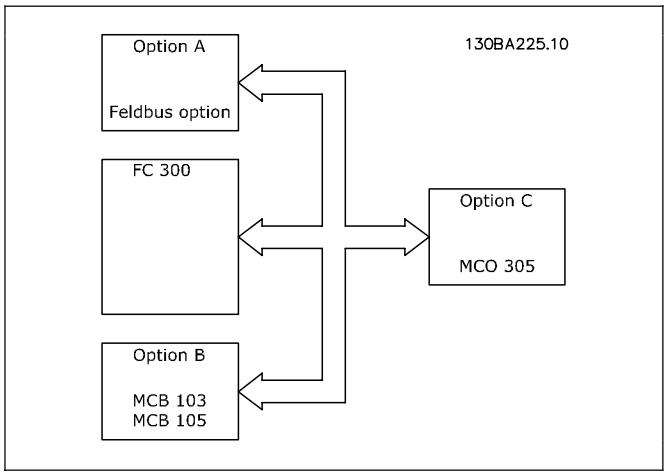
Die Schnittstelle zwischen einer MCO 305 und der FC 300 Steuerkarte ermöglicht sowohl das Lesen und Schreiben von allen Parametern als auch das Lesen des Status von allen Eingängen sowie die Steuerung von allen Ausgängen. Zusätzlich können verschiedene Prozessdaten wie das Statuswort und der aktuelle Motorstrom mit dem MCO 305 Anwendungsprogramm ausgelesen werden.

MCO 305 steuert den FC 300 über Soll-Drehzahl/Drehmoment; sehen Sie dazu auch den Abschnitt „PID-Regelung".

Feldbus-Schnittstelle (z.B. PROFIBUS und DeviceNet): MCO 305 hat einen Lese/Schreib-Zugang zu den erhaltenen bzw. gesendeten Daten über verschiedene Feldbus-Schnittstellen (dies erfordert eine Feldbus-Modul als Option).

Relais Option MCB 105: Die Relais-Ausgänge von MCB 105 können durch das MCO 305 Anwendungsprogramm gesteuert werden.

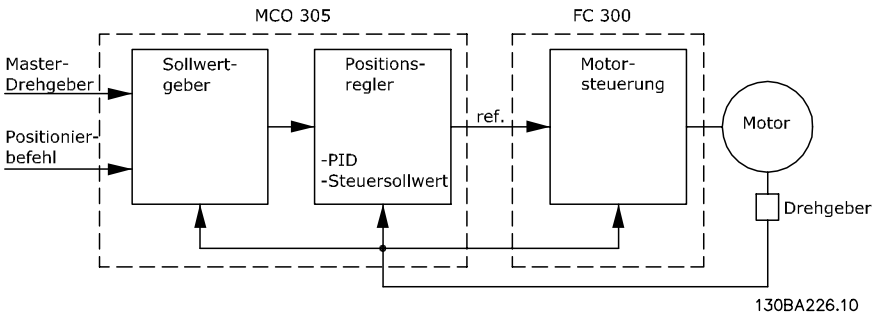
Mehrzweck-I/O-Option MCB 103: Mit dem MCO 305 Anwendungsprogramm kann der Status der Eingänge gelesen und können die Ausgänge gesteuert werden.



MCO 305 Anwendungsprogramme und Konfigurationsdaten werden über die FC 300 Schnittstelle (RS485 oder USB) oder via PROFIBUS DPV1 hoch- oder heruntergeladen (erfordert die Option PROFIBUS-Modul). Dasselbe gilt für Online-PC-Software-Funktionen wie Testfahrt und Fehlersuche (Debugging).

□ PID-Regelung

MCO 305 hat eine PID-Regelung (Proportional, Integral, Differential) für die Positionierung, die auf der Istposition (Drehgeber-Rückführung) und der Sollposition (berechnete Position) basiert. Die MCO 305 PID-Regelung steuert in allen Betriebsmodi die Position außer bei der Geschwindigkeits-Synchronisation, bei der statt dessen die Geschwindigkeit geregelt wird. Der FC 300 wirkt im MCO 305 Regelkreis wie ein „Verstärker" und muss deshalb für den angeschlossenen Motor und die Last optimiert werden, bevor die MCO 305 PID-Regelung eingerichtet werden kann. Der FC 300 kann in einem offenen oder geschlossenen Regelkreis innerhalb der MCO 305 Regelung betrieben werden, siehe folgendes Beispiel:



Einen Leitfaden für die Optimierung der MCO 305 PID-Regelung finden Sie im MCO 305 Produkthandbuch.

Einen Leitfaden für die Optimierung des FC 300 finden Sie im FC 300 Produkthandbuch.

□ Drehgeber

MCO 305 unterstützt verschiedene Drehgebertypen:

- Inkrementalgeber mit RS422 Signaltyp.
- Inkrementalgeber mit sinus–cosinus Signaltyp.
- Absolutgeber mit SSI Schnittstelle.

Master- und Feedback/Slave-Drehgebertypen können unabhängig voneinander ausgewählt werden; als Geber können Dreh- oder Lineargeber benutzt werden. Die Auswahl des Gebertyps hängt von den Anforderungen der Anwendung und von dem allgemein bevorzugten Typ ab. Es gibt drei wichtige Auswahlkriterien:

- Maximale Positioniergenauigkeit ist ±1 Geberinkrement.
- Um eine stabile und dynamische Steuerung sicherzustellen, werden mindestens 20 Geberinkremente pro PID-Regelungszyklus (Standard ist 1 Millisekunde) für die Mindestgeschwindigkeit der Anwendung benötigt.
- Die maximale Frequenz der MCO 305 Drehgebereingänge darf bei maximaler Geschwindigkeit nicht überschritten werden.

Der Drehgeber mit Rückführung (Feedback-Drehgeber) kann direkt auf die Motorwelle oder hinter die Getriebe und/oder anderen Übersetzungen montiert werden. Es gibt jedoch einige wichtige Problemkreise, die beim Montieren der Drehgeber beachtet werden müssen:

- Es sollte eine feste Verbindung zwischen Motor und Drehgeber sein. Schlupf, Nachlauf (Totgang) und Elastizität würden die Genauigkeit und Stabilität der Steuerung verringern.
- Wenn der Drehgeber mit langsamer Geschwindigkeit läuft, muss er eine hohe Auflösung haben um das oben Geforderte einzuhalten. (Mindestens 20 Drehgeber-Inkremente pro Abtastzyklus.)

□ Programmausführung

MCO 305 kann bis zu 90 Programme speichern. Aber nur eines dieser Programme kann zur gleichen Zeit ausgeführt werden. Es gibt drei Arten das Programm das ausgeführt werden soll zu bestimmen:

- Mit Parameter 33-80 *Aktivierte Programmnummer*.
- Über die digitalen Eingänge (Parameter 33-50 bis 33-59, 33-61 und 33-62).
- Mit der PC Software.

Ein Programm muss als *Autostart*-Programm definiert sein. Das Autostart-Programm wird automatisch nach dem Einschalten ausgeführt. Ohne Autostart-Programm kann man ein Programm nur mit der PC-Software ausführen.

Das Autostart-Programm wird immer zuerst ausgeführt. Wenn das Autostart-Programm beendet ist (kein LOOP oder EXIT Befehl) kann Folgendes auftreten:

1. Wenn Parameter 33-80 (*Aktivierte Programmnummer)* = -1 und kein Eingang (Parameter 33-50 bis 33-59, 33-61 und 33-62) als *Programmausführung starten* ([13] oder [14]) definiert ist: Es wird wieder das Autostart-Programm gestartet.
2. Wenn Parameter 33-80 (*Aktivierte Programmnummer)* ≠ -1 und kein Eingang (Parameter 33-50 bis 33-59, 33-61 und 33-62) als *Programmausführung starten* ([13] oder [14]) definiert ist: Es wird das ausgewählte Programm (Par. 33-80) ausgeführt.
3. Wenn ein Eingang (Parameter 33-50 bis 33-59, 33-61 und 33-62) als *Programmausführung starten* ([13] oder [14]) definiert ist und einer oder mehrere Eingänge als *Programmwahl* ([15]) bestimmt sind: Das ausgewählte Programm (Programmwahl-Eingänge) wird ausgeführt, sobald der Eingang für *Programmausführung starten* aktiviert wird.

Das aktive Programm kann über einen digitalen Eingang abgebrochen werden, wenn ein Eingang als *Programmausführung abbrechen* (Option [9] oder [10] in 33-50 bis 33-59, 33-61 und 33-62) festgelegt ist. Das abgebrochene Programm kann wieder über einen digitalen Eingang gestartet werden, wenn ein solcher als *Programmausführung fortsetzen* (Option [11] oder [12] in 33-50 bis 33-59, 33-61 und 33-62) definiert ist.

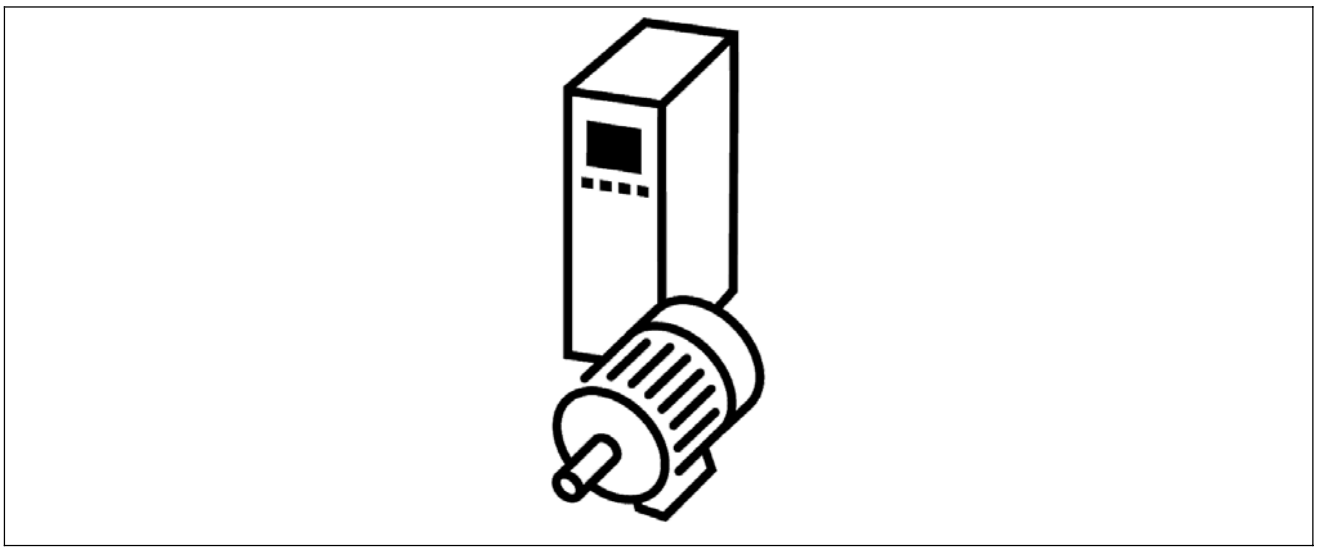
Das Starten des Autostart-Programms nach dem Einschalten kann durch Drücken der [Cancel]-Taste auf dem FC 300 LCP während des Hochfahrens vermieden werden. Die Taste muss solange gedrückt werden, bis die Meldung „Benutzerabbruch" (Fehler 119) im Display erscheint.

Ein temporäres Programm kann aus dem Editor (MCT10/APOSS) heraus ausgeführt werden. Temporäre Programme werden nur im RAM gespeichert und sind daher nach dem Ausschalten verloren. Das temporäre Programm kann auch in einem speziellen Debug-Modus ausgeführt werden, in dem es möglich ist, die Programmausführung zu beeinflussen sowie die Daten und Variablen auszulesen. (Details dazu finden Sie auch in der APOSS-Online-Hilfe.)

Das Verbinden eines PC mit MCT 10 mit einem Antrieb kann das aktive Programm abbrechen, z.B. wenn ein neues Programm heruntergeladen wird oder wenn mit dem Programm-Editor gearbeitet wird. ([Esc] bricht die Programmausführung ab.)

ACHTUNG!:
Wenn ein Fehler das aktive Programm beendet und keine Fehlerbehandlung (ON ERROR GOSUB xxxx) definiert ist, wird das Programm nicht mehr starten.

Funktionen und Beispiele



□ Positionierung

Grundsätzlich bedeutet „Positionierung" in Verbindung mit einem Antrieb, die Achse auf eine bestimmte Position fahren. Um eine exakte Positionierung zu erhalten, ist es notwendig in einem geschlossenen Regelkreis die Istposition auf Basis der Positionsrückführung eines Drehgebers zu steuern.

Eine Positionierung mit einer Steuerung in einem geschlossenen Regelkreis erfordert Folgendes: Eine festgesetzte Geschwindigkeit, Beschleunigung und Zielposition, dass ein Geschwindigkeitsprofil auf Basis der Istposition auf der Achse sowie der zuvor erwähnten Parameter berechnet ist, und dass die Achse entsprechend dem Geschwindigkeitsprofil bewegt wird bis die Zielposition erreicht ist.

Typische Anwendungen, bei denen eine exakte Positionierung notwendig ist, sind:

- Palettierer, zum Beispiel Flaschenkästen auf eine Palette stapeln.
- Sortiertische, zum Beispiel um Material in Wannen oder Fächern auf einem rotierenden Tisch zu füllen.
- Transportbänder, zum Beispiel um Material auf Länge zu schneiden.
- Aufzüge, zum Beispiel ein Fahrstuhl der in verschiedenen Ebenen hält.

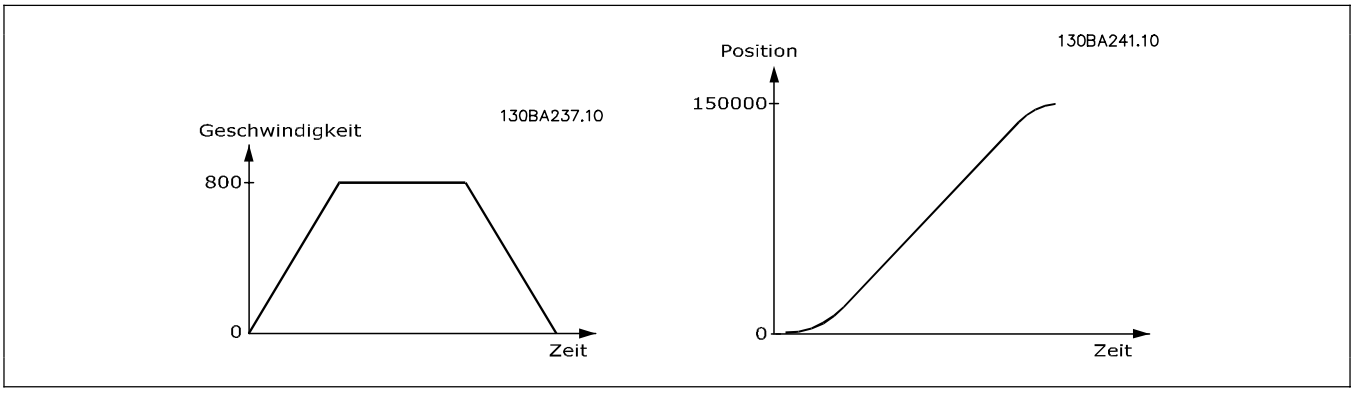
MCO 305 bietet drei Hauptpositionierungsarten:

- Absolut
- Relativ
- Touch Probe

Absolute Positionierung

Eine absolute Positionierung bezieht sich immer auf den absoluten Nullpunkt eines Systems, das bedeutet, dass dieser definiert sein muss, bevor eine absolute Positionierung ausgeführt werden kann. Wenn Inkrementalgeber eingesetzt werden, wird der Nullpunkt mit der HOME Funktion festgesetzt, die den Antrieb zum Referenzschalter fährt, stoppt und die Istposition als Nullpunkt definiert. Wenn Absolutgeber eingesetzt werden, ist der Nullpunkt durch den Drehgeber vorgegeben.

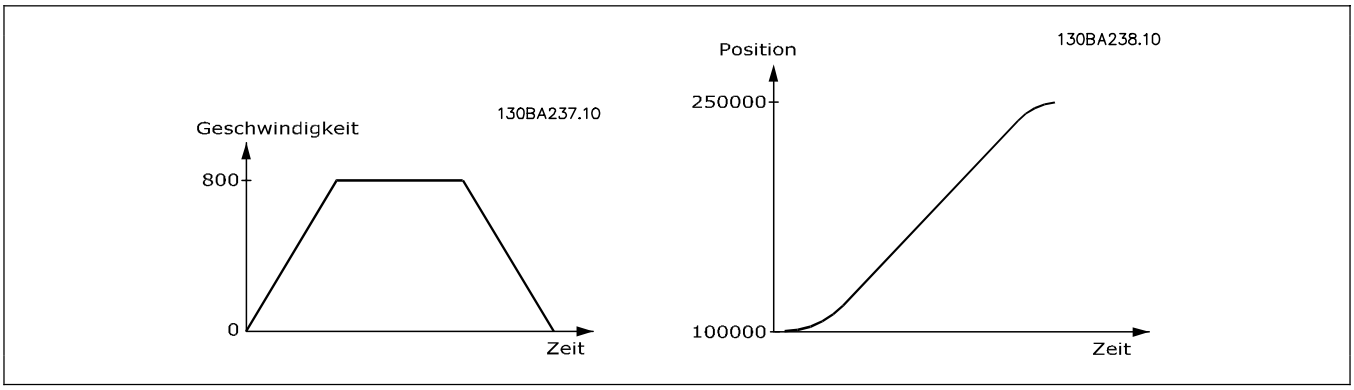
Wenn die Startposition 0 ist und bei einer absoluten Positionierung auf 150.000 die Zielposition 150.000 ist, wird der Antrieb also eine Distanz von 150.000 zurücklegen. Falls andererseits die Startposition 100.000 ist, bleibt bei einer absoluten Positionierung auf 150.000 die Zielposition weiterhin 150.000, aber der Antrieb wird nur über eine Distanz von 50.000 bewegt, weil er auf die Position 150.000 bezogen zum Nullpunkt fährt.



Relative Positionierung

Eine relative Positionierung ist immer auf die Istposition bezogen; deshalb ist es möglich eine Positionierung durchzuführen, ohne den absoluten Nullpunkt zu definieren.

Wenn die Startposition 100.000 ist, mit einer relativen Positionierung auf 150.000, dann ist die Zielposition 250.000 (100.000 + 150.000); die Fahrdistanz beträgt also 150.000.

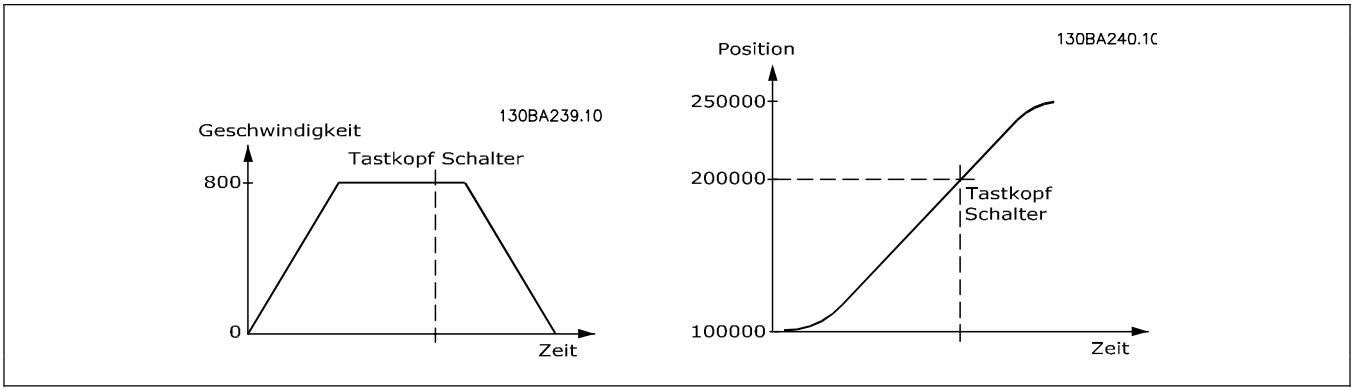


Touch-Probe Positionierung

Bei einer Touch-Probe Positionierung wird die Positionierung auf die Istposition bezogen wenn der Touch-Probe-Eingang aktiviert wird, das heißt die Zielposition ist die Position der Touch Probe plus der Positionierdistanz. Eine Touch-Probe Positionierung ist daher eine relative Positionierung bezogen auf einen Marker statt auf eine aktuelle Startposition.

Touch-Probe ist ein Sensor; es kann ein mechanischer Schalter sein, ein Näherungssensor, ein optischer Sensor oder Ähnliches. Sobald der Sensor aktiviert ist, zum Beispiel durch eine Kiste auf einem Transportband, wird die Referenz für die Positionierung gesetzt.

Bei einer Touch-Probe Positionierung auf Position 50.000 läuft der Antrieb, bis der Touch-Probe-Sensor zum Beispiel auf Position 200.000 aktiviert wird, und fährt dann weiter bis zu seiner Zielposition von 250.000 (200.000 + 50.000). Eine Touch-Probe-Positionierung wird auch „markerabhängige" Positionierung genannt.

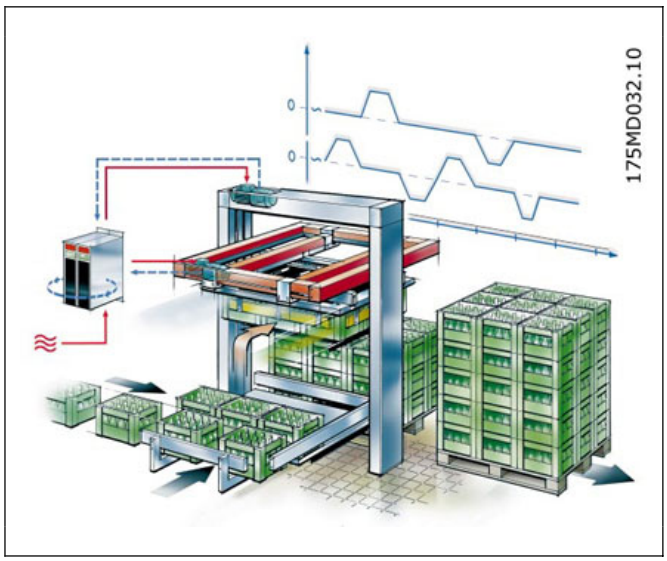


Anwendungsbeispiel: Palettierer für Flaschenkästen

Das folgende Beispiel zeigt einen Palettierer, der Flaschenkästen aufstapelt. Die Kästen werden mit einem Greifer packweise entladen und Lage für Lage auf die Palette gesetzt. Alle drei Positionierungsarten werden in diesem Beispiel benutzt und in drei Schritten erläutert.

ANMERKUNG: Das Folgende ist nur ein Beispiel und die gezeigten Einstellungen und Programme können nicht die vollständige Funktionalität abdecken, die eine reale Anwendung fordern würde.

Es wird vorausgesetzt, dass die Motor- und Drehgeber-Anschlüsse geprüft sind und dass alle grundlegenden Parameter wie Motor- und Drehgeberdaten sowie die PID-Regelung eingestellt sind. Anleitungen für die Einstellung der Parameter finden Sie in den Produkthandbüchern FC 300 und MCO 305 sowie in der Online-Hilfe.



Absolute Positionierung

Das absolute Positionieren wird mit folgender Funktion des Palettierers erklärt: Die horizontale Achse hat zwei feste Zielpositionen; eine ist über dem Greifer (Aufnehmer) und die andere über der Palette. Die horizontale Achse wird durch eine absolute Positionierung zwischen der Greiferposition und der Übergabeposition gesteuert.

Parameter-Einstellungen und Befehle für das Beispiel Palettierer (Absolute Positionierung)

Für eine absolute Positionierung sind folgende MCO 305 Parameter relevant:

Befehl	Beschreibung	Syntax	Parameter
Absolute Positionierung (ABS)			
ACC	Beschleunigung setzen.	ACC a	a = Beschleunigung
DEC	Verzögerung (negative Beschleunigung) setzen.	DEC a	a = Verzögerung
HOME	Maschinennullpunkt (Referenzschalter) anfahren und als Realnullpunkt setzen.	HOME	–
POSA	Achse absolut positionieren.	POSA p	p = Position in BE
VEL	Geschwindigkeit für relative und absolute Bewegungen sowie die maximal zulässige Geschwindigkeit zum Synchronisieren setzen.	VEL v	v = normierter Geschwindigkeitswert

☐ Programmbeispiel: Absolute Positionierung für das Anwendungsbeispiel Palettierer

```
/********************** Programmbeispiel absolute Positionierung **********************/
// Inputs:    1       Zur Greiferposition fahren
//            2       Zur Übergabeposition fahren
//            3       HOME Referenzschalter
//            8       Fehler löschen
// Outputs:   1       In Greiferposition
//            2       In Übergabeposition
//            8       Fehler
/****************************** Interrupts ************************************/
ON ERROR GOSUB errhandle
      // Bei Fehler in die Fehlerroutine springen; diese muss immer enthalten sein.
/*****************************  Grundeinstellungen  ******************************/
VEL 80          // Positionier-Geschwindigkeit bezogen auf Par. 32-80 Maximalgeschwindigkeit setzen
ACC 100         // Positionier-Beschleunigung bezogen auf Par. 32-81 kürzeste Rampe setzen
DEC 100         // Positionier-Verzögerung bezogen auf Par. 32-81 kürzeste Rampe setzen
/********************** Anwendungsparameter definieren ****************************/
LINKGPAR 1900 "Greiferposition" 0 1073741823 0
LINKGPAR 1901 "Übergabeposition" 0 1073741823 0
/******************* HOME (0) Position nach dem Hochfahren definieren *****************/
SET I_FUNCTION_3 1      // Eingang 3 als HOME Referenzschalter-Eingang setzen
HOME                    // Referenzschalter anfahren und Position auf 0 setzen
/************************** Hauptprogrammschleife *************************/
MAIN:
IF (IN 1 == 1) AND (IN 2 == 0) THEN          // wenn nur Eingang 1 high, zur Greiferposition fahren
   OUT 2 0              // Ausgang "in Übergabeposition" zurücksetzen
   POSA (GET 1900)      // Positionieren
   OUT 1 1              // Ausgang "in Greiferposition" setzen
ELSEIF (IN 1 == 0) AND (IN 2 == 1) THEN      // wenn nur Eingang 2 high, zur Übergabeposition fahren
   OUT 1 0              // Ausgang "in Greiferposition" setzen
   POSA (GET 1901)      // Positionieren
   OUT 2 1              // Ausgang "in Übergabeposition" setzen
ELSE
   MOTOR STOP           // Anhalten, falls beide Eingänge low oder high sind.
ENDIF
GOTO MAIN
/************************ Unterprogramm starten ***********************************/
SUBMAINPROG
/************************** Fehlerbehandlung ***************************************/
SUBPROG errhandle
   err = 1      // Fehler-Flag setzen, um solange in der Fehlerroutine zu bleiben, bis der Fehler gelöscht ist.
   OUT 8 1              // Ausgang für Fehler setzen
   WHILE err DO         // In der Fehlerroutine bleiben, bis die Reset-Meldung empfangen ist.
   IF IN 8 THEN         // Fehlermeldung zurücksetzen wenn Eingang 8 high.
      ERRCLR            // Fehler löschen.
      err=0             // Fehler-Flag zurücksetzen.
   ENDIF
   ENDWHILE
   OUT 8 0              // Ausgang Fehler zurücksetzen
RETURN
/*****************************************************************************/
ENDPROG
/***************************  Programmende  ***********************************/
```

□ Relative Positionierung

Die relative Positionierung wird mit folgender Funktion des Palettierers erklärt: Wenn die Übergabeposition verlassen wird, muss sich die vertikale Achse nur um eine Kastenhöhe nach oben bewegen, damit sie frei ist vom Stapel, bevor die horizontale Achse zur Greiferposition zurückfahren kann. Dies wird durch relatives Positionieren der „Kastenhöhe" und der „Aufwärtsrichtung" erreicht.

□ Parametereinstellungen und Befehle für das Beispiel Palettierer (Relative Positionierung)

Für eine relative Positionierung sind folgende MCO 305 Parameter relevant:

Befehl	Beschreibung	Syntax	Parameter
Relative Positionierung (REL)			
ACC	Beschleunigung setzen	ACC a	a = Beschleunigung
DEC	Negative Beschleunigung setzen.	DEC a	a = Verzögerung
POSR	Relativ zur Istposition positionieren	POSR d	d = Distanz zur Istposition in BE
VEL	Geschwindigkeit setzen	VEL v	v = normierter Geschwindigkeitswert

□ Programmbeispiel: Relative Positionierung für das Anwendungsbeispiel Palettierer

```
/************ Programmbeispiel zur relativen Positionierung für einen Palettierer ************/
//          Eingänge:    1     Positionieren
//                       8     Fehler zurücksetzen
//          Ausgänge:    1     in Position
//                       8     Fehler
/***************************** Interrupts ***************************************/
ON ERROR GOSUB errhandle       // Bei Fehler in die Fehlerroutine springen; diese muss immer enthalten sein.
/************************  Flags definieren  ***********************************/
flag = 0
/*********************   Grundeinstellungen   **********************************/
VEL 80        // Positioniergeschwindigkeit bezogen auf Par. 32-80 Maximalgeschwindigkeit setzen.
ACC 100       // Positionierbeschleunigung bezogen auf Par. 32-81 kürzeste Rampe setzen.
DEC 100       // Positionierverzögerung bezogen auf Par. 32-81 kürzeste Rampe setzen.
/******************* Anwendungsparameter definieren ******************************/
LINKGPAR 1900 "Box high" 0 1073741823 0
/**************************  Hauptprogrammschleife  ****************************/
MAIN:
IF (IN 1 == 1) AND (flag == 0) THEN     // 1 x Positionieren (durch Flag abgesichert) wenn Eingang 1 high.
   OUT 1 0                  // Ausgang "in Position" zurücksetzen.
   POSR (GET 1900)          // Positionieren
   OUT 1 1                  // Ausgang "in Position" setzen.
   flag = 1                 // "Flag" setzen, um sicherzustellen, dass die Distanz nur einmal gefahren wird.
ELSE
   MOTOR STOP               // Stopp wenn Eingang low ist.
   flag = 0                 // "Flag" zurücksetzen, um neue Positionierung freizugeben.
ENDIF
GOTO MAIN
/***********************  Unterprogramm starten  ********************************/
SUBMAINPROG
/*************************  Fehlerroutine  **************************************/
```

```
SUBPROG errhandle
err = 1        // Fehler-Flag setzen, um solange in der Fehlerroutine zu bleiben, bis der Fehler zurückgesetzt ist.
   OUT 8 1                 // Ausgang für Fehler setzen.
   WHILE err DO            // In der Fehlerroutine bleiben, bis die Reset-Meldung empfangen ist.
      IF IN 8 THEN         // Fehlermeldung zurücksetzen wenn Eingang 8 high.
         ERRCLR            // Fehler löschen.
         err=0             // Fehler-Flag zurücksetzen.
      ENDIF
   ENDWHILE
   OUT 8 0                 // Ausgang Fehler zurücksetzen.
   flag = 0                // "Flag" zurücksetzen, um neue Positionierung freizugeben.
RETURN
/*****************************************************************************/
ENDPROG
/*********************  Programmende  ***********************************/
```

□ Touch-Probe Positionierung

Die Touch-Probe Positionierung wird mit folgender Funktion des Palettierers erklärt:

Wenn die horizontale Achse in der Übergabeposition ist, gibt es für die vertikale Achse zahlreiche Zielpositionen abhängig von der Höhe des schon vorhandenen Kastenstapels, der wiederum von der Kastenhöhe und der Anzahl der Lagen abhängt. Dies wird mit einer Touch-Probe Positionierung gesteuert, wobei der Touch-Probe-Sensor das obere Ende des Stapels erkennt, um die Übergabeposition zu diesem zu berechnen.

□ Parametereinstellungen und Befehle für das Beispiel Touch-Probe Positionierung

Für eine Touch-Probe Positionierung sind folgende MCO 305 Parameter relevant:

32-0*	Drehgeber 2 – Slave	Seite 187
32-6*	PID-Regelung	Seite 194
32-8*	Geschwindigkeit & Beschleunigung	Seite 196
33-4*	Grenzwertbehandlung	Seite 209

Befehl	Beschreibung	Syntax	Parameter
Touch Probe			
ON INT	Interrupt-Eingang definieren	ON INT n GOSUB name	n = Nummer des Eingangs, der überwacht werden soll 1 - 8 = Reaktion auf steigende Flanke –1 - 8 = Reaktion auf fallende Flanke name = Name des Unterprogramms
ACC	Beschleunigung setzen	ACC a	a = Beschleunigung
DEC	Negative Beschleunigung setzen	DEC a	a = Verzögerung
POSR	Relativ zur Istposition positionieren	POSR d	d = Distanz zur Istposition in BE
CVEL	Geschwindigkeit für drehzahlgeregelte Motorbewegungen setzen	CVEL v	v = Geschwindigkeitswert (negativer Wert für Reversieren)
CSTART	Drehzahlmodus starten	–	–

□ Programmbeispiel: Touch-Probe Positionierung für die Anwendung Palettierer

```
/**************  Programmbeispiel Touch-Probe Positionierung für Palettierer  ************/
//      Inputs:      1     Positionieren
//                   2     Touch-Probe
//                   8     Fehler löschen
//      Outputs:     1     in Position
//                   8     Fehler
/********************************** Interrupts ***********************************/
ON ERROR GOSUB errhandle     // Bei Fehler in die Fehlerroutine springen; diese muss immer enthalten sein.
ON INT 2 GOSUB tp_handler    // Touch-Probe-Routine aufrufen wenn positive Flanke an Eingang 2.
/*********************************  Flags definieren  ********************************/
flag = 0
tp_active = 0
/*****************************  Grundeinstellungen  ******************************/
VEL 80       // Positioniergeschwindigkeit bezogen auf Par. 32-80 Maximalgeschwindigkeit setzen.
ACC 100      // Positionierbeschleunigung bezogen auf Par. 32-81 kürzeste Rampe setzen.
DEC 100      // Positionierverzögerung bezogen auf Par. 32-81 kürzeste Rampe setzen.
/************************** Anwendungsparameter definieren  *************************/
LINKGPAR 1900 "Touch probe distance" 0 1073741823 0
/**************************  Hauptprogrammschleife  *******************************/
MAIN:
IF (IN 1 == 1) AND (flag == 0) THEN    // 1 x Bewegung starten (durch Flag abgesichert) wenn Eingang 1 high.
   OUT 1 0                  // Ausgang "in Position" zurücksetzen.
   CVEL 80                  // Konstante Geschwindigkeit setzen.
   CSTART                   // Mit konstanter Geschwindigkeit starten.
   tp_active = 0            // "tp_active" zurücksetzen, um ein neue Touch-Probe Positionierung freizugeben.
   flag = 1                 // "Flag" setzen, um sicherzustellen, dass die Distanz nur einmal gefahren wird.
ELSE
   MOTOR STOP               // Stopp wenn Eingang low ist.
   flag = 0                 // "Flag" zurücksetzen, um neuen Start freizugeben.
ENDIF
GOTO MAIN
/****************************** Unterprogramme starten ***************************/
SUBMAINPROG
/****************************** Touch-Probe Routine ******************************/
SUBPROG tp_handler
   IF (tp_active == 0) THEN
      POSR (GET 1900)    // Zur Touch-Probe Zielposition fahren.
      WAITAX             // Programmausführung anhalten bis die Position erreicht ist.
      // (Dies ist notwendig, weil NOWAIT ON automatisch in einem Unterprogramm,
      // das durch einen Interrupt aufgerufen wird, gesetzt wird).
      OUT 1 1            // Ausgang "in Position" setzen.
      tp_active = 1      // "tp_active" setzen, um sicherzustellen,
                         // dass die Touch-Probe Positionierung nur einmal ausgeführt wird.
   ENDIF
RETURN
/******************************** Fehlerroutine **********************************/
SUBPROG errhandle
   err = 1            // Fehler-Flag setzen, um in der Fehlerroutine zu bleiben, bis der Fehler zurückgesetzt ist.
   OUT 8 1                  // Ausgang Fehler setzen.
   WHILE err DO             // In der Fehlerroutine bleiben, bis die Reset-Meldung empfangen ist.
      IF IN 8 THEN          // Fehlermeldung zurücksetzen wenn Eingang 8 high.
         ERRCLR             // Fehler löschen.
         err=0              // Fehler-Flag zurücksetzen.
      ENDIF
   ENDWHILE
   OUT 8 0                  // Ausgang Fehler zurücksetzen.
   flag = 0                 // "Flag" zurücksetzen, um eine neue Positionierung freizugeben.
RETURN
/*****************************************************************************/
ENDPROG
/*****************************  Programmende  ********************************/
```

□ Synchronisation

Eine Synchronisation wird in Anwendungen benutzt, in denen zwei oder mehrere Achsen einander in Geschwindigkeit oder Position folgen müssen. Es kann ein einfaches Master-Slave-System sein, in dem ein Slave der Geschwindigkeit oder Position eines Masters folgt. Es kann auch ein Multi-Achsensystem sein, wo mehrere Slaves der Geschwindigkeit oder Position eines gemeinsamen Master-Signals folgen. Eine elektronische Synchronisation ist äußerst flexibel im Vergleich zu einer mechanischen Welle, Kette oder einem Treibriemen, weil die Getriebeübersetzung und der Positionsoffset während des Betriebs eingestellt werden kann. Geschwindigkeit und Position des Slave-Antriebs werden basierend auf ein Master-Drehgebersignal, ein Feedback-Drehgebersignal sowie dem gesetzten Getriebeverhältnis gesteuert.

Während der Synchronisation ist der Slave immer durch die maximale Geschwindigkeit und Beschleunigung/Verzögerung (Parameter Gruppe 33-8*) begrenzt. Zusätzlich kann die erlaubte Abweichung zwischen Master- und Slave-Geschwindigkeit durch den Parameter 33-14 beschränkt sein, z.B. bedeutet Par. 33-14 = 5 %, dass der Slave nur 5 % schneller oder langsamer sein kann, als die aktuelle Master-Geschwindigkeit, wenn Positionskorrekturen gemacht werden.

MCO 305 bietet die drei Hauptarten der Synchronisation:

Für den synchronen Betrieb von zwei oder mehreren Antrieben können Sie Folgende benutzen:

- Geschwindigkeitssynchronisation
- Positionssynchronisation
- Markersynchronisation

□ Geschwindigkeitssynchronisation (SYNCV)

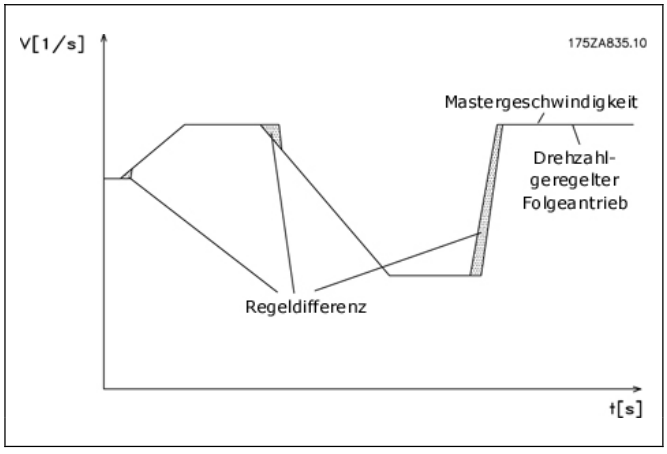
Die Geschwindigkeitssynchronisation (SYNCV) ist eine Geschwindigkeitssteuerung im geschlossenen Regelkreis, bei der die Mastergeschwindigkeit multipliziert mit dem Getriebefaktor der Positions-Sollwert ist und die aktuelle Geschwindigkeit durch den Slave-Drehgeber gemessen wird; Positionsabweichungen werden nicht korrigiert. Beachten Sie jedoch, dass das Benutzen des Integral-Anteils der PID-Regelung zum teilweisen Ausgleich der Positionskorrektur führt, weil die Integralsumme der Geschwindigkeit der Position entspricht.

Der Slave muss mindestens so schnell und dynamisch sein wie der Master, um eine exakte Synchronisation zu erhalten, das heißt der Slave muss in der Lage sein, die maximale Geschwindigkeit, Beschleunigung und Verzögerung des Masters zu erreichen.

Schon während der Projektierungsphase ist es deshalb wichtig zu überlegen, ob die am wenigsten dynamische Achse zum Master erklärt wird, weil diese Achse sowieso die Rahmenbedingung der Systemleistung bestimmen wird.

Typische Anwendungen sind:

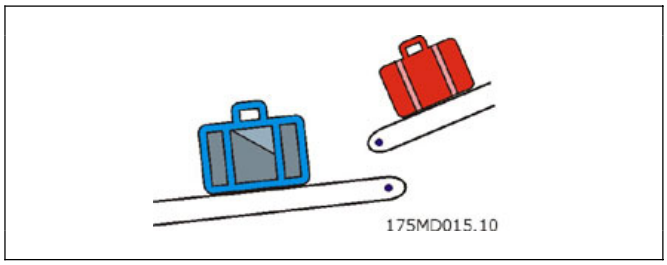
- Synchronisieren von zwei oder mehr Transportbändern
- Strecken von Materialien
- Mischen



Regelungsverhalten bei Geschwindigkeitssynchronisation.

□ Anwendungsbeispiel: Koffertransportband

Zwei oder mehrere Transportbänder müssen mit der gleichen Geschwindigkeit laufen, um eine gleichmäßige Übergabe der Koffer von einem Transportband auf das nächste zu erhalten.



Zusätzlich zum Start und Stopp der Geschwindigkeitssynchronisation ist im Programmbeispiel ein manueller Modus enthalten, der es erlaubt die Geschwindigkeit über die digitalen Eingänge zu erhöhen oder zu verringern.

ANMERKUNG: Das Folgende ist nur ein Beispiel und die gezeigten Einstellungen und Programme können nicht die vollständige Funktionalität abdecken, die eine reale Anwendung fordern würde.

Es wird vorausgesetzt, dass die Motor- und Drehgeber-Anschlüsse geprüft sind und dass alle grundlegenden Parameter wie Motor- und Drehgeberdaten sowie die PID-Regelung eingestellt sind. Anleitungen für die Einstellung der Parameter finden Sie in den Produkthandbüchern FC 300 und MCO 305 sowie in der Online-Hilfe.

□ Parametereinstellungen und Befehle für das Anwendungsbeispiel Koffertransportband

Folgende MCO 305 Parameter sind relevant für eine Geschwindigkeitssynchronisation:

Befehl	Beschreibung	Syntax	Parameter
SYNCV	Geschwindigkeitssynchronisation	SYNCV	–
ON ERROR GOSUB	Fehlerunterprogramm definieren	ON ERROR GOSUB name	name = Name des Unterprogramms

□ Programmbeispiel: Geschwindigkeitssynchronisation

```
/******************  Beispielprogramm Geschwindigkeitssynchronisation  ********************/
// Eingänge:    1      Start/Stopp Synchronisation
//              2      Start manuellen Modus
//              3      Geschwindigkeit manuell erhöhen
//              4      Geschwindigkeit manuell verringern
//              8      Fehler löschen
// Ausgänge:    1      Im Synchronisations-Modus
//              2      Im manuellen Modus
//              8      Fehler
/********************************** Interrupts ***************************************/
ON ERROR GOSUB errhandle       // Bei Fehler in die Fehlerroutine springen; diese muss immer enthalten sein.
/*************************  Grundeinstellungen  ***********************************/
VEL 100      // Maximale Slave-Geschwindigkeit bezogen auf Par. 32-80 Maximalgeschwindigkeit setzen.
ACC 100      // Maximale Slave-Beschleunigung bezogen auf Par. 32-81 kürzeste Rampe setzen.
DEC 100      // Maximale Slave-Verzögerung bezogen auf Par. 32-81 kürzeste Rampe setzen.
/*********************** Anwendungsparameter definieren ****************************/
LINKGPAR 1900 "Manuelle Geschwindigkeit" 0 100 0
LINKGPAR 1901 "Geschwindigkeitsstufe" 0 10 0
/************************** Flags und Variablen definieren ***************************/
sync_flag = 0
done = 0
err = 0
man_vel = 0
```

```
/************************ Hauptprogrammschleife ************************/
MAIN:
IF (IN 1 == 1) AND (sync_flag == 0) THEN        // Synchronisierung einmal starten, wenn Eingang 1 high.
   SYNCV            // Modus Geschwindigkeitssynchronisation starten
   sync_flag = 1    // "sync_flag" setzen, um sicherzustellen, dass die Synchronisation nur einmal startet.
   OUT 1 1          // Ausgang "Im Synchronisations-Modus" setzen.
ELSE
   MOTOR STOP             // Anhalten falls Eingang 1 low.
   sync_flag = 0          // Nach Stopp "sync_flag" zurücksetzen.
   OUT 1 0                // Ausgang "Im Synchronisations-Modus" zurücksetzen.
ENDIF
IF (IN 2 == 1) AND (sync_flag == 0) THEN
   // Manuellen Modus starten, wenn Eingang 2 high und die Synchronisation nicht läuft.
   OUT 2 1                // Ausgang "Im manuelles Modus" setzen.
   man_vel = GET 1900     // Geschwindigkeit manuell auf Parameter 1900 setzen.
   CVEL man_vel
   CSTART                       // Konstanten Drehzahlmodus starten.
   WHILE (IN 2 == 1) DO         // Im manuellen Modus bleiben, solange Eingang 2 high.
      CVEL man_vel              // Geschwindigkeit manuell aktualisieren.
      IF (IN 3 == 1) AND (done == 0) THEN
         // Geschwindigkeit manuell stufenweise erhöhen, wenn Eingang 3 gesetzt ist.
         man_vel = man_vel + GET 1901
         done = 1
      ELSEIF (IN 4 == 1) AND (done == 0) THEN
         // Geschwindigkeit manuell um eine Stufe verringern, wenn Eingang 3 gesetzt ist.
         man_vel = man_vel - GET 1901
         done = 1
      ELSE
         done = 0
      ENDIF
   ENDWHILE
   CSTOP        // Anhalten, wenn der manuelle Modus verlassen wird.
   OUT 2 0      // Ausgang "Im manuellen Modus" zurücksetzen, wenn der manuelle Modus verlassen wird.
ENDIF
GOTO MAIN
/****************************** Unterprogramm starten  *******************************/
SUBMAINPROG
/******************************** Fehlerbehandlung  ********************************/
SUBPROG errhandle
   err = 1       //  Fehler-Flag setzen, um solange in der Fehlerroutine zu bleiben, bis der Fehler gelöscht ist.
   OUT 8 1               // Ausgang Fehler setzen.
   OUT 1 0               // Ausgang "Im Synchronisations-Modus" bei einem Fehler zurücksetzen.
   OUT 2 0               // Ausgang "Im manuellen Modus" bei einem Fehler zurücksetzen.
   WHILE err DO          // In der Fehlerroutine bleiben, bis die Reset-Meldung empfangen ist.
      IF (IN 8) AND NOT (IN 1) AND NOT (IN 2) THEN
         // Fehler zurücksetzen, wenn der Eingang 8 high und die Eingänge 1+2 low.
         ERRCLR         // Fehler löschen
         err=0          // Fehler-Flag zurücksetzen.
      ENDIF
   ENDWHILE
   OUT 8 0               // Ausgang Fehler zurücksetzen
   sync_flag = 0         // sync_flag nach einem Fehler zurücksetzen
   done = 0              // "done"-Flag nach einem Fehler zurücksetzen
RETURN
/*****************************************************************************/
ENDPROG
/****************************** Programmende *******************************/
```

□ Position/Winkel-Synchronisation (SYNCP)

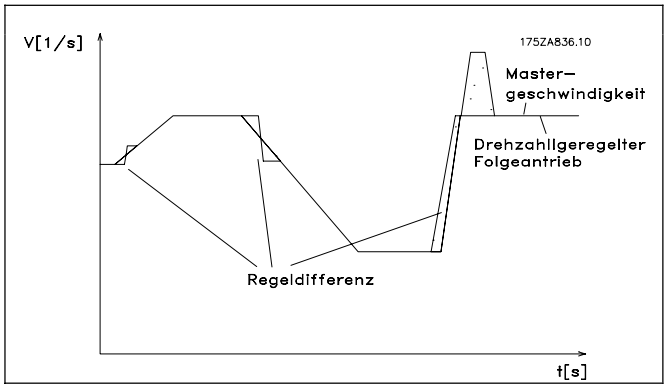
Eine Positionssynchronisation (SYNCP) ist eine Positionsregelung mit Rückführung eines bewegten Ziels, wobei der Sollwert (Sollposition) die Master-Position multipliziert mit der Getriebeübersetzung ist und ein jeder Positionsoffset berücksichtigt wird. Die Slave-Position wird basierend auf diesen Sollwert und der aktuellen Istposition des Slave-Drehgebers gesteuert. Jede Positionsabweichung wird kontinuierlich entsprechend der maximalen Geschwindigkeit, Beschleunigung und Verzögerung des Slaves korrigiert. Die Getriebeübersetzung ist als Bruch gesetzt (Zähler und Nenner) um Rundungsfehler zu vermeiden, z.B. wenn Primzahlen benutzt werden. Die Getriebeübersetzung muss 100 % genau sein; sogar der kleinste Rundungsfehler würde dazu führen, dass die Position nach gewisser Zeit wegdriftet.

Beim Starten der Positionssynchronisation rastet die aktuelle Slave-Position auf die aktuelle Master-Position ein. Daher ist es notwendig, den Slave unter Beachtung der physikalischen Position des Masters in die richtige physikalische Position zu bringen. Dies kann manuell oder durch eine automatische Homefahrt ausgeführt werden (erfordert einen externen Referenzschalter oder Absolutgeber).

Der Slave muss schneller und dynamischer als der Master sein, um sowohl bei maximaler Master-Geschwindigkeit als auch während der Beschleunigung/Verzögerung eine exakte Synchronisation zu erreichen. Das heißt, der Slave muss die maximale Geschwindigkeit, Beschleunigung und Verzögerung des Masters erreichen können, damit er in der Lage ist diesen einzuholen, falls er hinter dem Master läuft. Schon während der Projektierungsphase ist es daher wichtig, zu überlegen, ob die am wenigsten dynamische Achse zum Master erklärt wird, weil diese Achse sowieso die Rahmenbedingung der Systemleistung bestimmen wird.

Typische Anwendungen sind:

- Flaschenwaschanlagen.
- Folienverpackung.
- Verpackungsmaschinen.
- Transportbänder.
- Mehrfach-Achsen-Hebeanlagen.
- Abfüllanlagen.
- Druckmaschinen.
- Fliegende Messer.

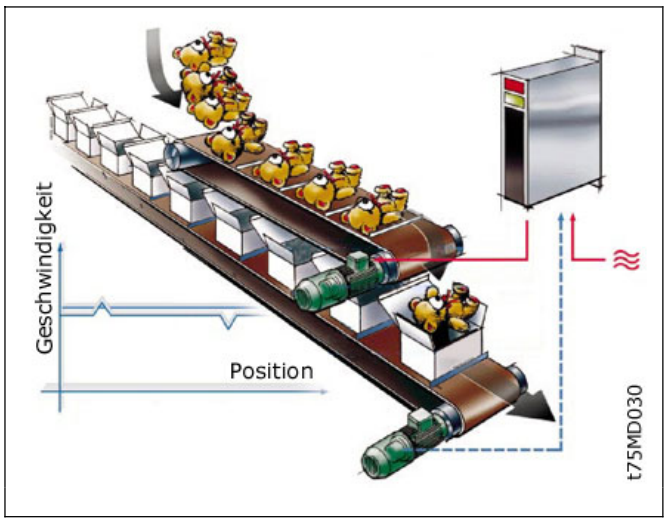


Regelungsverhalten bei Positionssynchronisation

□ Anwendungsbeispiel: Verpacken mit festen Produktabständen

Diese Anwendung besteht aus zwei Transportbändern: Eines befördert leere Kartons, ein anderes Teddybären. Aufgabe der Anlage ist es, die Teddybären in die Kartons zu packen. Beides, Kartons und Teddybären kommen mit festen Abständen und es ist sichergestellt, dass es zwischen den Drehgebern und den Kartons und Teddys keinen Schlupf gibt. Daher ist eine Positionssynchronisation auf Basis der Drehgeber ausreichend.

Beim Starten muss sichergestellt werden, dass der Master (Karton-Förderband) immer auf der gleichen Position ist, während das Teddy-Förderband eine Homefahrt benötigt, bevor die Synchronisation gestartet wird.



Es gibt drei Möglichkeiten, um sicherzustellen, dass die Teddys beim Start passend zu den Kartons ausgerichtet sind:

- Physikalische Position des Home-Referenzschalters justieren.
- Home-Offset in Parameter 33-01 angleichen.
- Positionsoffset für Synchronisation in Parameter 33-12 angleichen.

ANMERKUNG: Das Folgende ist nur ein Beispiel und die gezeigten Einstellungen und Programme können nicht die komplette Funktionalität abdecken, die eine reale Anwendung fordern würde.

Es wird vorausgesetzt, dass die Motor- und Drehgeber-Anschlüsse geprüft sind und dass alle grundlegenden Parameter wie Motor- und Drehgeberdaten sowie die PID-Regelung eingestellt sind. Anleitungen für die Einstellung der Parameter finden Sie in den Produkthandbüchern FC 300 und MCO 305 sowie in der Online-Hilfe.

☐ Parametereinstellungen und Befehle für das Anwendungsbeispiel Positionssynchronisation

Folgende MCO 305 Parameter sind relevant für eine Positionssynchronisation:

Befehl	Beschreibung	Syntax	Parameter
DEF SYNCORIGIN	Definiert das Verhältnis Master:Slave für den nächsten SYNCP oder SYNCM Befehl.	DEFSYNCORIGIN master slave	master = Sollposition in qc slave = Sollposition
MOVESYNCORIGIN	Synchronisationsursprung relativ verschieben.	MOVESYNCORIGIN mwert	mwert = Relativer Offset
PULSACC	Beschleunigung für den virtuellen Master setzen.	PULSACC a	a = Beschleunigung in Hz/s
PULSVEL	Geschwindigkeit für den virtuellen Master setzen.	PULSVEL v	v = Geschwindigkeit in Pulsen pro Sekunde [Hz]
SYNCP	Winkel/Positionssynchronisation	SYNCP	–
SYNCSTAT	Flag für Synchronisationsstatus abfragen.	erg = SYNCSTAT	–
SYNCERR	Aktuellen Synchronisationsfehler des Slaves abfragen.	erg = SYNCERR	–

☐ Programmbeispiel: Positionssynchronisation

```
/************************  Beispielprogramm Positionssynchronisation  ********************/
// Eingänge:    1  Start/Stopp Synchronisation
//              2  Start Homefahrt
//              3  Home Referenzschalter
//              4  Offset erhöhen
//              5  Offset verringern
//              8  Fehler löschen
// Ausgänge:    1  Innerhalb der Synchronisationsgenauigkeit das Genauigkeitsfenster in Par. 33-13 setzen
//              2  Homefahrt ausgeführt
//              8  Fehler
/***************************** Interrupts ************************************/
ON ERROR GOSUB errhandle  // Bei Fehler in die Fehlerroutine springen; diese muss immer enthalten sein.
```

```
/*************************    Grundeinstellungen    ******************************/
VEL 100       // Maximale Slave-Geschwindigkeit bezogen auf Par. 32-80 Maximalgeschwindigkeit setzen.
ACC 100       // Maximale Slave-Beschleunigung bezogen auf Par. 32-81 kürzeste Rampe setzen.
DEC 100       // Maximale Slave-Verzögerung bezogen auf Par. 32-81 kürzeste Rampe setzen.
/********************** Anwendungsparameter definieren  ************************/
LINKGPAR 1900 "Offset Schrittweite" 0 10000 0
LINKGPAR 1901 "Offset Typ" 0 1 0
/**********************  Parameter und Flags setzen  ***************************/
SET I_FUNCTION_3 1              // Eingang 3 als Home Referenzschalter-Eingang definieren
next_step = 0
home_done = 0
new_offset = 0
/*************************  Hauptprogrammschleife  ***************************/
MAIN:
IF (IN 2 == 1) THEN             // Wenn Eingang 2 high, Homefahrt starten
   HOME                         // Auf Home-Position fahren und diese auf 0 setzen
   home_done = 1                // Flag home_done setzen
   OUT 2 1                      // Ausgang "Home ausgeführt" setzen
ENDIF
IF (IN 1 == 1) AND (home_done == 1) THEN
   // Synchronisation starten, wenn Eingang 1 = 1 und Homefahrt ausgeführt
   SYNCP                                // Modus Positionssynchronisation starten
   old_offset = GET SYNCPOSOFFS
   WHILE (IN 1 == 1) DO                 // Im Synchronisationsmodus bleiben, solange Eingang 1 = 1
      IF (IN 4 == 1) THEN
         GOSUB increase_offset
      ELSEIF (IN 5 == 1) THEN
         GOSUB decrease_offset
      ENDIF
      IF (SYNCSTAT & 4) THEN
         OUT 1 1
      ELSE
         OUT 1 0
      ENDIF
   ENDWHILE
   MOTOR STOP                   // Anhalten wenn Eingang 1 low.
   home_done = 0                // Flag home_done nach dem Anhalten zurücksetzen
   OUT 1 0
   OUT 2 0                      // Ausgang "Homefahrt ausgeführt" nach dem Anhalten zurücksetzen
   IF (new_offset != old_offset) AND (GET 132 == 0) THEN       // Absoluten Offset speichern, falls geändert.
      SAVE AXPARS               // ANMERKUNG: Mehr als 10000 x Speichern kann das PROM zerstören.
   ENDIF
ENDIF
GOTO MAIN
/************************** Unterprogramm starten ******************************/
SUBMAINPROG
/*****************************  Offset erhöhen  ********************************/
SUBPROG increase_offset
   IF (Next_step) THEN              // Prüfen, ob der nächste Offset-Schritt freigegeben ist.
      IF (GET 1901 == 0) THEN       // Absoluter Offset
         new_offset = old_offset + GET 1900     // Vorhandenen Offset lesen und Offset-Schrittweite addieren
         SET SYNCPOSOFFS new_offset             // Neuen Positionsoffset setzen
      ELSE                                      // Relativer Offset
```

```
            MOVESYNCORIGIN GET 1900              // Relativen Offset mit Offset-Schrittweite ausführen
        ENDIF
    ENDIF
    Next_step=0                          // Nächsten Offset-Schritt abschalten
    ON TIME 500 GOSUB Enb_Step           // Nächsten Offset-Schritt nach 500 ms anschalten
RETURN
/*************************  Offset reduzieren  ***********************************/
SUBPROG decrease_offset
    IF (Next_step) THEN                  // Prüfen, ob nächster Offset-Schritt freigegeben
        IF (GET 1901 == 0) THEN          // Absoluter Offset
            new_offset = GET SYNCPOSOFFS - GET 1900
            // Vorhandenen Offset lesen und Wert des Offset-Schritts abziehen
            SET SYNCPOSOFFS new_offset           // Neuen Positionsoffset setzen
        ELSE                                     // Relativer Offset
            MOVESYNCORIGIN (- GET 1900)          // Relativen Offset mit –Offset-Schrittweite ausführen
        ENDIF
    ENDIF
    Next_step=0                          // Nächsten Offset-Schritt abschalten
    ON TIME 500 GOSUB Enb_Step           // Nächsten Offset-Schritt nach 500 ms anschalten
RETURN
/*************************  Neuen Offset-Schritt freigeben  ******************************/
SUBPROG Enb_step
    Next_step = 1          // Nächsten Offset-Schritt freigeben
RETURN
/*****************************  Fehlerroutine  ******************************/
SUBPROG errhandle
    err = 1        // Fehler-Flag setzen, um solange in der Fehlerroutine zu bleiben, bis der Fehler gelöscht ist.
    OUT 8 1                // Ausgang Fehler setzen.
    OUT 2 0                // Bei Fehler Ausgang "Homefahrt ausgeführt" zurücksetzen
    WHILE err DO                         // In der Fehlerroutine bleiben, bis die Reset-Meldung empfangen ist.
        IF (IN 8) AND NOT (IN 1) THEN    // Fehler zurücksetzen wenn Eingang 8 high und Eingang 1 low
            ERRCLR                       // Fehler löschen
            err=0                        // Fehler-Flag zurücksetzen
        ENDIF
    ENDWHILE
    OUT 8 0                // Ausgang Fehler zurücksetzen
    home_done = 0          // Nach einem Fehler home_done Flag zurücksetzen
RETURN
/*****************************************************************************/
ENDPROG
/***************************** Programmende *********************************/
```

□ Markersynchronisation (SYNCM)

Eine Markersynchronisation (SYNCM) ist eine erweiterte Positionssynchronisation bei der zusätzliche Positionskorrekturen gemacht werden, um einen Slave-Marker an einen Master-Marker anzugleichen. Master- und Slave-Markersignale können der Drehgeber-Nullimpuls sein oder an den digitalen Ausgängen angeschlossene externe Sensoren. Wie bei der Positionssynchronisation ist es möglich Getriebeübersetzung und Offset anzugleichen. Zusätzlich kann ein Markerverhältnis gesetzt werden, z.B. 1 Master-Marker zu 3 Slave-Marker, das bedeutet dass jeder Master-Marker mit jedem dritten Slave-Marker abgeglichen wird.

Die Markersignale können durch Definition eines Positionsfensters überwacht werden; nur ein Marker (der erste) wird innerhalb des Toleranzfensters akzeptiert und jedes Markersignal außerhalb des Toleranzfensters wird ignoriert. Ohne Toleranzfenster wird jedes Markersignal inklusive Rauschen und Schwankung (Jitter) akzeptiert und benutzt, um die Slave-Position zu korrigieren. Der erste Master-Marker und der erste Slave-Marker nach dem Starten werden nicht überwacht, weil das System nicht weiß, wo der erste Marker sein wird. Sobald aber der erste Marker erkannt ist, ist auch die erwartete Position der folgenden Marker bekannt, weil der Markerabstand individuell für Master und Slave in den Parametern festgelegt sein muss.

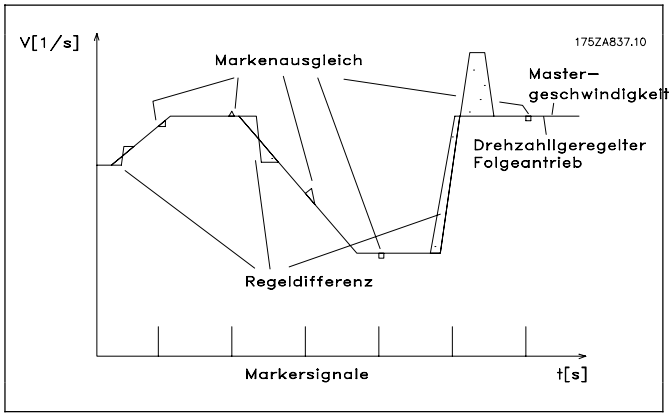
Eine Markersynchronisation verhält sich nach dem Starten anfangs wie eine Positionssynchronisation, aber sobald der erste Satz der Marker erkannt wurde, startet die Markerkorrektur. Welche Marker für die erste Markerkorrektur benutzt werden, wird in Parameter 33-23 festgelegt. Durch die Definition des Startverhaltens wird außerdem bestimmt, ob der Slave immer auf den Master warten muss, ob er auf den Master aufholt oder nur die kleinste Korrektur ausführt. Sehen Sie dazu auch die detaillierte Beschreibung der verfügbaren Möglichkeiten in Parameter 33-23. Homefahrten sind vor dem Starten nicht notwendig, weil die Markerkorrektur den Slave automatisch dem Master angleicht.

Der Slave muss schneller und dynamischer als der Master sein, um sowohl bei maximaler Master-Geschwindigkeit als auch während der Beschleunigung/Verzögerung eine die Markerkorrektur auszuführen und eine exakte Synchronisation zu erreichen. Das heißt, der Slave muss die maximale Geschwindigkeit, Beschleunigung und Verzögerung des Masters erreichen können, damit er in der Lage ist diesen einzuholen, falls er hinter dem Master läuft. Schon während der Projektierungsphase ist es daher wichtig, zu überlegen, ob die am wenigsten dynamische Achse zum Master erklärt wird, weil diese Achse sowieso die Rahmenbedingung der Systemleistung bestimmen wird.

Typische Anwendungen sind:

Grundsätzlich die gleichen Anwenden wie bei der Positionssynchronisation, aber solche bei denen eine oder mehrere der folgenden Bedingungen erfüllt sein müssen:

- Automatische Anpassung nach dem Start notwendig.
- Getriebeübersetzung kann nicht exakt auf 100 % gesetzt werden.
- Es gibt einen Schlupf irgendwo zwischen dem Drehgeber und dem Teil, das synchronisiert werden muss.
- Variierende Abstände zwischen den Produkten.



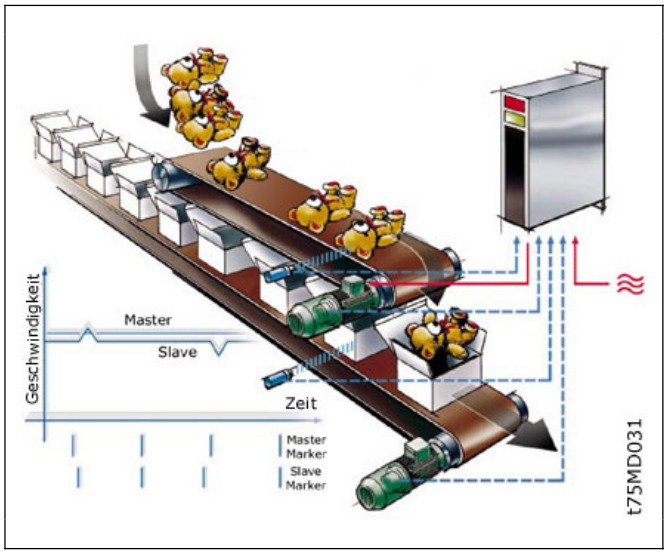
Regelungsverhalten bei Markersynchronisation

□ Anwendungsbeispiel: Verpacken mit variierenden Abständen und Schlupf

Diese Anwendung besteht aus zwei Transportbändern, eines befördert leere Kartons und das andere die Teddybären. Aufgabe der Anlage ist es, die Teddybären in die Kartons zu packen. Beide, Kartons und Teddys werden durch Reibung befördert und können sich daher auf dem Transportband bewegen. Das bedeutet, dass es kein festes Verhältnis zwischen den Drehgebern und der Position von Karton und Teddy gibt und der Abstand variieren kann. Daher ist es notwendig für Kartons (Master) und Teddys (Slave) eine externe Markererkennung zu benutzen, um die Teddybär-Position zur Karton-Position zu synchronisieren.

Der Abgleich kann durch Justieren der physikalischen Position der Markererkennung oder durch Justieren des Positionsoffsets in Parameter 33-12 erreicht werden.

Zusätzlich zum Starten und Stoppen der Markersynchronisation führt das Programmbeispiel eine Messung des *Markerabstands Master* und *Slave* aus. Damit wird der durchschnittliche Abstand zwischen den erkannten Markern berechnet und die Parameter (33-17 und 33-18) *Markerabstand* automatisch gesetzt.



ANMERKUNG: Das Folgende ist nur ein Beispiel und die gezeigten Einstellungen und Programme können nicht die komplette Funktionalität abdecken, die eine reale Anwendung fordern würde.

Es wird vorausgesetzt, dass die Motor- und Drehgeber-Anschlüsse geprüft sind und dass alle grundlegenden Parameter wie Motor- und Drehgeberdaten sowie die PID-Regelung eingestellt sind. Anleitungen für die Einstellung der Parameter finden Sie in den Produkthandbüchern FC 300 und MCO 305 sowie in der Online-Hilfe.

□ Parametereinstellungen und Befehle für das Anwendungsbeispiel Markersynchronisation

Die folgenden MCO 305 Parameter sind relevant für eine Markersynchronisation:

Befehl	Beschreibung	Syntax	Parameter
DEF SYNCORIGIN	Definiert das Verhältnis Master:Slave für den nächsten SYNCP oder SYNCM Befehl.	DEFSYNCORIGIN master slave	master = Sollposition in qc slave = Sollposition
MOVE SYNCORIGIN	Synchronisationsursprung relativ verschieben.	MOVESYNCORIGIN mwert	mwert = Relativer Offset
PULSACC	Beschleunigung für Master-Simulation setzen.	PULSACC a	a = Beschleunigung in Hz/s
PULSVEL	Geschwindigkeit für den virtuellen Master setzen.	PULSVEL v	v = Geschwindigkeit in Pulsen pro Sekunde (Hz)
SYNCM	Winkel-/Positionssynchronisation mit Markerkorrektur.	SYNCM	–
SYNCSTAT	Flag für Synchronisationsstatus abfragen.	erg = SYNCSTAT	
SYNCERR	Aktuellen Synchronisationsfehler des Slaves abfragen.	erg = SYNCERR	–
IPOS	Letzte Index- bzw. Markerposition des Slaves abfragen.	erg = IPOS	–
MIPOS	Letzte Index- bzw. Markerposition des Masters abfragen.	erg = MIPOS	–

☐ Programmbeispiel: Markersynchronisation

```
/******************  Programmbeispiel Markersynchronisation  ***********************/
// Eingänge:  1      Start/Stopp Synchronisation
//            2      Markerabstand Slave messen
//            3      Markerabstand Master messen
//            5      Master-Marker
//            6      Slave-Marker
//            8      Fehler löschen
// Ausgänge:  1      Innerhalb der Synchronisationsgenauigkeit Genauigkeitsfenster in Par. 33-13 setzen.
//            2      Marker-Messung aktiviert
//            8      Fehler
***************************** Interrupts ***************************************/
ON ERROR GOSUB errhandle   // Bei Fehler in die Fehlerroutine springen; diese muss immer enthalten sein.
/*****************************  Grundeinstellungen  *****************************/
VEL 100          // Maximale Slave-Geschwindigkeit bezogen auf Par. 32-80 Maximalgeschwindigkeit setzen.
ACC 100          // Maximale Slave-Beschleunigung bezogen auf Par. 32-81 kürzeste Rampe setzen.
DEC 100          // Maximale Slave-Verzögerung bezogen auf Parameter 32-81 kürzeste Rampe setzen.
/*********************  Anwendungsparameter definieren  *************************/
LINKGPAR 1900 "Geschwindigkeitsmessung" 0 100 0
/************************  Parameter und Flags setzen  ***************************/
SET SYNCMTYPM 2            // Markertyp Master auf externen Marker setzen
SET SYNCMTYPS 2            // Markertyp Slave auf externen Marker setzen
sync_flag = 0
/************************ Hauptprogrammschleife *************************/
MAIN:
IF (IN 1 == 1) AND (sync_flag == 0) THEN      // Wenn Eingang 1 high, Synchronisation 1 x starten
   SYNCM                                      // Marker-Synchronisations-Modus starten
   sync_flag = 1                              // "done"-Flag
ELSE
   MOTOR STOP                          // Anhalten, wenn Eingang 1 low.
   sync_flag = 0                       // Nach dem Anhalten sync_flag zurücksetzen.
ENDIF
IF (IN 2 == 1) AND (sync_flag == 0) THEN          // Markerabstand Slave messen
   GOSUB slave_measure                            // ANMERKUNG: Slave-Motor dreht sich!
ELSEIF (IN 3 == 1) AND (sync_flag == 0) THEN      // Markerabstand Master messen
   GOSUB master_measure                           // Master muss laufen
ENDIF
GOTO MAIN
/************************ Unterprogramme starten ******************************/
SUBMAINPROG
/********************** Markerabstand Slave messen ****************************/
SUBPROG slave_measure
   OUT 2 1                                 // Ausgang "Marker-Messung aktiviert" setzen
   CVEL GET 1900                           // Messgeschwindigkeit setzen
   CSTART                                  // Drehzahlmodus starten
   old_ipos = IPOS                         // "alte" Markerposition lesen
   marker_number = 0                       // Variable zurücksetzen
   total_dist = 0                          // Variable zurücksetzen
   skip_first = 0                          // Variable zurücksetzen
   WHILE (IN 2 == 1) DO                    // Im Modus "messen" bleiben, solange Ausgang 2 high.
      new_ipos = IPOS                      // "Neue" Markerposition lesen
      IF (new_ipos != old_ipos) THEN                // Prüfen, ob ein neuer Marker erkannt wurde.
         marker_distance = new_ipos - old_ipos      // Markerabstand berechnen
      IF (marker_distance < 0) THEN                 // Vorzeichen ändern, falls negativ
            marker_distance = (marker_distance * -1)
         ENDIF
         IF (skip_first == 0) THEN       // Den ersten Wert nicht verwenden, er könnte falsch sein.
```

```
            skip_first = 1
        ELSE
            marker_number = marker_number + 1    // Zähler um 1 erhöhen
            total_dist = total_dist + marker_distance // Markerabstände zusammenfassen
        ENDIF
        old_ipos = new_ipos             // "alte" Markerposition als "neue" Markerposition setzen
    ENDIF
  ENDWHILE
  CSTOP                     // Anhalten, wenn die Slave-Marker-Messung verlassen wird.
  SET SYNCMPULSS (total_dist rnd marker_number)
                            // Durchschnittlichen Markerabstand berechnen und Parameter setzen.
  OUT 2 0                   // Ausgang " Marker-Messung aktiviert " zurücksetzen
RETURN
/*********************  Markerabstand Master messen  **************************/
SUBPROG master_measure
  OUT 2 1                           // Ausgang "Marker-Messung aktiviert" setzen
  old_mipos = MIPOS                 // "alte" Markerposition lesen
  marker_number = 0                 // Variable zurücksetzen
  total_dist = 0                    // Variable zurücksetzen
  skip_first = 0                    // Variable zurücksetzen
  WHILE (IN 2 == 1) DO              // Im Messmodus bleiben solange Eingang 2 high
    new_mipos = MIPOS               // "neue" Markerposition lesen
    IF (new_mipos != old_mipos) THEN                // Prüfen, ob ein neuer Marker erkannt wurde
      marker_distance = new_mipos - old_mipos       // Markerabstand berechnen
      IF (marker_distance < 0) THEN                 // Falls negativ Vorzeichen ändern
        marker_distance = (marker_distance * -1)
      ENDIF
      IF (skip_first == 0) THEN     // Den ersten Wert nicht benutzen, er könnte falsch sein.
        skip_first = 1
      ELSE
        marker_number = marker_number + 1    // Zähler erhöhen
        total_dist = total_dist + marker_distance // Markerabstände zusammenfassen
      ENDIF
      old_mipos = new_mipos         // "alte" Markerposition auf "neue" Markerposition setzen
    ENDIF
  ENDWHILE
  SET SYNCMPULSM (total_dist rnd marker_number)
  // durchschnittlichen Markerabstand berechnen und Parameter setzen
  OUT 2 0                           // Ausgang "Marker-Messung aktiviert" zurücksetzen
RETURN
/*******************************  Fehlerroutine  ***********************************/
SUBPROG errhandle
  err = 1      // Fehler-Flag setzen, um solange in der Fehlerroutine zu bleiben, bis der Fehler gelöscht ist.
  OUT 8 1                   // Ausgang Fehler setzen.
  OUT 2 0                   // Bei Fehler Ausgang "Marker-Messung aktiviert" zurücksetzen
  WHILE err DO              // In der Fehlerroutine bleiben, bis die Reset-Meldung empfangen ist.
    IF (IN 8) AND NOT (IN 2) THEN       // Wenn Eingang 8 high und Eingang 2+3 low Fehler zurücksetzen
      ERRCLR                            // Fehler löschen
      err=0                             // Fehler-Flag zurücksetzen
    ENDIF
  ENDWHILE
  OUT 8 0                   // Ausgang Fehler zurücksetzen
  sync_flag = 0             // sync_flag nach Fehler zurücksetzen
RETURN
/*****************************************************************************/
ENDPROG
/******************************** Programmende ********************************/
```

□ Kurvenscheibensteuerung (CAM-Modus)

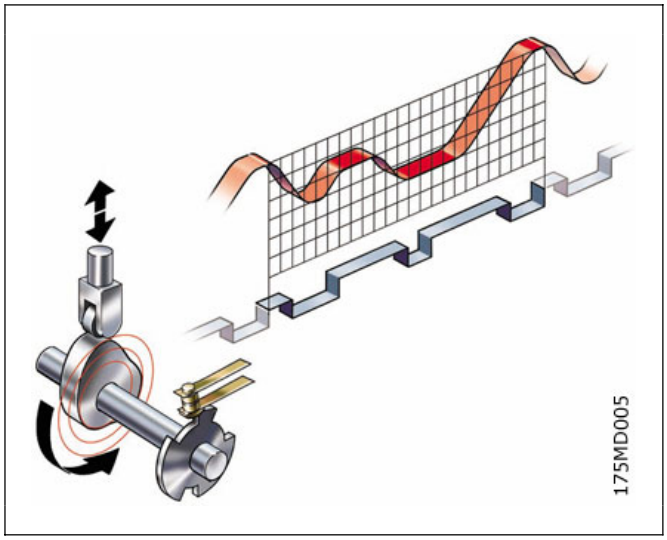
Um Kurvenscheibensteuerungen zu realisieren, benötigen Sie je nach Anwendung mindestens eine Kurve, die die Slave-Position in Abhängigkeit von der Master-Position sowie das Ein- und Auskuppelverhalten beschreibt. Natürlich sind für eine Kurvenscheibensteuerung weit mehr Parameter erforderlich, die zusammen mit den Fixpunkten der Kurve ein Kurvenprofil ergeben.

Die Synchronisation im CAM-Mode (Befehl SYNCC können Sie auch mit Markerkorrektur durchführen (SYNCCMM und SYNCCMS). Dies wäre zum Beispiel erforderlich, wenn die Produkte unregelmäßig auf einem Band transportiert werden oder wenn addierende Fehler ausgeglichen werden müssen.

Für die Erstellung des Kurvenprofils nutzen Sie den → *CAM-Editor*. Dann setzen Sie die Fixpunkte der Kurve und definieren die für Ihre Anwendung erforderlichen Parameter.

Alle Werte können Sie in physikalischen oder benutzerdefinierten Einheiten unter einer Windows-Oberfläche eingeben. Das Kurvenprofil können Sie ständig grafisch kontrollieren und so Geschwindigkeit und Beschleunigung der Slave-Achse prüfen.

Prinzipskizze: Links die mechanische Kurvenscheibe und die mechanische Nockenwelle, rechts die Kurven für die elektronische Kurvenscheibensteuerung und das elektronische Nockenschaltwerk:



□ Interpolation, Tangentenpunkte, Genauigkeit und Array

Interpolation

Der *CAM-Editor* berechnet aus den Fixpunkten die Kurve mit Hilfe einer Spline-Interpolation. Diese ist für ein minimales Drehmoment optimiert. Um Drehzahlsprünge bei mehrmaligem Kurvendurchlauf zu verhindern, wird die Geschwindigkeit am Anfang und Ende gleichgesetzt. Für diese Berechnung können Sie zwischen drei Kurventypen wählen. In jedem Fall berücksichtigt die Interpolation die Steigung der Kurve am Anfang und Ende: Entweder wird die Steigung am Anfang und Ende gemittelt, oder die Steigung am Anfang der Kurve wird auch für das Ende der Kurve benutzt, oder die Steigung am Anfang und Ende der Kurve wird auf [0] gesetzt.

Tangentenpunkte für gerade Abschnitte

Für Bereiche, in denen die Geschwindigkeit konstant und die Beschleunigung „0" sein muss, benutzen Sie Tangentenpunkte. Zwischen diesen Punkten wird statt eines Splines eine Gerade gelegt.

Genauigkeit

Die Fixpunkte werden direkt als Interpolationspunkte übernommen, sofern dies der Intervallabstand zulässt. Der *CAM-Editor* führt zwischen den Interpolationspunkten eine lineare Interpolation durch. Wird durch den gewählten Intervallabstand ein Fixpunkt nicht getroffen, fehlt der entsprechende Slave-Sollwert in der Interpolationstabelle. Wenn Sie → ☑ *Ausrichten an Gitter* aktivieren, können Sie solche Abweichungen vermeiden.

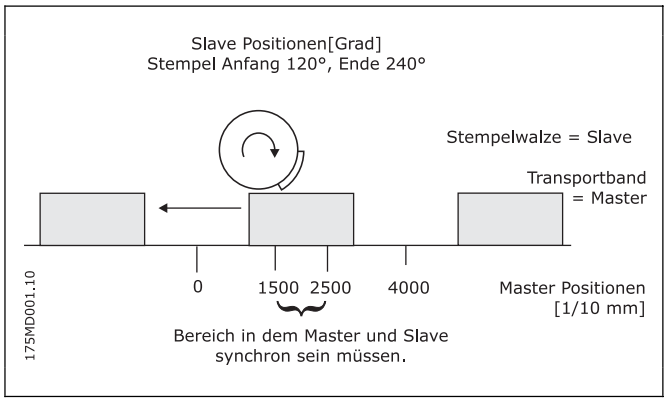
Interne Realisation als Array

Intern werden die Kurvenprofile als Arrays realisiert, die Sie mit einer DIM-Anweisung und dem Befehl SETCURVE aufrufen.

□ Anwendungsbeispiel: Kartons mit Haltbarkeitsdatum stempeln

Das folgende Beispiel zeigt, wie Sie Schritt für Schritt die Kurve für diese Anwendung der Kurvenscheibensteuerung editieren und anschließend in Ihr Steuerungsprogramm einbinden.

Eine Walze soll auf Kartons eine 10 cm lange Aufschrift stempeln. Der Stempel entspricht einem Walzenabschnitt von 120 Grad. Pro Minute werden 60 Kartons auf dem Band transportiert. Die Kartons werden exakt in immer gleichem Abstand (z.B. durch ein mechanisches Raster) auf dem Band transportiert. Während des Bedruckens müssen Stempelwalze und Karton synchron laufen:



□ Schritt für Schritt die Kurve editieren

1. FC 300 mit den erforderlichen Parametern einstellen.
2. Wählen Sie diese CNF-Datei aus; APOSS und damit die ausgewählte Datei werden daraufhin automatisch im *CAM-Editor* geöffnet.
3. Ermitteln Sie den Getriebefaktor des Masters in MU-Einheiten.

   Die Eingabe soll in 1/10 mm Auflösung möglich sein.

   Der Antrieb ist mit dem Transportband mit einer Getriebeübersetzung von 25:11 verbunden; das heißt der Motor macht 25, das Zahnriemenrad 11 Umdrehungen.

   Getriebefaktor = 25/11

   Inkrementalgeber direkt am Master-Antrieb; Dreheberauflösung = 4096

   Das Zahnriemenrad hat 20 Zähne/Umdrehung; 2 Zähne entsprechen 10 mm, daher entspricht 1 Umdrehung = 100 mm Transportbandvorschub bzw. 1000/10 mm.

   Skalierfaktor ist demnach 1000.

$$\frac{\text{Getriebefaktor} * \text{Drehgeberauflösung} * 4}{\text{Skalierfaktor}} \text{qc} = 1\,\text{MU}$$

$$\frac{25/11 * 4096 * 4}{1000} \text{qc} = \frac{25 * 4096 \times * 4}{1000 \times 11} \text{qc} = 1\,\text{MU}$$

$$= \frac{2048}{55} \text{qc} = 1\,\text{MU} = \frac{\text{Par. 33 - 10 Syncfaktor Master}}{\text{Par. 33 - 11 Syncfaktor Slave}}$$

   Tragen Sie diese Werte in der Registerkarte → *Synchronisation* ein (die gewählten Einheiten sollten immer ganzzahlig sein):

   Par. 33-10 *Syncfaktor Master* = 2048
   Par. 33-11 *Syncfaktor Slave* = 55

4. Getriebefaktor des Slaves in Benutzereinheiten BE eingeben:

   Getriebefaktor = 5/1
   Drehgeberauflösung (Inkrementalgeber) = 500

   Eine Umdrehung der Walze ist 360 Grad. Es soll mit einer Auflösung von 1/10 Grad gearbeitet werden; daher wird eine Walzenumdrehung in 3600 Arbeitseinheiten eingeteilt:
   Skalierfaktor = 3600

$$\frac{\text{Getriebefaktor} * \text{Drehgeberauflösung} * 4}{\text{Skalierfaktor}} \text{qc} = 1\,\text{BE}$$

$$\frac{5/1 * 500 * 4}{3600} \text{qc} = \frac{5 * 500 * 4}{3600} \text{qc} = 1\,\text{BE}$$

$$= \frac{25}{9} \text{qc} = 1\,\text{BE} = \frac{\text{Par. 32 - 12 Benutzerfaktor Zähler}}{\text{par. 32 - 11 Benutzerfaktor Nenner}}$$

   Tragen Sie diese ganzzahligen Werte ein in die Registerkarte → *Encoder*:

   Par. 32-12 *Benutzerfaktor Zähler* = 25
   Par. 32-11 *Benutzerfaktor Nenner* = 9

5. Damit die Fixpunkte auf den Interpolationspunkten liegen, bestimmen Sie in der Registerkarte → *Kurven-Daten* einen ganzzahligen Teiler für die Intervalle. Benutzen Sie dazu den Button → *Einstellen*.
   Eine komplette Zykluslänge des Masters ist 400 mm; dies entspricht 4000 MU.
   Die → *Anzahl Intervalle* = 40 ergibt eine vernünftige Intervallzeit von 25 ms.

6. Definieren Sie → *Fixpunkte* für das Transportband (Master) und die Walze (Slave). Die Funktion → *Ausrichten an Gitter* sollte aktiviert sein.

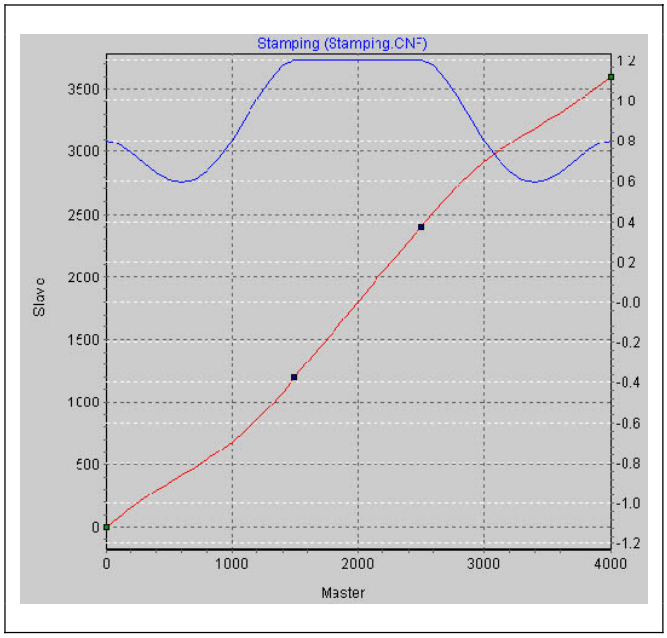
7. Zwischen der Position 1500 und 2500 müssen Master und Slave synchron mit gleicher Geschwindigkeit fahren. Dies erfordert eine Gerade, die mit zwei Tangentenpunkten bestimmt wird.

   Mit einem Doppelklick in der Spalte → *Typ* ändern Sie den Fixpunkt der Position 2500.

   Oder Sie bewegen den Cursor auf den Fixpunkt 2500, klicken auf die rechte Maustaste und wählen im darauf folgenden Kontext-Menü → *Typ ändern*. Da immer zwei Tangentenpunkte benötigt werden, wird der vorhergehende (auf 1500) gleich mit geändert

8. Aktivieren Sie die grafische Darstellung der → ☑ *Geschwindigkeit* um die entsprechende Geschwindigkeitskurve zu sehen:
9. Tragen Sie in der Registerkarte → *Kurven-Info* die → *Zyklen / min Master* = 60 ein. Das ist die Anzahl der Kartons, die (maximal) pro Minute bearbeiten werden.
10. Prüfen Sie, ob die Beschleunigung des Slaves innerhalb des Limits liegt. Aktivieren Sie dazu die Darstellung der → ☑ *Beschleunigung* und des → ☑ *Beschl.Limits.*
11. Um die Kurve in Ihre Steuerung zu laden, müssen Sie zuerst die Datei als CNF-Datei speichern; klicken Sie dazu auf → *Sichern als CNF.*

    In der Titelleiste sehen Sie den Namen der Kurve und die Anzahl der Array-Elemente. Letzteres benötigen Sie für die DIM-Anweisung bei der Programmierung.



12. Laden Sie die CNF-Datei mit den veränderten Parametern und den – automatisch erzeugten – Kurvenarrays mit *Parameter* → *Wiederherstellen aus Datei* in die Steuerung.

□ Programmbeispiel: Kartons mit Haltbarkeitsdatum stempeln

Da die Kurve intern als Array gespeichert wird, muss im Programm als erstes die DIM-Anweisung stehen:

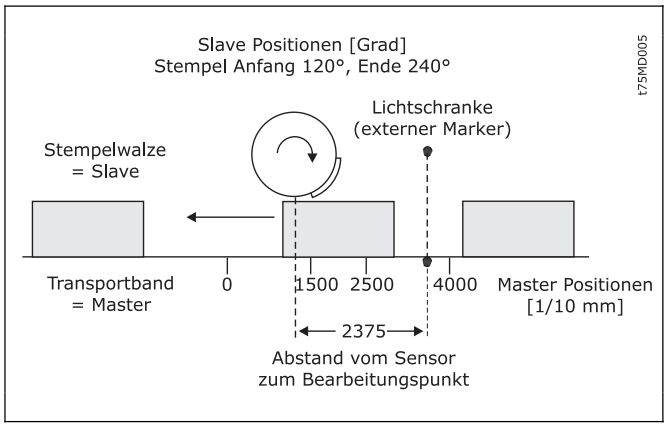
```
DIM stempel[92]     // Anzahl der Elemente aus Titelleiste des CAM-Editors
HOME                // Slave Achse führt eine Homefahrt durch (Schalter für Nullstellung oben)
                    // Danach befindet sich der Slave in der Nullposition (0 Grad)
                    // (entfällt, falls ein Absolutgeber eingesetzt wird)
SETCURVE stempel    // Kurve "stempel" setzen
                    // angenommen ein Karton steht mit Vorderkante am Bearbeitungspunkt
                    // und der Master steht still
DEFMCPOS 1000       // 1000 entspricht dieser Position (Vorderkante Karton)
POSA CURVEPOS       // Slave auf die, der Master-Position entsprechenden Kurvenposition fahren
SYNCC 0             // In den CAM-Mode wechseln und bleiben
SYNCCSTART 0        // Walze sofort mit eingestellter max. Geschwindigkeit einkuppeln
      // dies verursacht keine Bewegung, da Master steht und auf korrekter Position ist
      // jetzt kann der Master gestartet werden
anf:                // leere Hauptschleife, damit Programm nicht beendet wird
                    // hier könnten weitere Verarbeitungen gemacht werden
GOTO anf
```

□ Anwendungsbeispiel: Kartons bedrucken mit Markerkorrektur

In diesem Beispiel werden die Kartons nicht in exakt gleichen Abständen transportiert, daher benötigen Sie Marker, mit denen ein Karton erkannt und die Synchronisation korrigiert werden kann.

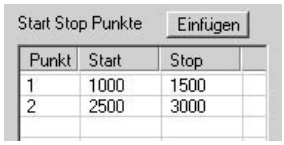
Im Folgenden wird beschrieben, wie Sie die Kurve des vorgehenden Beispiels für diese Anwendung anpassen.

Wieder soll eine Walze auf Kartons eine 10 cm lange Aufschrift stempeln. Auf dem Band werden pro Minute maximal 60 Kartons transportiert. Während des Bedruckens müssen Stempelwalze und Karton synchron laufen.



□ Kurve für die Synchronisation mit Marker editieren

1. Schritte 1 bis 9 wie im vorhergehenden Beispiel.
10. Definieren Sie in der Liste der → *Start-Stop-Punkte* die Punktepaare für das Ein- und Auskuppeln. Am Anfang des Kartons soll eingekuppelt und bis zum Ende des Kartons ausgekuppelt werden.



11. Bestimmen Sie in der Registerkarte → *Kurven-Daten* die Position, in der die Walze stoppen soll, wenn im Programm keine andere Slave-Stop-Position definiert wird:

    Die Walze soll immer auf Position 0 Grad zurückfahren: → *Slave-Stop-Position* = 0

12. Die Lichtschranke (externer Marker) ist 237,5 mm vom Bearbeitungspunkt (= Stempel berührt den Karton) entfernt und erkennt den Anfang des Kartons (entspricht Master-Position 1000). Der Markerabstand beträgt demnach 2375. Tragen Sie diesen Wert in die Registerkarte → *Synchronisation* ein und definieren Sie die erlaubte Toleranz für das Auftreten der Marker und den externen Markertyp = 2 für den Master.

Par. 33-17	*Markerabstand Master*	= 2375
Par. 33-21	*Master-Marker Toleranzfenster*	= 200
Par. 33-19	*Markertyp Master*	= 2

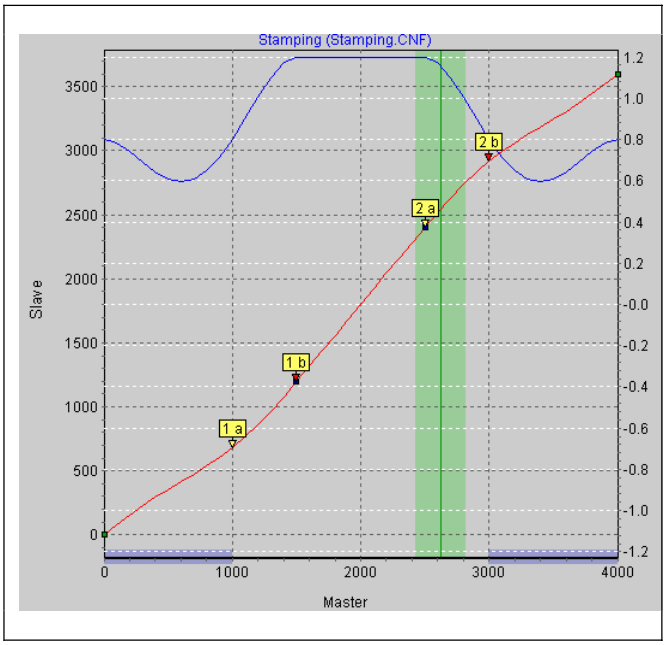
    Tragen Sie die Master-Position in der *Registerkarte* → *Kurven-Daten* ein:

Master-Marker-Position	= 100

13. Für die Festlegung, wann die Korrektur der Synchronisation frühestens beginnen kann und wann sie beendet sein muss, betrachten Sie das Kurvenprofil. Die grüne senkrechte Linie zeigt, an welcher Master-Position der Marker erkannt wird, der hellgrüne Bereich zeigt das Toleranz-fenster für das Auftreten des Master-Markers.

    Die Korrektur darf frühestens beginnen, wenn ein Karton fertig bedruckt ist, denn jede Änderung der Geschwindigkeit während des Bedruckens würde den Karton beschädigen. Und die Korrektur muss vollständig beendet sein, wenn der nächste Karton den Bearbei-tungspunkt erreicht. In diesem Beispiel sind die Master-Positionen Ende und Anfang eines Kartons gut geeignet:

    Korrektur Start = 3000
    Korrektur Ende = 1000



    Tragen Sie die Werte in die Registerkarte → *Kurven-Daten* ein; der Bereich wird im Kurvenprofil blau schraffiert gezeigt.

14. Prüfen Sie, ob Geschwindigkeit und Beschleunigung des Slaves innerhalb des Limits bleiben. Aktivieren Sie dazu die Darstellung der → ☑ *Geschwindigkeit* und des → ☑ *Geschw.Limits* und danach die Darstellung der → ☑ *Beschleunigung* und des → ☑ *Beschl.Limits.*
15. Klicken Sie auf → *Speichern als CNF* um die Datei zu speichern, zum Beispiel „marker".
16. Laden Sie die CNF-Datei mit den veränderten Parametern und den – automatisch erzeugten – Kurvenarrays mit Parameter → *Wiederherstellen aus Datei* in die Steuerung.

□ Programmbeispiel: Kartons bedrucken mit Markerkorrektur

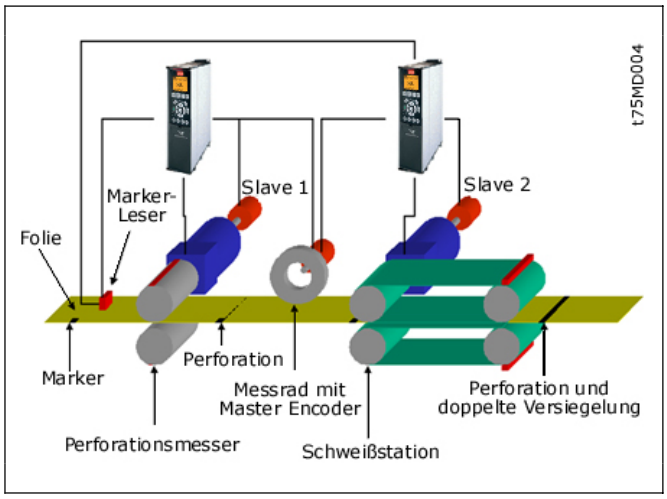
Da die Kurve intern als Array gespeichert wird, muss in Ihrem Programm als erstes die DIM-Anweisung stehen:

```
DIM marker[112]     // Anzahl der Elemente aus Titelleiste des CAM-Editors
HOME          // Slave Achse führt eine Homefahrt durch (Schalter für Nullstellung oben)
              // Danach befindet sich der Slave in der Nullposition (0 Grad)
              // (entfällt, falls ein Absolutgeber eingesetzt wird)
SETCURVE marker    // Stempelkurve mit Marker setzen
dist = GET SYNCMPULSM          // Abstand zum Sensor
DEFMCPOS (1000-dist)           // Das ist die Stelle, die dem Sensorsignal entspricht
SET SYNCMSTART 2000            // Zählen des Masterpulses beginnt erst
                               // wenn nächste Flanke von Sensor kommt
SYNCCMM 0          // Im CAM-Mode synchronisieren bis Motor Stopp
SYNCCSTART 1       // Walze mit Start-Punktepaar 1 einkuppeln
      // Synchronbetrieb
WAITI 4 ON         // Warten auf Eingangssignal, wenn Transportband abgeschaltet wird
SYNCCSTOP 2 0      // Walze mit Stopp-Punktepaar 1 auskuppeln und bei Position 0 Grad an
```

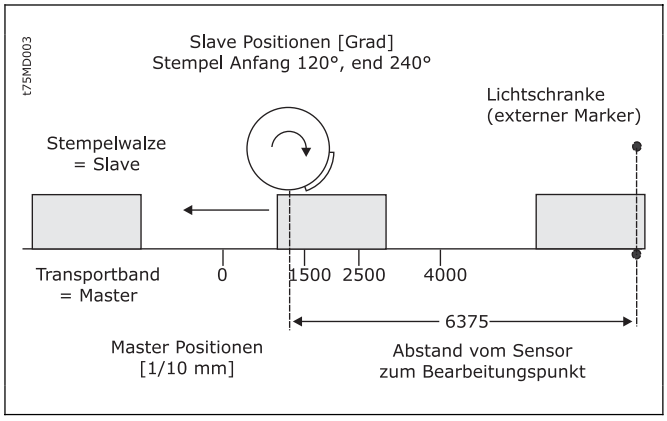
□ Wenn der Abstand des Sensors größer als eine Masterzykluslänge ist

Bei vielen Anwendungen kann der Marker nicht innerhalb einer Masterzykluslänge angebracht werden, z.B. bei folgender Maschine zur Produktion von Plastiktüten:

Da hier zwischen den Slaves keine Marker eingebaut werden können, gibt es in dieser Anwendung nur einen Markerleser, die Schweißstation liegt aber viel weiter als eine Masterzykluslänge entfernt. Da der Abstand des Sensors größer als eine Masterzykluslänge ist, wird ein Puffer für die Markerabweichung angelegt. Bei Erscheinen des Markers wird der Wert in den Puffer geschrieben und mit Erscheinen des nächsten Markers ausgelesen.



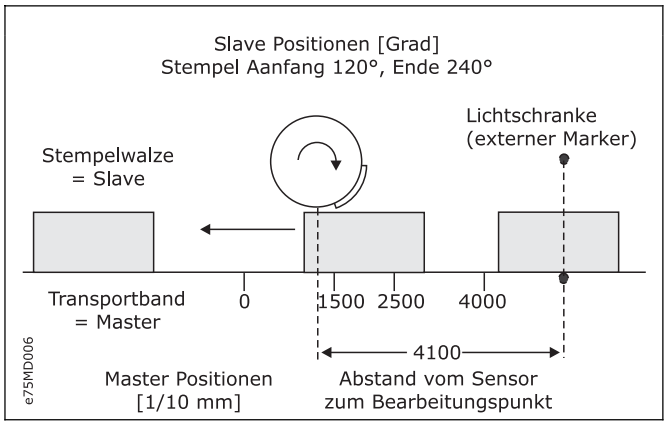
Um zu beurteilen, in welchem Bereich korrigiert werden darf, subtrahieren Sie so oft die Masterzykluslänge, bis der Wert < 1 Masterzykluslänge ist. Dies ist der maximal erlaubte Abstand zum Korrigieren. In diesem Beispiel ist dieser also 6375 – 4000 = 2375 und damit der gleiche Korrekturbereich wie im vorangegangen Beispiel.



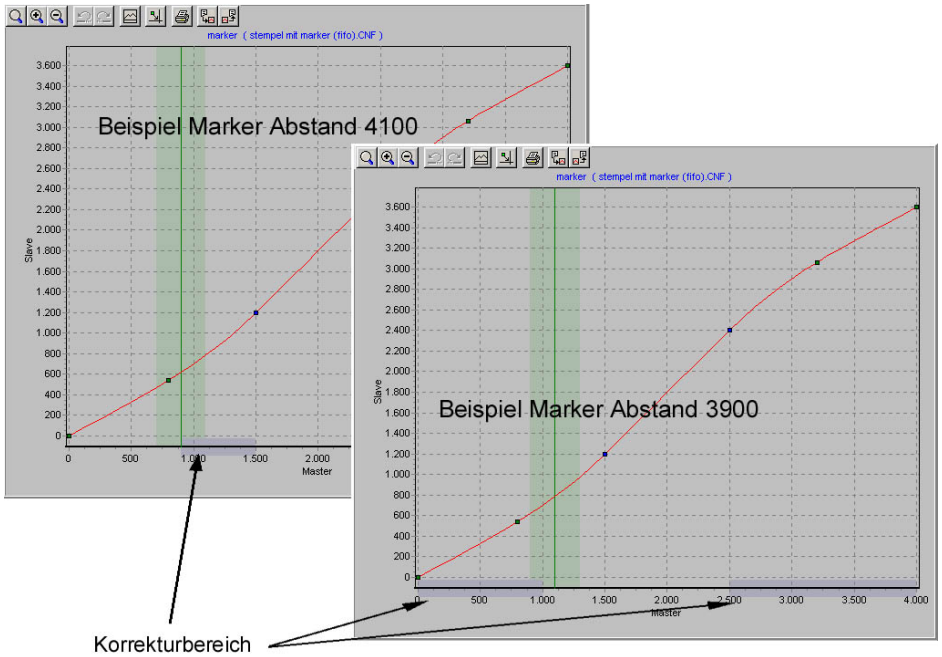
□ Problemfälle bei der Festlegung des Markerabstandes

Wenn der Marker so nah am Bearbeitungspunkt angebracht ist, dass nach Erkennen des Markers keine Zeit bleibt, die Synchronisation zu korrigieren, können Sie das Problem nur durch eine mechanische Veränderung des Markers beheben.

Der gleiche Effekt könnte aber auch auftreten, wenn der Markerabstand größer als die Masterzykluslänge ist und nach Subtraktion dieses Wertes ebenfalls ein zu geringer Abstand bleibt, zum Beispiel:



Bei Erscheinen des Markers wird der Wert in den Puffer geschrieben. Erst wenn der nächste Marker erkannt wird, wird der Puffer ausgelesen. Das bedeutet, dass der Marker erst bei der Master-Position 900 „erkannt" wird und in unserem Beispiel nur noch wenig Zeit bleibt, den Fehler zu korrigieren. Es ist der gleiche Effekt, als wäre der Sensor um den Wert (Abstand – Mastertaktlänge) bzw. (4100 – 4000), also nur 10 mm vor dem Bearbeitungspunkt montiert.



Daher wäre es besser, den Sensor so zu montieren, dass der Abstand zum Bearbeitungspunkt entweder kleiner oder wesentlich größer als eine Masterzykluslänge ist, hier zum Beispiel im Abstand von 3900. Dann kann man von 2500 bis 1000 korrigieren.

Oder man montiert den Sensor weiter weg, zum Beispiel im Abstand von 7900. Dies wirkt genau so, als wäre der Sensor um Abstand – Masterzykluslänge (7900 – 4000), also 3900 vor dem Bearbeitungspunkt montiert. Genügend Zeit also, um die Synchronisation zu korrigieren.

Falls dies mechanisch nicht möglich ist, muss man die Werte etwas manipulieren, damit man die Lösung mit dem Puffer vermeiden kann. Gehen Sie folgendermaßen vor:

Subtrahieren Sie vom tatsächlichen Abstand einen Wert x, damit der Abstand < Masterzykluslänge wird, zum Beispiel 4100 – 200 = 3900. Den Wert x subtrahieren Sie auch von der Master-Position, also 1000 – 200 = 800.

Tragen Sie beide Werte in die Registerkarten → *Synchronisation* und → *Kurven-Daten* ein:

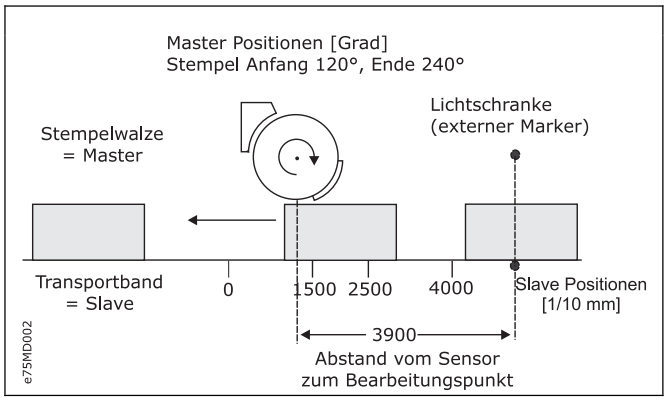
Par. 33-17 *Markerabstand Master*	= 3900
Master-Marker-Position	= 800

Da nun kein Puffer erzeugt wird, könnte man zum Beispiel von 2500 bis 800 korrigieren.

◻ Anwendungsbeispiel: Slave-Synchronisation mit Marker

In folgendem Beispiel ist das Transportband der Slave und die Stempelwalze der Master, da für einen gleichmäßig Druck die Farbaufnahme und Farbabgabe kontinuierlich ablaufen müssen. Pro Minute werden maximal 20 Kartons auf dem Band transportiert. Der Abstand der Kartons ist nicht größer als eine Masterzykluslänge. Während des Bedruckens müssen Stempelwalze und Karton synchron laufen.

Im Gegensatz zur Synchronisation mit Marker-korrektur des Masters wird hier die Slave-Position korrigiert und nicht die Kurve.



□ Kurve für Slave-Synchronisation editieren

1. FC 300 mit den erforderlichen Parameter einstellen und diese Benutzerparameter mit *Parameter → speichern in Datei* mit der Extension „CNF" sichern.
2. Diese CNF-Datei muss im *CAM-Editor* geöffnet sein.
3. Ermitteln Sie den Getriebefaktor des Masters in MU-Einheiten:

   Getriebefaktor = 5/1

   Drehgeberauflösung (Inkrementalgeber) = 500

   Eine Umdrehung der Walze ist 360 Grad. Es soll mit einer Auflösung von 1/10 Grad gearbeitet werden. Das bedeutet, dass eine Umdrehung der Walze in 3600 Arbeitseinheiten eingeteilt wird:
   Skalierfaktor = 3600

   $$\frac{\text{Getriebefaktor} * \text{Drehgeberauflösung} * 4}{\text{Skalierfaktor}} \text{qc} = 1 \text{ MU}$$

   Geben Sie diese ganzzahligen Wert in der Registerkarte → *Synchronisation* ein:

   Par. 33-10 *Syncfaktor Master* = 25
   Par. 33-11 *Syncfaktor Slave* = 9

4. Getriebefaktor des Slaves in Benutzereinheiten BE eingeben: Die Eingabe soll in 1/10 mm Auflösung möglich sein.

   Der Antrieb ist mit dem Transportband mit einer Getriebeübersetzung von 25:11 verbunden; das heißt der Motor macht 25, das Zahnriemenrad 11 Umdrehungen.
   Getriebefaktor = 25/11

   Inkrementalgeber direkt am Master-Antrieb; Drehgeberauflösung = 4096

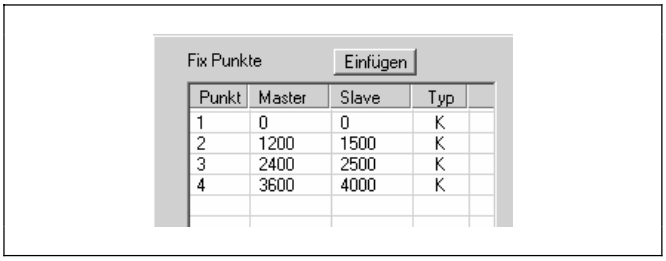
   Das Zahnriemenrad hat 20 Zähne/Umdrehung, 2 Zähne entsprechen 10 mm, daher entspricht 1 Umdrehung = 100 mm Transport. Der Skalierfaktor ist demnach 1000.

   $$\frac{\text{Getriebefaktor} * \text{Drehgeberauflösung} * 4}{\text{Skalierfaktor}} \text{qc} = 1 \text{ BE}$$

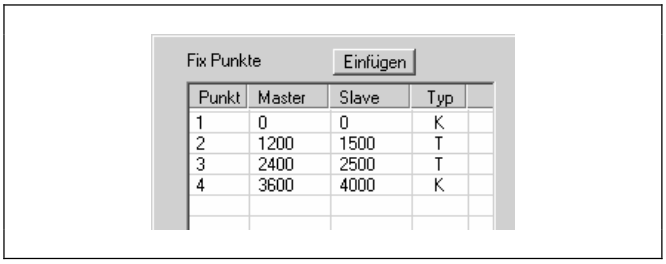
   Geben Sie diese Werte in der Registerkarte → *Encoder* ein:

   Par. 32-12 *Benutzerfaktor Zähler* = 2048
   Par. 32-11 *Benutzerfaktor Nenner* = 55

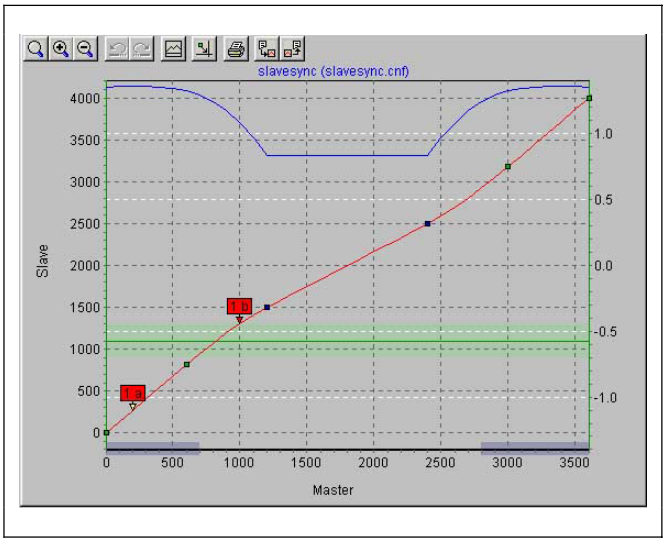
5. Damit die Fixpunkte auf den Interpolationspunkten liegen, bestimmen Sie in der Registerkarte → *Kurven-Daten* einen ganzzahligen Teiler für die Intervalle. Für eine komplette Zykluslänge des Masters von 3600 (= 360 Grad) ergibt die → *Anzahl Intervalle* = 36 eine vernünftige Intervallzeit von 27,7 ms. Geben Sie diese Werte in der Registerkarte → *Kurven-Daten* mit dem Button *Einstellen* ein.
6. Definieren Sie → *Fixpunkte* für die Walze (Slave) und das Transportband (Master). Die Funktion → *Ausrichten an Gitter* sollte aktiviert sein.



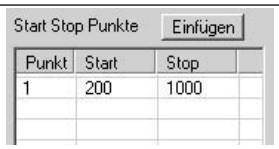
7. Zwischen den Master-Positionen 1200 bis 2400 müssen Master und Slave synchron mit gleicher Geschwindigkeit fahren. Dafür benötigen Sie eine Gerade, die mit zwei Tangentenpunkten bestimmt wird. Mit einem Doppelklick in der Spalte → *Typ* definieren Sie für die Position 2400 einen Tangentenpunkt; der davor liegende wird automatisch angepasst.



Aktivieren Sie die grafische Darstellung der → ☑ *Geschwindigkeit* um den Verlauf zu sehen.



8. Tragen Sie in der Registerkarte → *Kurven-Info* die → *Zyklen / min Master* = 20 ein. Das ist die Anzahl der Kartons, die (maximal) pro Minute bearbeiten werden.
9. Prüfen Sie, ob die Beschleunigung des Slaves innerhalb des Limits liegt. Aktivieren Sie dazu die Darstellung der → ☑ *Beschleunigung* und des → ☑ *Beschl. Limits*.



10. Definieren Sie in der Liste → *Start-Stop-Punkte* um die Synchronisation am Anfang zu starten. Zwischen 20 und 100 Grad soll mit etwas Sicherheitsabstand eingekuppelt werden, denn bei 120 Grad muss aufsynchronisiert sein.
11. Bestimmen Sie in der Registerkarte → *Kurven-Daten* die Position, in der das Transportband stoppen soll, wenn im Programm keine andere *Slave-Stop-Position* definiert wird:

    Das Transportband soll immer auf Position 0 halten: → *Slave-Stop-Position* = 0
12. Die Lichtschranke (externer Marker) ist 390 mm vom Bearbeitungspunkt (= Stempel berührt den Karton) entfernt und erkennt den Anfang des Kartons (entspricht Slave-Position 1000). Der Markerabstand beträgt demnach 3900. Tragen Sie diesen Wert in die Registerkarte → *Synchronisation* ein und definieren Sie die erlaubte Toleranz für das Auftreten der Marker und den externen *Markertyp* = 2 für den Slave:

Par. 33-18	*Markerabstand Slave*	= 3900
Par. 33-22	*Slave-Marker Toleranzfenster*	= 200
Par. 33-20	*Markertyp Slave*	= 2

    Tragen Sie die Slave-Position in der Registerkarte → *Kurven-Daten* ein:

    Slave Marker-Position = 1000
13. Für die Festlegung, wann die Korrektur der Synchronisation frühestens beginnen kann und wann sie beendet sein muss, betrachten Sie das Kurvenprofil. Die grüne waagrechte Linie zeigt, an welcher Master-Position der Marker erkannt wird, der hellgrüne Bereich zeigt das Toleranzfenster für das Auftreten des Master-Markers.

    Die Korrektur darf frühestens beginnen, wenn ein Karton fertig bedruckt ist, denn jede Änderung der Geschwindigkeit während des Bedruckens würde den Druckstempel und/oder den Karton beschädigen. Und die Korrektur muss vollständig beendet sein, wenn der nächste Karton den Bearbeitungspunkt erreicht. In diesem Beispiel sind die Slave-Positionen Ende und Anfang eines Kartons gut geeignet. Tragen Sie die Werte in die Registerkarte → *Kurven-Daten* ein:

    Korrektur Start = 2800
    Korrektur Ende = 750
14. Prüfen Sie, ob die Geschwindigkeit und Beschleunigung des Slaves innerhalb des Limits bleiben. Aktivieren Sie dazu die Darstellung der → ☑ *Geschwindigkeit* und des → ☑ *Geschw. Limits* und danach die Darstellung der → ☑ *Beschleunigung* und des → ☑ *Beschl. Limits.*
15. Klicken Sie auf den Button → *Sichern als CNF* zum speichern.
16. Laden Sie die CNF-Datei mit den veränderten Parametern und den – automatisch erzeugten – Kurvenarrays mit *Parameter* → *Wiederherstellen aus Datei* in den FC 300.

□ Programmbeispiel: Slave-Synchronisation mit Marker

Um die Master-Position zu bestimmen wird ein Schalter am Master vorausgesetzt, der die Nullposition signalisiert. Um den Slave in die richtige Position zu fahren, wird dieser bis zur Lichtschranke vorwärts gefahren. Dies entspricht dem Kartonanfang = 1000. Dann fährt man den Slave um 2900 (= Markerabstand 3900–1000) weiter; damit steht der Slave mit dem Kartonanfang 1000 genau vor dem Bearbeitungspunkt, also an Slave-Position 0.

```
DIM slavesync[108]          // Anzahl der Elemente aus Titelleiste des CAM-Editors
HOME          // Slave führt eine Homefahrt durch (Schalter für Nullstellung oben)
              // Danach befindet sich der Slave in der Nullposition (0 Grad)
              // (entfällt bei einem Absolutdrehgeber)
DEFMCPOS 0                  // Kurve beginnt bei Master-Position 0
SET SYNCMSTART 2000         // Zählen des Masterpulses beginnt erst
                            // wenn nächste Flanke vom Sensor kommt
SETCURVE slavesync          // Kurve für die Slave-Synchronisation setzen
              // zum Start fahren
CSTART
CVEL 10                     // langsam vorwärts fahren bis Lichtschranke kommt
oldi = IPOS                 // oldi = letzte Markerposition des Slaves
WHILE (oldi == IPOS) DO     // Warten bis Karton erkannt
ENDWHILE
POSA (IPOS + 2900)          // Karton um 2900 nach vorne fahren
SYNCCMS 0                   // Im CAM-Modus synchronisieren
SYNCCSTART 1                // Mit Start-Stop-Punktepaar 1 einkuppeln
```

□ Nockenschaltwerk

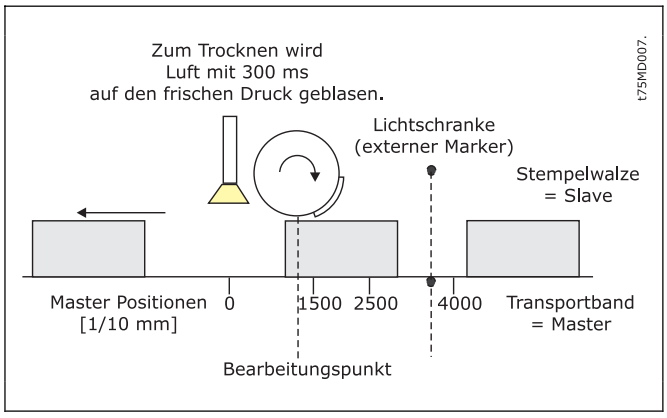
Die mechanische Nockenwelle wird ebenfalls durch eine (oder mehrere) Kurven nachgebildet. Um ein Nockenschaltwerk zu realisieren, muss es möglich sein, den Slave immer wieder an bestimmten Master-Positionen ein- und auszukuppeln.

Dies ist mit APOSS mit den Interrupt-Befehlen ON MAPOS .. GOSUB und ON APOS .. GOSUB möglich. Man kann immer dann ein Unterprogramm aufrufen, wenn eine definierte Master-Position (und zwar in positiver oder negativer Richtung) passiert wurde.

In Verbindung mit einem Kurvenprofil, in dem mehrere Start-Stop-Punktepaare zum Aus- und Einkuppeln definiert wurden, kann man viele Anwendungen wie sie in der Verpackungsindustrie typisch sind realisieren.

□ Programmbeispiel für ein Nockenschaltwerk

Nach dem Bedrucken eines Kartons soll der frische Druck sofort im Luftstrom getrocknet werden:

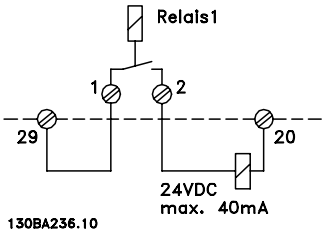


```
ON MCPOS 2500 GOSUB trocknen          // Unterprogramm aufrufen, wenn die
                    // Master-Position 2500 in positiver Richtung  passiert wurde
SUBMAINPROG
    SUBPROG trocknen
    OUT 1 1                         // Trockner einschalten
    DELAY 300                       // 300 ms trocknen
    OUT 1 0                         // Trockner ausschalten
RETURN
ENDPROG
```

□ Mechanische Bremssteuerung

In Anwendungen, die durch MCO 305 gesteuert werden und über eine elektromechanische Bremse verfügen, macht es normalerweise Sinn, die Bremse vom MCO 305 Anwendungsprogramm zu steuern, um zu vermeiden, dass die Positioniersteuerung versucht den Motor zu bewegen, während die Bremse noch eingekuppelt ist.

Die Bremssteuerung im MCO 305 Anwendungsprogramm kann mit der mechanischen Bremssteuerung des FC 300 kombiniert werden. Dazu schaltet man zwei Ausgänge in Serie: z.B. durch Setzen des digitalen Ausgangs 29 auf *Mechanische Bremssteuerung* (Par. 5-31) und des Relaisausgangs 1 auf *MCO gesteuert* (Par. 5-40 [0]). Die Bremse wird dann wie in der Abbildung gezeigt verbunden.



□ Programmbeispiel: Relative Positionierung mit einer mechanischen Bremssteuerung

```
/*********************************************************************/
      Eingänge:     1       Positionieren
                    8       Fehler löschen
      Ausgänge:     1       in Position
                    8       Fehler
                    11      Relaisausgang für mechanische Bremse
/************************** Interrupts **********************************/
ON ERROR GOSUB errhandle   // Bei Fehler in die Fehlerroutine springen; diese muss immer enthalten sein.
/************************  Flags definieren  ******************************/
flag = 0
/************************  Grundeinstellungen  ****************************/
VEL 80          // Positioniergeschwindigkeit bezogen auf Par. 32-80 Maximalgeschwindigkeit setzen.
ACC 100         // Positionierbeschleunigung bezogen auf Par. 32-81 kürzeste Rampe setzen.
DEC 100         // Positionierverzögerung bezogen auf Par. 32-81 kürzeste Rampe setzen.
/*******************  Anwendungsparameter definieren  ********************/
LINKGPAR 1900 "Box Höhe" 0 1073741823 0
LINKGPAR 1901 "Verzögerung der Bremse beim Schließen" 0 1000 0
LINKGPAR 1902 "Verzögerung der Bremse beim Öffnen" 0 1000 0
/*******************  Betriebssicheren Antrieb initialisieren  ************/
GOSUB engage       // Sicherstellen, dass die mechanische Bremse nach dem Einschalten geschlossen ist.
/************************  Hauptprogrammschleife  *************************/
MAIN:
IF (IN 1 == 1) AND (flag == 0) THEN
   // Einmal positionieren (abgesichert durch Flag) wenn Eingang 1 high.
   GOSUB disengage              // Mechanische Bremse vor dem Starten öffnen.
   OUT 1 0                      // Reset "in Position" Ausgang.
   POSR (GET 1900)              // Positionieren
```

```
   OUT 1 1                          // "in Position" Ausgang setzen.
   flag = 1            // "flag" setzen, um sicherzustellen, die Distanz nur einmal durchfahren wird.
ELSEIF (IN 1 == 0) AND (flag == 1) THEN       // Einmal anhalten, wenn Eingang 1 low.
   MOTOR STOP                  // Anhalten wenn Eingang low.
   flag = 0                    // Reset "flag" um eine neue Positionierung freizugeben.
   GOSUB engage                // Mechanische Bremse nach dem Anhalten schließen.
ENDIF
GOTO MAIN
/*********************  Unterprogramm starten  ***************************/
SUBMAINPROG
/*****************  Mechanische Bremse einkuppeln  ************************/
SUBPROG engage
   OUT 11 0                    // Mechanische Bremse schließen.
   DELAY (GET 1901)
   // Warten, um sicherzustellen, dass die Bremse eingekuppelt ist, bevor der Motor freigegeben wird.
   MOTOR OFF                   // Positioniersteuerung anhalten und Motor in Leerlauf.
RETURN
/***************** Mechanische Bremse auskuppeln ***************************/
SUBPROG disengage
   MOTOR ON                    // Antrieb freigeben und Positioniersteuerung starten.
   DELAY (GET 1902)
   // Warten, um sicherzustellen, dass der Motor bestromt ist, bevor die Bremse geöffnet wird.
   OUT 11 1                    // Mechanische Bremse öffnen.
RETURN
/**********************  Fehlerbehandlung  ********************************/
SUBPROG errhandle
   OUT 11 0  // Bremse bei Auftreten eines Fehlers schließen.
   err = 1    // Fehler-Flag setzen, um solange in der Fehlerroutine zu bleiben, bis der Fehler gelöscht ist.
   OUT 8 1                  // Ausgang Fehler setzen.
   WHILE err DO             // In der Fehlerroutine bleiben, bis die Reset-Meldung empfangen ist.
      IF IN 8 THEN          // Fehlermeldung zurücksetzen wenn Eingang 8 high.
         ERRCLR             // Fehler löschen.
         err=0              // Fehler-Flag zurücksetzen.
      ENDIF
   ENDWHILE
   OUT 8 0                  // Ausgang Fehler zurücksetzen.
   flag = 0                 // "Flag" zurücksetzen, um neue Positionierung freizugeben.
RETURN
/*************************************************************************/
ENDPROG
/************************** Programmende ********************************/
```

□ Ruckbegrenzung

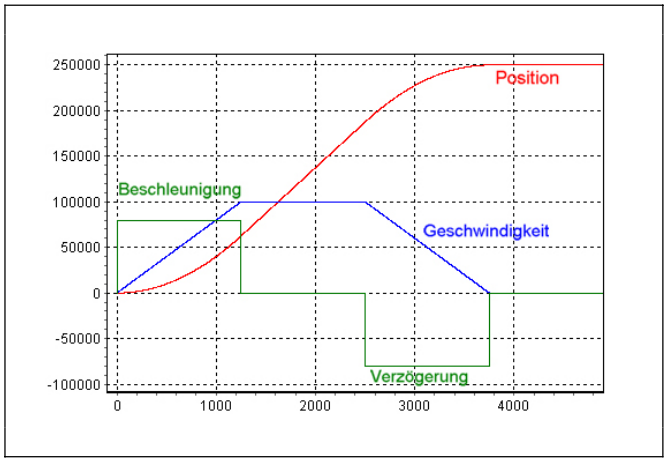
□ Wie ruckbegrenzte Bewegungen funktionieren

Ruckbegrenzte Bewegungen sind ähnlich den normalen trapezförmigen Bewegungen, außer dass der Anwender die „Sanftheit" der Beschleunigung und Verzögerung steuern kann. Dadurch kann den Ruck, der durch eine unmittelbare Beschleunigung einer trapezförmigen Bewegung verursacht wird, begrenzt werden.

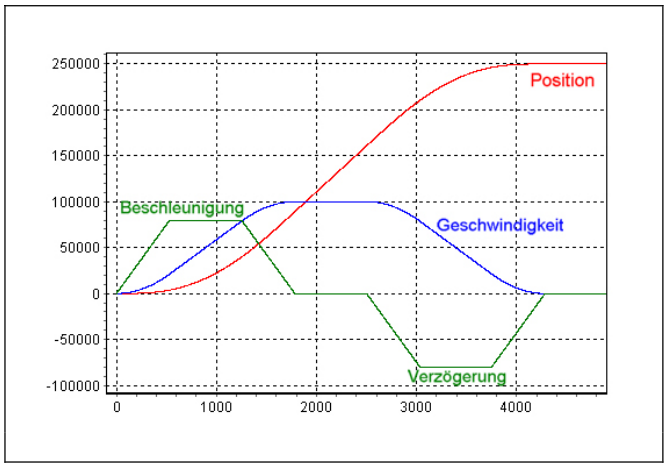
Typische Anwendungen, die ruckfreie Bewegungen erfordern, sind:

- Fahrstuhl
- Bewegung von schweren Lasten

Beispielhaft zeigt das nebenstehende Diagramm die Beschleunigungs-, Geschwindigkeits- und Positionskurve einer trapezförmigen Bewegung von einer Position zur anderen. Die scharfen Wechsel der Beschleunigung zwingen den Motor zu einem Ruck am Anfang und am Ende jeder Geschwindigkeitsrampe.



Das Diagram zeigt die gleiche Bewegung mit einer Ruckbegrenzung. Beachten Sie, dass nun die Beschleunigung nicht mehr unmittelbar ausgeführt wird und dass die „Ecken" der Geschwindigkeitskurve abgerundet sind. Dies resultiert in eine sanftere Motorbewegung. Es dauert außerdem etwas länger, die Zielposition zu erreichen, weil der Motor länger braucht um auf die maximale Beschleunigung zu beschleunigen.



Um die „Sanftheit" der Beschleunigungsrampe zu steuern, stehen 4 Parameter zur Verfügung:

Parameter Ruckdauer

JERKMIN: Konstante Beschleunigungsrampe beim Anfahren. Dies definiert die Zeitspanne in Millisekunden, die beim Anfahren notwendig ist, um von 0 die maximale Beschleunigung zu erreichen.

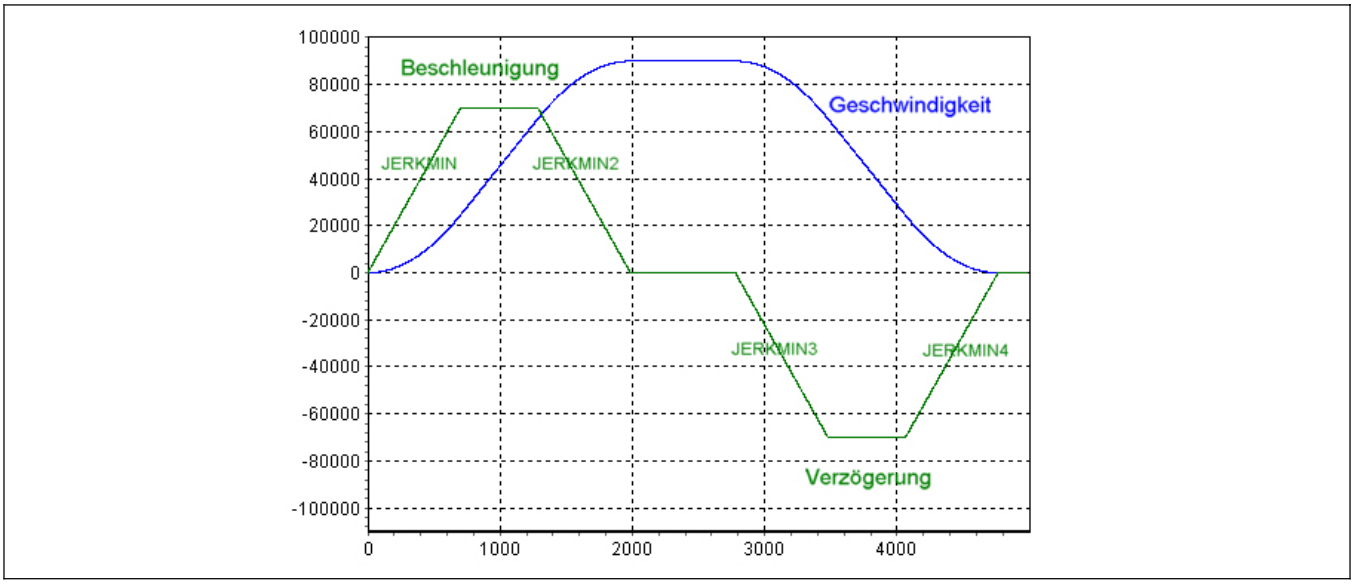
JERKMIN2: Konstante Rücknahme der Beschleunigung. Dies definiert die Zeitspanne [ms] in der die maximale Beschleunigung auf 0 Beschleunigung reduziert werden soll (d.h. normalerweise auf konstante maximale Geschwindigkeit). Wenn „0" gesetzt ist, wird der gleiche Wert wie bei JERKMIN verwendet.

JERKMIN3: Konstante Verzögerungsrampe beim Anhalten. Dies definiert die Zeitspanne [ms], die notwendig ist, um von 0 die maximale Verzögerung zu erreichen. Wenn „0" gesetzt ist, wird der gleiche Wert wie bei JERKMIN verwendet.

JERKMIN4: Konstante Zunahme der Verzögerung. Dies definiert die Zeitspanne [ms], die notwendig ist, um von der maximalen Verzögerung auf 0 zu kommen (das ist normalerweise die Geschwindigkeit 0). Wenn „0" gesetzt ist, wird der gleiche Wert wie bei JERKMIN verwendet.

Diese Konstanten entsprechen der „Steigung" in den verschieden Teilen der Beschleunigungskurve (siehe folgendes Diagramm). Je größer die Werte, desto sanfter wird beschleunigt und/oder gebremst, in gleichem Maße werden die Rampen immer länger.

ACHTUNG!:
Die durch die *Ruckdauer* JERKMIN definierte Beschleunigungssteigung wird immer benutzt, wenn beim Anfahren beschleunigt wird und nicht nur, wenn von 0 auf die maximale Beschleunigung beschleunigt wird. Das gleiche gilt sinngemäß für die drei anderen Parameter auch: JERKMIN2 wird immer benutzt, wenn die Beschleunigung zurückgenommen wird, usw.



Ruckbegrenzte Bewegungen erreichen normalerweise nicht die Geschwindigkeits- und Beschleunigungsgrenzen, die für die Steuerung gesetzt sind (z.B. Begrenzungen durch die Befehle VEL, ACC, DEC, etc.). Im Diagramm oben sieht man diese Begrenzung an den „Plateaus" in der Beschleunigungskurve. Falls die aktuelle Geschwindigkeit und/oder Beschleunigung außerhalb dieser Grenzen ist, wenn die ruckbegrenzte Bewegung startet, wird die Bewegung entsprechend beschleunigt oder verzögert, um sie innerhalb dieser gesetzten Grenzen zu bringen.

Es ist wichtig zu verstehen, dass für ruckbegrenzte Bewegungen die „Beschleunigung" als „Anfahren" in jede Richtung definiert ist (d.h. entweder vorwärts oder rückwärts) und ebenso die „Verzögerung" als „Abbremsen" in jede Richtung. Das Ergebnis davon ist, dass maximale Geschwindigkeit, maximale Beschleunigung, maximale Verzögerung und die vier *Ruckdauer*-Werte alle unabhängig von der Bewegungsrichtung sind. Dies kann wichtige Konsequenzen haben, wenn eine ruckbegrenzte Bewegung die Motorrichtung ändern muss, besonders wenn sich die maximale Verzögerung von der maximalen Beschleunigung unterscheidet. In diesem Fall garantiert die ruckbegrenzte Bewegung, dass die Verzögerungsrampe bei exakt Geschwindigkeit 0 sanft in eine Beschleunigungsrampe mündet, wenn die Richtung wechselt und ohne weder die Verzögerungs- noch die Beschleunigungsgrenzen zu erreichen.

Eine ruckbegrenzte Bewegung kann in drei verschiedenen Situationen benutzt werden:

1. Anhalten aus der aktuellen Geschwindigkeit und Beschleunigung (wobei die endgültige Position nicht wichtig ist).
2. Wechsel von der aktuellen Geschwindigkeit und Beschleunigung in eine definierte konstante Geschwindigkeit (wobei die Positionen nicht wichtig sind).
3. Fahren von der aktuellen Position (und der aktuellen Geschwindigkeit und Beschleunigung) und Anhalten auf einer definierten Position.

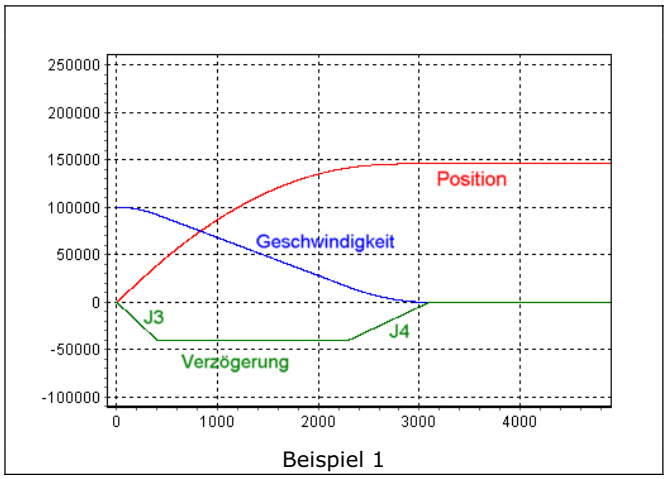
☐ Beispiele

In den folgenden Beispielen ist die maximale Beschleunigung auf einen höheren Wert als die maximale Verzögerung gesetzt, so dass der Motor schneller anlaufen als bremsen kann. Ebenso ist JERKMIN kleiner gesetzt als JERKMIN2, JERKMIN2 kleiner als JERKMIN3 und JERKMIN3 kleiner als JERKMIN4, damit die verschiedenen Kurvensegmente im Diagramm besser zu unterscheiden sind. Die vier JERKMIN Werte sind mit J1, J2, J3 und J4 gekennzeichnet.

Anhalten

Das nebenstehende Diagramm zeigt eine Stopp-Bewegung, die mit einer positiven konstanten Geschwindigkeit beginnt.

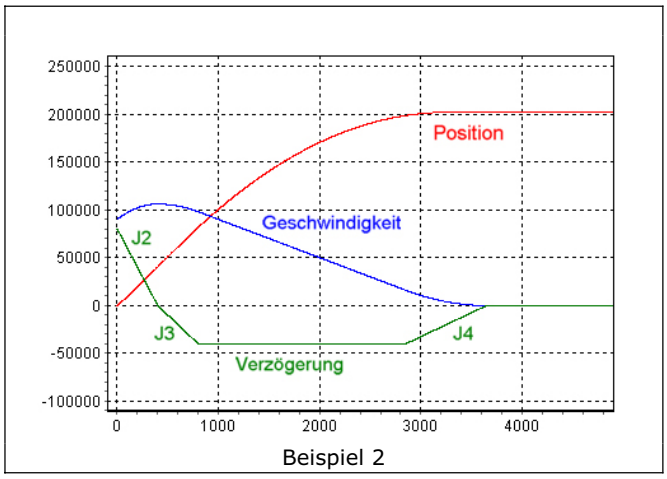
Die Kurve besteht aus einem Segment Verzögerungsrampe (JERKMIN3), gefolgt von einem Segment konstanter Verzögerung (bei maximaler Verzögerung) und schließlich einem Segment Zunahme der Verzögerung auf Geschwindigkeit 0 (JERKMIN4).



Beispiel 1

Dieses Diagramm zeigt eine Stopp-Bewegung, die mit positiver Geschwindigkeit und positiver Beschleunigung beginnt.

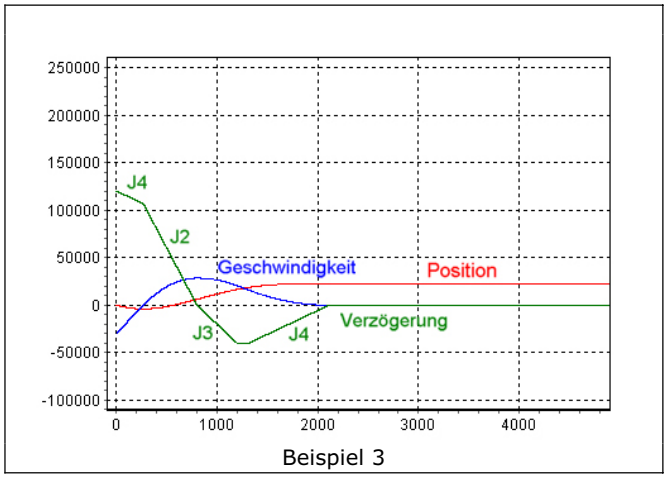
Da die anfängliche Beschleunigung positiv ist, muss die Kurve mit einer Verzögerungsrücknahme auf Beschleunigung 0 beginnen (JERKMIN2). Es folgt dann ein Segment Verzögerungsrampe beim Anhalten (JERKMIN3), ein Segment konstante Verzögerung und ein Segment Zunahme der Verzögerung auf Geschwindigkeit 0 (JERKMIN4).



Beispiel 2

Das folgende Diagramm zeigt ein Stopp-Bewegung, die mit einer negativen Geschwindigkeit und einer sehr hohen Verzögerung beginnt. (Es ist eine Verzögerung, weil die Geschwindigkeit abnimmt.) Da jedoch die anfängliche Verzögerung so groß ist, ist der Motor nicht in der Lage ohne Überschwingen (über Geschwindigkeit 0 und zurück) zu stoppen.

Daher startet die Kurve mit einer Zunahme der Verzögerung (JERKMIN4) um die Verzögerung so stark wie möglich zu verlangsamen, bevor die Geschwindigkeit 0 erreicht wird. Bei Geschwindigkeit 0 wird aus der „Verzögerung" eine „Beschleunigung", weil sich die Richtung geändert hat. Demzufolge wird die Kurve mit einer Rücknahme der Beschleunigung fortgesetzt (JERKMIN2) bis Beschleunigung 0 erreicht ist. Der Motor fährt nun mit einer konstanten positiven Geschwindigkeit und daher wird die Kurve ganz normal mit einer Verzögerungsrampe (JERKMIN3), einem Segment konstanter Verzögerung (sehr kurz in diesem Beispiel) und einer Zunahme der Verzögerung auf Geschwindigkeit 0 (JERKMIN4) beendet.

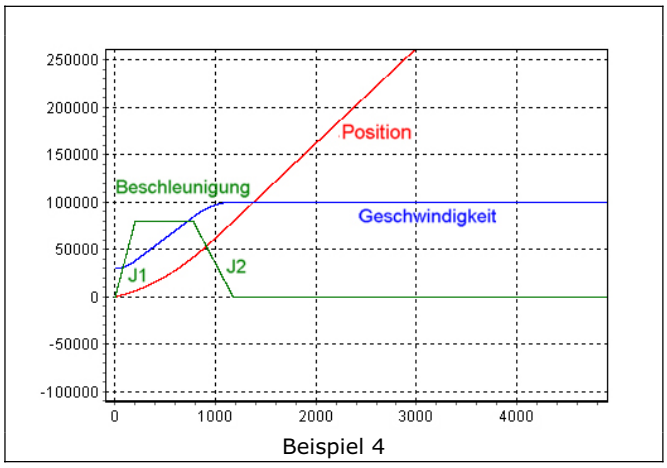


Beispiel 3

In eine konstante Geschwindigkeit wechseln

Dieses Diagramm zeigt eine Bewegung, die mit einer positiven konstanten Geschwindigkeit beginnt und diese auf eine höhere positive konstante Geschwindigkeit steigert.

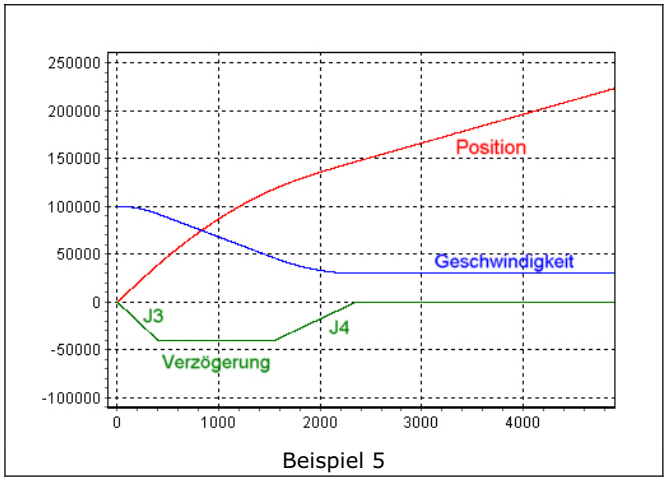
Diese Kurve besteht aus einem Segment Beschleunigungsrampe (JERKMIN), gefolgt von einem Segment konstanter Beschleunigung (bei maximaler Beschleunigung) und schließlich einer Rücknahme der Beschleunigung auf konstante Geschwindigkeit (JERKMIN2). Beachten Sie, dass die Verzögerungswerte JERKMIN3 und JERKMIN4 nicht benutzt werden, weil es nie eine Verzögerung gibt.



Beispiel 4

Dieses Diagramm zeigt eine Bewegung, die mit einer hohen positiven konstanten Geschwindigkeit beginnt und seine Geschwindigkeit auf eine niedrigere positive konstante Geschwindigkeit verringert.

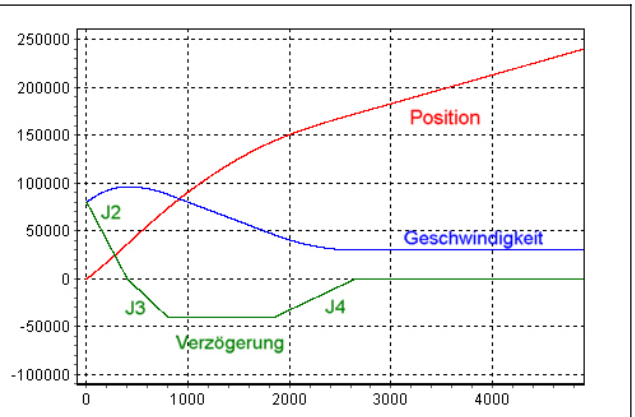
Diese Kurve besteht aus einem Segment Verzögerungsrampe (JERKMIN3), gefolgt von einem Segment konstanter Verzögerung (mit maximaler Verzögerung) und schließlich einer Zunahme der Verzögerung auf eine konstante Geschwindigkeit (JERKMIN4). Beachten Sie, dass die Beschleunigungswerte JERKMIN and JERKMIN2 nicht benutzt werden, weil es nie zu einer Beschleunigung kommt.



Beispiel 5

Dieses Diagramm ist ähnlich dem vorhergehenden, außer dass es mit einer positiven Beschleunigung beginnt.

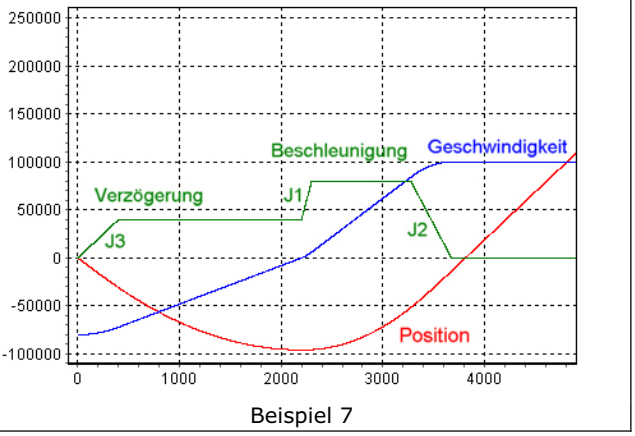
In diesem Fall muss die Kurve mit einer Rücknahme der Beschleunigung beginnen (JERKMIN2). Sobald die Beschleunigung 0 erreicht ist, wird wie im vorhergehenden Beispiel 5 fortgefahren.



Beispiel 6

Das Diagramm zeigt eine Bewegung, die mit einer negativen konstanten Geschwindigkeit beginnt und dann die Richtung zu einer positiven konstanten Geschwindigkeit wechselt. Diese Kurve muss durch Abbremsen der Geschwindigkeit starten, damit sie sich „umdreht". Daher beginnt die Kurve mit einer Verzögerungsrampe (JERKMIN3) bis sie die maximale Verzögerung erreicht.

Die Verzögerung wird mit maximaler Verzögerung fortgesetzt, bis die Geschwindigkeit 0 erreicht ist. Beachten Sie, dass es kein Segment mit Verzögerungsrampe gibt, weil die Bewegung nicht anhält. Exakt bei Geschwindigkeit 0 reversiert die Richtung und die Bewegung wird nun in die andere Richtung beschleunigt. Weil aber in diesem Beispiel die maximale Beschleunigung höher ist als die maximale Verzögerung, kann ein Segment Beschleunigung eingefügt werden (JERKMIN benutzend). Die Kurve endet normal mit einem Segment konstanter Beschleunigung und einer Rücknahme der Beschleunigung auf konstante Geschwindigkeit (JERKMIN2).



Beispiel 7

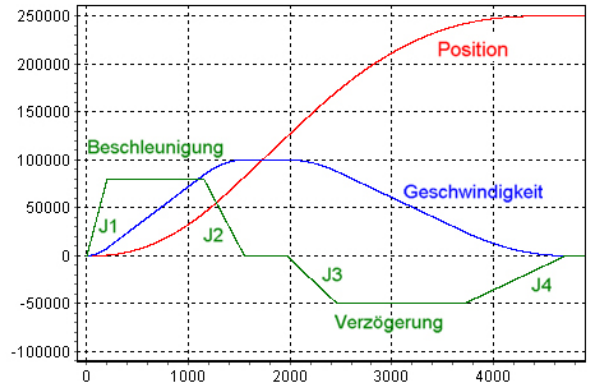
Auf eine definierte Position fahren

Das folgende Diagramm zeigt eine „normale" Bewegung, die von einer Position, an der sie gestoppt hatte, vorwärts fährt, um an einer anderen Position anzuhalten. Die Kurve startet mit einer Verzögerung auf maximale Geschwindigkeit. Dieser Teil der Kurve ist ähnlich der ersten in *„In eine konstante Geschwindigkeit wechseln"* (Beispiel 4). Dieser Kurve wechselt einfach in eine konstante Geschwindigkeit, bei der die konstante Geschwindigkeit die maximale Geschwindigkeit ist.

Daher besteht die Kurve aus einer Beschleunigungsrampe beim Anfahren (JERKMIN), einem Segment konstanter Beschleunigung mit maximaler Beschleunigung und dann einer Rücknahme der Beschleunigung auf maximale Geschwindigkeit (JERKMIN2).

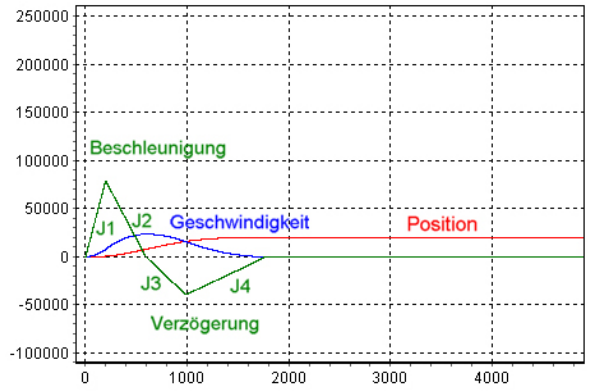
Die Bewegung wird mit maximaler Geschwindigkeit fortgesetzt, bis es notwendig wird die Verzögerungsrampe zu starten, die die Bewegung an der gewünschten Position anhält.

Die Verzögerungsrampe ist identisch zum ersten Beispiel in „Anhalten". Die Kurve besteht aus einer Verzögerungsrampe beim Anhalten (JERKMIN3), gefolgt von einer konstanten Verzögerung (mit maximaler Verzögerung) und schließlich einer Zunahme der Verzögerung auf Geschwindigkeit 0 (JERKMIN4), um an der gewünschten Position anzuhalten.



Beispiel 8

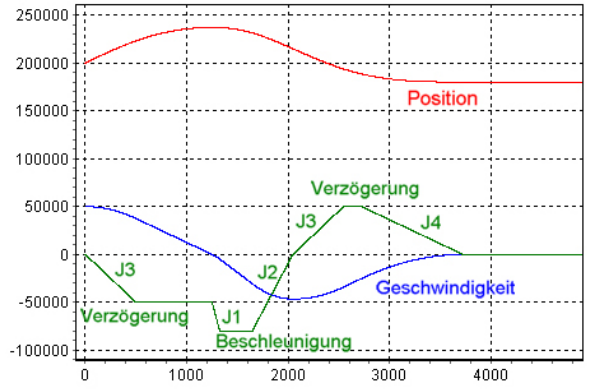
Dieses Diagramm zeigt eine typische „kurze" Bewegung, bei der die maximale Geschwindigkeit nicht erreicht werden kann. In diesem Fall wird so lange wie möglich mit einer Beschleunigung gefahren (JERKMIN). Abhängig davon, wie weit entfernt die Zielposition ist, kann dabei die maximale Beschleunigung erreicht werden oder nicht. An dieser Stelle wird dann die Verzögerung zurückgenommen (JERKMIN2) und sofort mit einem Segment Verzögerungsrampe fortgefahren (JERKMIN3). Abhängig von der Zielposition kann es wieder ein konstantes Verzögerungssegment geben oder nicht. Die Kurve endet mit einer Zunahme der Verzögerung bis Geschwindigkeit 0 in der Zielposition.



Beispiel 9

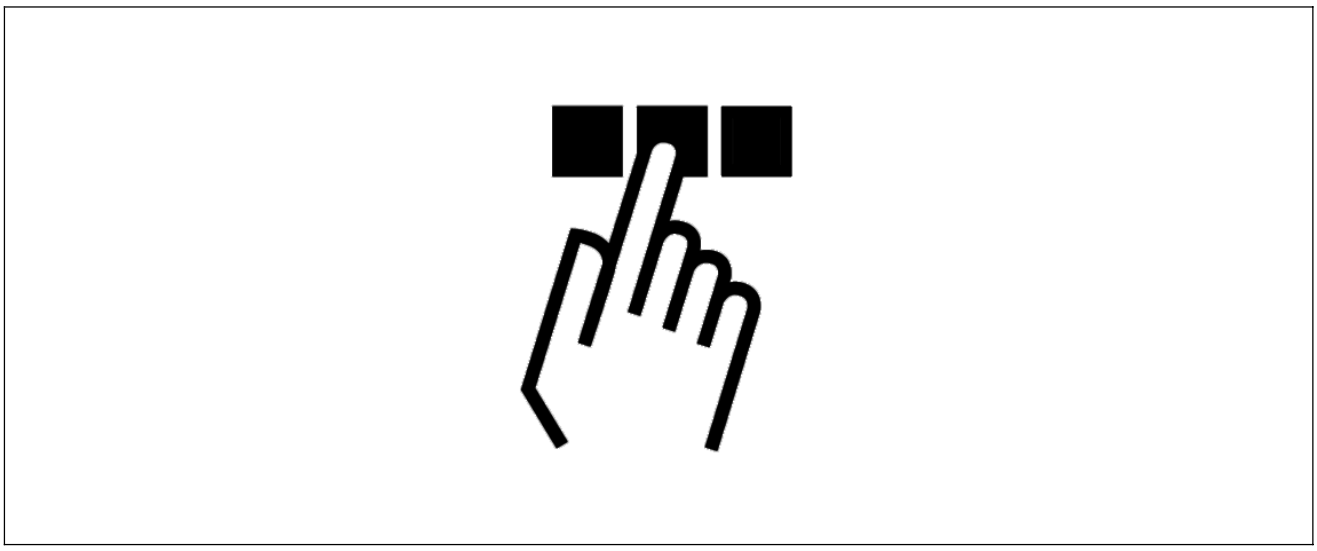
Das Diagramm zeigt ein Beispiel bei dem der Motor anfänglich in die „falsche" Richtung fährt, umgedreht werden und „zurück" auf die Zielposition fahren muss. Weil er „umgedreht" werden muss, startet die Kurve mit einer Verzögerungsrampe (JERKMIN3) bis zur maximalen Verzögerung. Dadurch wird die Geschwindigkeit verlangsamt bis der Motor umdreht. Es wird weiter mit maximaler Verzögerung abgebremst, bis die Geschwindigkeit 0 erreicht ist und die Richtung wechselt.

Exakt an diesem Punkt wird der Motor beschleunigt, aber in die andere Richtung. Von diesem Punkt an ist die Kurve gleich der einer normalen Bewegung zu einer Zielposition, außer dass die ganze Kurve invertiert wird, weil die Richtung gewechselt hat. Die Kurve hat ein Segment Beschleunigungsrampe (Rückwärtsfahrt), sie kann ein Segment konstante Beschleunigung haben oder auch nicht, sie hat ein Segment Beschleunigungsrampe, sie hat oder hat nicht ein Segment konstante Geschwindigkeit, sie hat ein Segment Verzögerungsrampe, sie hat oder hat nicht ein Segment konstante Verzögerung und sie hat schließlich eine Verzögerung zum Stoppen auf der Zielposition.



Beispiel 10

PC Software Benutzeroberfläche



□ APOSS Benutzeroberfläche

Sie sollten mit der Windows-Oberfläche und der Windows-Terminologie vertraut sein, denn diese Gebrauchsanweisung erklärt nicht die Grundlagen, aber alle Besonderheiten der PC Benutzeroberfläche.

Zum Programmieren der MCO 305 Option wird das VLT® Motion Control Tool MCT 10 benutzt. Damit starten Sie auch die integrierte APOSS-Software zum Entwickeln von Steuerungsprogrammen und zum Editieren von Kurven.

Projekte können offline oder mit *Networking* online programmiert werden.

- Online: Wenn MCT 10 eine Verbindung zum Antrieb hergestellt hat, benutzt APOSS diese Verbindung, die MCT 10 schon hergestellt hat.
- Offline: Alle Funktionen, die es erlauben die Antriebe zu steuern oder mehrere Antriebe zu verbinden oder aktuelle Parameter auszulesen, sind freigegeben.

Der Betriebsmodus wird durch MCT 10 beim Starten von APOSS ausgewählt und kann nicht geändert werden, während APOSS läuft.

Wenn APOSS von MCT 10 aus gestartet wird, wird nur ein Antrieb verbunden. Daher sind alle Funktionen, mit denen APOSS Antriebe steuern oder mehrere Antriebe verbinden kann gesperrt.

Wenn MCT 10 weder online noch offline benutzt wird, wird APOSS in einem Stand-alone Modus betrieben.

□ Das APOSS Fenster

Jedes geöffnete Fenster repräsentiert ein APOSS Programm, das mit einem FC 300 verbunden werden kann. Sie können also mindestens so viele Editierfenster öffnen, wie Sie Steuerungen ausgewählt haben.

Titelleiste

Die Titelleiste zeigt Nr. und Name des angeschlossenen FC 300. Tritt ein Fehler auf, wird die Fehlernummer ebenfalls in der Titelleiste der Steuerung, die den Fehler ausgelöst hat, angezeigt.

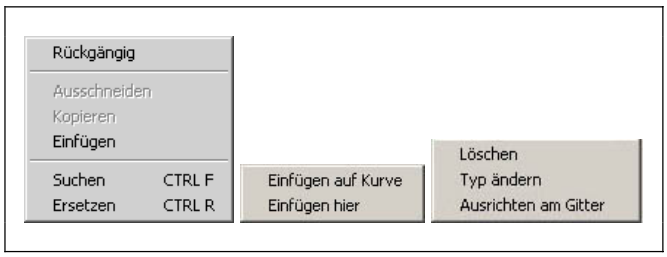
Symbolleiste

Die Symbolleiste bietet neben den Standardfunktionen *Neue Datei, Datei Öffnen* usw. weitere: Von *Info* an nach rechts: *Steuerung auswählen, Schnittstelle öffnen, Schnittstelle schließen und CAM-Editor.*

Kontext-Menüs

An manchen Programmstellen werden Kontext-Menüs angeboten, wenn Sie auf die rechte Maustaste klicken.

Zum Beispiel im Editierfenster oder im → *CAM-Editor* zum Einfügen oder Löschen von Fixpunkten. Die Kontext-Menüs werden automatisch wieder verlassen, wenn die ausgewählte Funktion ausgeführt wird oder wenn Sie mit der linken Maustaste an eine beliebige andere Stelle im Bildschirm klicken.



Editier-Fenster

In diesem Fenster schreiben Sie Ihre Programme mit Hilfe der Funktionen des Menüs *Bearbeiten* wie mit einem Texteditor. Verschiedene Farben erleichtern Ihnen die Unterscheidung zwischen Kommentaren, Programmteilen, Operatoren, Ziffern usw. Sie können die Farbzuordnung mit Einstellungen → *Farben Editor* ändern.

Kommunikationsfenster

Das Kommunikationsfenster ist der untere Teil des Edierfensters. Es zeigt die Meldungen der Steuerung, einschließlich der programmierten PRINT Befehle und Meldungen des Compilers.

□ Tastatur

Alle Tasten außer [Esc] werden genau so benutzt wie in Standard Windows-Anwendungen, zum Beispiel die Pfeil- und Richtungstasten.

[Esc]-Taste

Neben den üblichen Funktionen einer [Esc]-Taste können Sie damit im Programm APOSS jederzeit ein laufendes Programm abbrechen.

ACHTUNG!:
Ein laufender Antrieb wird mit der maximal erlaubten Geschwindigkeit abgebremst!

□ Shortcuts

Tasten werden häufig als sog. Shortcuts mit anderen Tasten entweder als Tastenkombination oder als Tastenfolgen verwendet. Bei einer Tastenkombination müssen Sie die erste Taste gedrückt halten, während Sie die zweite drücken, z.B. [Umschalt] + [Einfg], um den Inhalt der Zwischenablage einzufügen. Bei Tastenfolgen können Sie die Tasten nacheinander drücken, z.B. [Alt] + [B] um das Menü *Bearbeiten* zu öffnen.

Kopieren, Ausschneiden, Einfügen

Die Funktionen Kopieren, Ausschneiden und Einfügen entsprechen exakt der Windows-Spezifikation, zum Beispiel Kopieren mit [Strg] + [Einfg] oder [Strg] + [C].

Cursor positionieren

… entspricht ebenfalls der Windows-Spezifikation, zum Beispiel „Zum Dateiende springen" mit [Strg] + [Ende] oder „Gehe zu Zeile n" mit [Strg] + [G].

Erweitern einer Markierung

… entspricht auch genau der Windows-Spezifikation, zum Beispiel „… um eine Zeile nach unten" mit [Umschalt] + [↓]-Taste**.**

Funktion Rückgängig

Sie können [Alt] + [Rücktaste] oder [Strg] + [Z] benutzen, um die letzte Aktion rückgängig zu machen.

ACHTUNG!:
Datei → *Speichern* löscht den Undo-Speicher.

Makro aufzeichnen

Dieses Shortcut könnte besonders beim Editieren hilfreich sein: [Strg] + [Umschalt] + [R].

□ Funktionstasten

Häufig benötigte Funktionen sind auf die Funktionstasten gelegt, z.B. können Sie die → *Befehlshilfe* für das komfortable Programmieren öffnen. Oder Sie rufen mit [F1] die Online-Hilfe auf. Alle anderen Funktionstasten werden an der passenden Stelle erwähnt.

□ Menü Datei

Das Menü *Datei* enthält Befehle zum Schließen, Speichern, Drucken und Beenden eines Programms. Alle Befehle erreichen Sie wie üblich per Mausklick oder mit der Tastenkombination [Alt] und dem unterstrichenen Buchstaben.

Datei → Neu

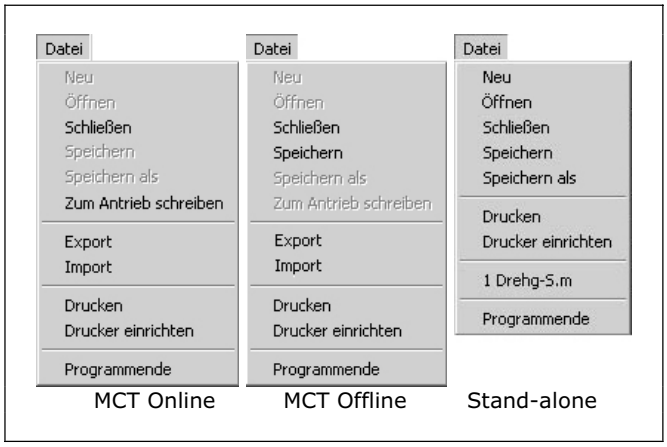
Für neue Dateien benutzen Sie MCT 10 oder im Stand-alone-Modus *Datei → Neu*.

Datei → Öffnen

Wählen Sie die Datei mit MCT 10 aus. Damit wird automatisch APOSS und die Datei geöffnet. Im Stand-alone-Modus benutzen Sie dazu *Datei → Öffnen*.

Datei → Speichern als

Bitte benutzen Sie die Funktionen des MCT 10 um eine Programmdatei (*.m) umzubenennen oder zu kopieren. Oder benutzen Sie → *Speichern als* im Stand-alone-Modus.



Zum Antrieb schreiben

Wenn Sie → *Zum Antrieb schreiben* wählen, wird die aktuell editierte Datei kompiliert, eine Verbindung zum Antrieb hergestellt und dann die kompilierte Datei in einen temporären Speicher in die Steuerung herunter geladen.

Wenn der Download beendet ist, wird das Programm im permanenten Speicher gesichert. Wenn MCT 10 es anfordert, wird auch der Quellcode in den Antrieb herunter geladen.

Export / Import

Die Export/Import-Funktion ermöglicht einen direkten Zugang zu den .m Dateien im MTC 10 online Modus:

Durch Klicken auf *Datei → Export* wird das „Sichern als" Dialogfeld zum Sichern der .m Datei im gewünschten Verzeichnis geöffnet.

Datei → Import öffnet das „Datei öffnen" Dialogfeld, mit dem Sie eine früher gesicherte .m Datei wieder importieren können.

ACHTUNG!:
Diese importierte Datei überschreibt die existierende .m Datei, die gerade editiert wird, d.h. es wird alles gelöscht, was gerade editiert wird und mit dem Inhalt der importierten Datei ersetzt. Benutzen Sie „Abbrechen", wenn Sie die Originaldatei nicht überschreiben wollen.

Programmende

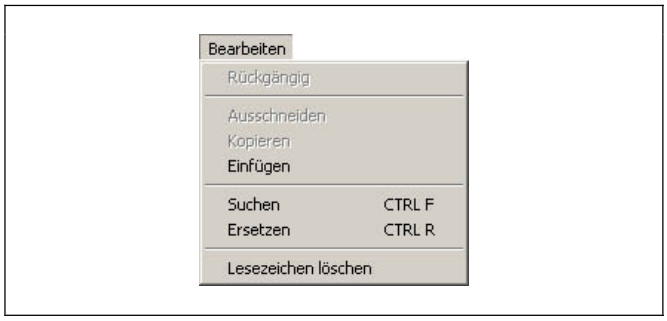
Das Programm APOSS kann durch Klicken auf → *Programmende* oder auf das ☒ Symbol beendet werden. Falls Sie eine neue oder geänderte Datei noch nicht gespeichert haben, haben Sie jetzt die Möglichkeit, dies zu tun.

ACHTUNG!:
Programmende beendet aber nicht ein laufendes Programm in der Steuerung. Ein Programm können Sie nur mit [Esc] abbrechen oder beenden. Dazu muss auch die Datei, die mit der Steuerung verbunden ist, geöffnet sein bzw. wieder geöffnet werden.

ACHTUNG!:
Wenn die Steuerung bei *Datei* → *Programmende* dennoch stehen bleibt, kann es daran liegen, dass von der Steuerung PRINT-Befehle geschickt werden, die nun nicht mehr im Kommunikationsfenster dargestellt werden können.

□ Menü Bearbeiten

Das Menü *Bearbeiten* bietet die zum Programmieren notwendigen Editierhilfen, von denen Sie die meisten auch – wie in Windows gewohnt – über Tasten und Tastenkombinationen erreichen können.



Einige Editierhilfen erreichen Sie nur über Tastenkombinationen, z.B.

Zeilenweises Löschen	[Strg] + [Y]
Gehe zu Zeile n	[Strg] + [G]
Zeile darüber einfügen	[Strg] + [Umschalt] + [N]

Tabulatoren

Nutzen Sie die Tabulatoren und die verschiedenen Farben, um das Programm optisch zu strukturieren. Die Tab-Schritte sind fest eingebaut.

Zeilennummer

Innerhalb des Programms können Sie sich an den Zeilennummern orientieren. Die Syntaxprüfung zum Beispiel stellt nicht nur den Cursor in die entsprechende Zeile, sondern nennt auch die Zeilennummer mit dem falschen Befehl.

Die aktuelle Zeilennummer finden Sie in der Statuszeile, zum Beispiel 13:1. Der Cursor steht dann in der Zeile 13 auf Schreibposition 1.

Suchen und Ersetzen

Suchen und Ersetzen ist gemäß den Windows-Konventionen realisiert und mit einigen nützlichen Funktionen ergänzt.

Klicken Sie auf *Bearbeiten* → *Suchen* oder drücken Sie [Strg] + [F] und geben im folgenden Dialogfeld den gesuchten Begriff ein. Mit [F3] können Sie dann von einer Fundstelle zur nächsten springen.

Klicken Sie auf → *Alle Markieren* und es werden sofort alle Fundstellen am linken Rand mit einem blauen Dreieck markiert. Sie können dann mit [F2] von einer Fundstelle zur anderen springen.

Reguläre Ausdrücke

Diese Funktion ist in Suchen und Ersetzen mit folgenden Syntax-Regeln realisiert:

Wildcards	? (für beliebiges Zeichen), + (für ein oder mehrere Suchbegriffe), * (für kein oder mehrere Zeichen).
Zeichengruppe	Zeichen in eckigen Klammern werden als Gruppe gesucht; der Bereich wird mit Bindestrich angegeben, z.B. [a-c].
Logisches ODER	Unterausdrücke werden mit Hilfe des Pipeline-Symbols | mit ODER verknüpft.
Unterausdrücke in Anführungszeichen	Ein regulärer Ausdruck sollte in Anführungszeichen gesetzt werden und wird als eine Einheit behandelt
Code-Umschalt-zeichen	Abläufe wie \t, etc. werden durch ein äquivalentes einzelnes Zeichen ersetzt. \\ stellt den Backslash dar.

Lesezeichen löschen

Wenn im Editor Lesezeichen benutzt werden, werden diese gespeichert und mit der Programmdatei wiederhergestellt.

Klicken Sie auf → *Lesezeichen löschen,* um alle vorhandenen Lesezeichen aus dem Editor zu löschen.

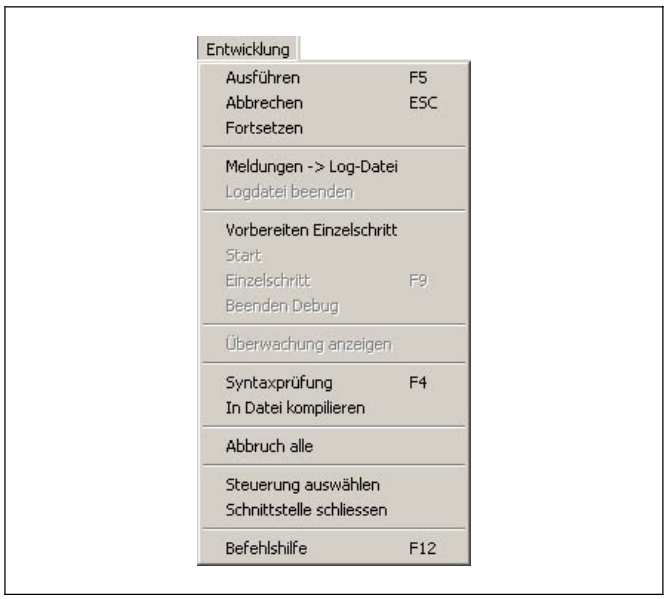
□ Menü Entwicklung

Mit den Funktionen des Menüs *Entwicklung* können Sie die Programme ausführen, abbrechen, fortsetzen oder bei der Fehlersuche schrittweise ausführen. Ein Debug-Modus sowie die Möglichkeit, während der Programmausführung die Variablen zu ändern, erleichtern das Programmieren.

Bevor Sie jedoch beginnen, müssen Sie immer eine Steuerung bzw. einen FC 300 auswählen.

In der *Befehlshilfe* finden Sie mehrere hilfreiche Funktionen: Erstens listet sie übersichtlich alle APOSS-Befehle auf, die auch sofort in das Editierfenster übernommen werden können. Zweitens können Sie hier mit der Teach-in-Programmierung arbeiten.

Im Offline-Modus sind alle Funktionen, die einen Zugriff auf den Antrieb erfordern, nicht verfügbar. Die meisten Funktionen im Menü *Entwicklung* sind daher gesperrt. APOSS benutzt den angeschlossen Antrieb, den MCT 10 bereits verbunden hat.



□ Ausführen [F5]

Das Programm das geöffnet und im Editor dargestellt ist, wird gestartet.

Dazu wird das Programm kompiliert und in den FC 300 geladen. Gleichzeitig wird das Programm in den temporären Bereich des RAM's geladen, der mit jedem weiteren Ausführen überschrieben wird. Beim Programmieren haben Sie so einen schnellen unkomplizierten Arbeitsablauf zum Testen.

ACHTUNG!:
Allerdings ist es nicht möglich, eine kompiliertes Programm wieder zurück in den PC zu holen, bzw. die Quelldatei wieder im PC zu bearbeiten. Daher sollten Sie alle Programme grundsätzlich auch auf der Festplatte des PCs speichern.

□ Abbrechen [Esc] und Abbruch alle

Klicken Sie auf *Entwicklung* → *Abbrechen* oder drücken Sie [Esc] um das Programm sofort abzubrechen. Dabei werden auch eventuell aktive Fahrprozesse vorzeitig beendet.

ACHTUNG!:
Es wird mit der maximal zulässigen Verzögerung abgebremst.

Falls das Programm in mehreren Steuerungen läuft, benutzen Sie *Entwicklung* → *Abbruch alle*, um die laufenden Programme abzubrechen.

□ Programm Fortsetzen

Klicken Sie auf *Entwicklung* → *Fortsetzen*, um das eben abgebrochene Programm fortzusetzen. Dabei werden auch die unterbrochenen Fahrprozesse zu Ende ausgeführt.

Wenn ein Programm mit einer Fehlermeldung abgebrochen wurde, können Sie es – nachdem Sie den Fehler behoben und/oder die Fehlermeldung gelöscht haben – mit dieser Funktion wieder → *Fortsetzen*.

☐ Meldungen -> Log-Datei

Mit dieser Funktion starten Sie die Protokollierung der Meldungen in eine Datei. Beachten Sie, dass *Logdatei beenden* erst aktiviert wird, wenn die Protokollierung gestartet wurde.

☐ Debug-Modus

Das schrittweise Abarbeiten (Tracing) eignet sich vor allem für den Test von neu entwickelten Programmen und kann bei der Fehlersuche sehr hilfreich sein.

Vorbereiten Einzelschritt

Mit *Entwicklung → Vorbereiten Einzelschritt* wird das geöffnete Programm für den Debug-Modus vorbereitet: Es wird kompiliert und eine Debug-Datei erzeugt, das Programm wird in den FC 300 geladen und es werden alle ausführbaren Programmzeilen durch blaue Punkte gekennzeichnet. Nun sind auch die entsprechenden Menüpunkte verfügbar.

Haltepunkte setzen

Sie können vor jede mit einem blauen Punkt markierte Programmzeile durch Doppelklick einen Haltepunkt setzen. Dieser wird rot markiert.

Die Programmausführung stoppt dann, bevor diese Programmzeile – die gelb markiert wird – ausgeführt wird.

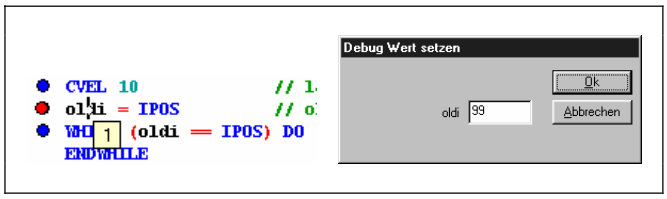
Ein weiterer Doppelklick ändert die roten Haltepunkte wieder in blaue Markierungen für die Programmzeilen, die beim Tracing übersprungen werden sollen, im Debug-Modus also nicht angehalten wird.

ACHTUNG!:
In Abhängigkeit von der Geschwindigkeit der Programmausführung und Kommunikation sollte die Anzahl der Haltepunkte auf ein vernünftiges Maß begrenzt werden. Maximal erlaubt sind 10 Haltepunkte.

ACHTUNG!:
ON PERIOD Funktionen sollten Sie beim Debugging deaktivieren, da der interne Timer während der Pausen bei den *Einzelschritten* weiterläuft. Das Programm versucht dann später die ON PERIOD Funktionen nachzuholen, was zu Problemen führen kann.

Variablen lesen oder online ändern

Im Debug-Modus können Sie nach der Programmausführung den aktuellen Wert der Variablen auslesen. Klicken Sie mit der linken Maustaste auf die Variable und der Wert wird solange dargestellt, bis Sie den Mauscursor wieder bewegen.



Im Debug-Modus können Sie die Variablen während der Programmausführung ändern (Debug Wert setzen). Achten Sie dabei darauf, dass eine solche Änderung im Programm auch sinnvoll ist. Klicken Sie mit der → rechten Maustaste auf die Variable und setzen Sie im darauf folgenden Feld den gewünschten neuen Wert.

Start (Debug) und Einzelschritt

Die Programmausführung stoppt beim ersten Haltepunkt und wartet auf eine Eingabe:

Um die nächste Programmzeile auszuführen, klicken Sie auf *Entwicklung → Einzelschritt* oder tasten [F9].

Um das Programm bis zum nächsten Haltepunkt abzuarbeiten, klicken Sie auf *Entwicklung → Ausführen* oder tasten [F5].

Mit [F9] hält das Programm also vor der nächsten Programmzeile, mit [F5] vor jedem Haltepunkt.

Programmausführung im Debug-Modus abbrechen

Klicken Sie auf *Entwicklung* → *Abbrechen* oder tasten Sie [Esc] um die Programmausführung sofort abzubrechen; dabei werden auch eventuell aktive Fahrprozesse vorzeitig beendet.

ACHTUNG!:
Es wird mit der maximal zulässigen Verzögerung abgebremst.
Danach steht der Cursor in der Programmzeile, die als Nächstes ausgeführt werden sollte. Sie können mit *Entwicklung* → *Ausführen* [F5] oder → *Einzelschritt* [F9] fortfahren.

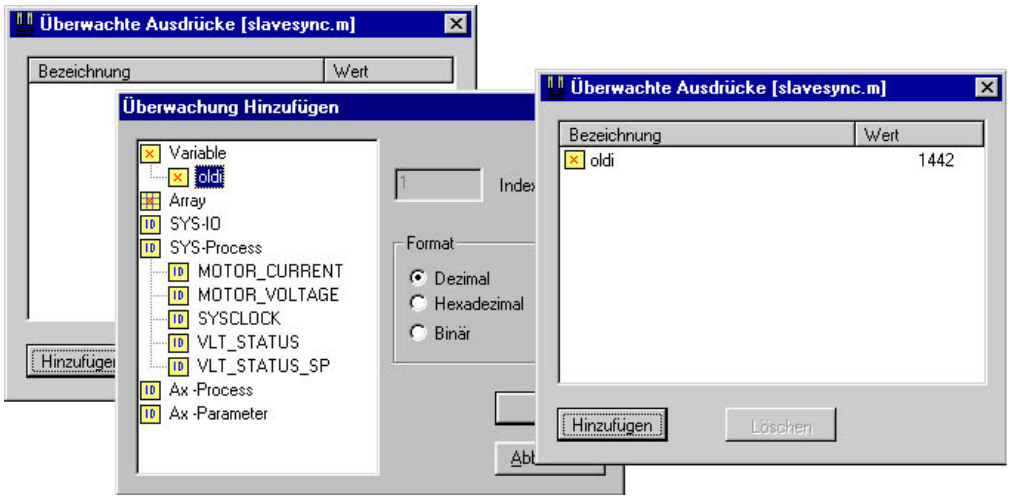
Beenden Debug

Mit *Entwicklung* → *Beenden Debug* wird die Programmausführung sofort beendet und der Debug-Modus verlassen. Die Markierung der Programmzeilen wird entfernt, die Haltepunkte werden aber weiter angezeigt, damit sie beim nächsten Debugging wieder benutzt werden können. Wenn Sie also Programmzeilen einfügen, „wandern" die Haltepunkte mit.

□ Überwachung anzeigen

Diese Funktion ermöglicht die Online-Überwachung der Variablen, Arrays, System- und Achsprozessdaten (gemäß der SYSVAR Indizes) und Achsenparameter.

Klicken Sie auf *Entwicklung* → *Überwachung anzeigen* und im folgenden Dialogfenster auf → *Hinzufügen*. Das nächste Dialogfenster bietet die Variablen, Arrays und Parameter zur Auswahl:



Mit Doppelklick auf den gewünschten Typ, zum Beispiel Variable erhalten Sie alle im Programm verwendeten Variablen zur Auswahl. Markieren Sie den Ausdruck, der überwacht werden soll und wählen Sie aus, in welchem Format (Dezimal, Hexadezimal, Binär) er angezeigt werden soll. Dann klicken Sie auf *OK*.

Sie können weitere Ausdrücke zur *Überwachung* → *Hinzufügen* und natürlich auch wieder → *Löschen*. Es können maximal 10 Ausdrücke gleichzeitig überwacht werden.

ACHTUNG!:
Das Überwachungsfenster wird ständig aktualisiert. Daher sollte in Abhängigkeit von der Geschwindigkeit der Programmausführung und Kommunikation die Anzahl der überwachten Ausdrücke auf ein vernünftiges Maß begrenzt werden.

ACHTUNG!:
Die Überwachung der Arrays ist auf die ersten 250 Elemente begrenzt.

Überwachungsfenster ändern

Zum Ändern der Größe dieses Dialogfensters, stellen Sie den Cursor an die untere rechte Ecke des Dialogfensters und klicken – sobald der Cursor seine Form ändert – und ziehen ihn in die gewünschte Richtung.

Überwachungsfenster schließen

Klicken Sie auf *Entwicklung* → *Überwachung schließe*n oder auf das Schließen-Symbol im Dialogfenster. Wenn Sie es später erneut öffnen, werden die zuvor ausgewählten Ausdrücke wieder online überwacht und angezeigt.

□ Syntaxprüfung [F4]

Das Programm wird abgebrochen, sobald ein fehlerhafter Befehl gefunden wird. Im Kommunikationsfenster wird die Zeilennummer genannt und eine Fehlerbeschreibung ausgegeben. Automatisch wird der Cursor in die Zeile genau an die Position mit dem Syntaxfehler gestellt und das Programm stoppt an dieser Stelle.

Die *Syntaxprüfung* erzeugt zusätzlich zur Prüfung eine Debug-Datei und speichert diese als „temp.ad$".

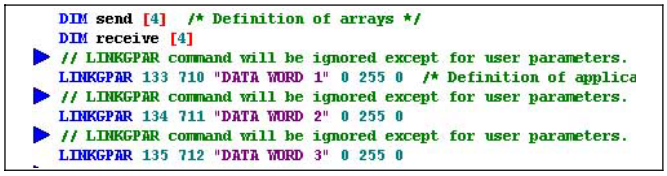
□ In Datei kompilieren

Mit dieser Funktion kann man die aktuelle Datei kompilieren und als binäre Datei speichern. Ein „Speichern als"-Dialog bietet die Eingabe eines Dateinamens an; als Standard wird der aktuelle Dateiname mit „.bin" als Dateierweiterung benutzt.

ACHTUNG!:
Diese Funktion ist nur verfügbar, wenn *Binär-Datei erzeugen* in Einstellungen → *Optionen* aktiviert ist.

□ VLT5000 > MCO 305 Konvertierung

Dieser Konverter prüft Ihre früheren Programme, erstellt eine Zusammenfassung der erforderlichen Änderungen und fügt Kommentare an den Stellen ein, wo Änderungen notwendig sind. Beachten Sie: Der Konverter ändert nicht automatisch Ihr Programm, siehe Beispiel „LINKGPAR Befehl".



□ Befehlshilfe [F12]

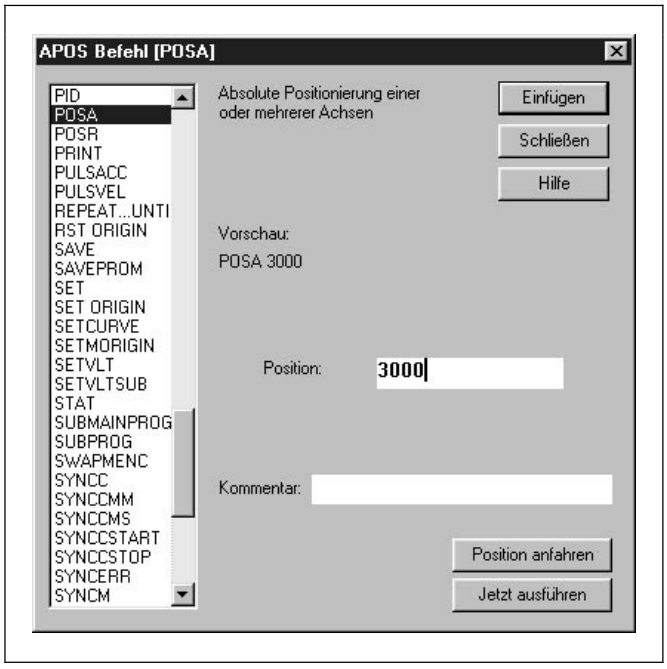
Die *Befehlshilfe* zeigt alle Befehle mit der entsprechenden Syntax, die Sie ganz einfach in Ihr Programm → *Einfügen* können.

Sie erhalten ausführliche Informationen zu einem markierten Befehl, wenn Sie auf → *Hilfe* klicken oder [F1] drücken.

Außerdem kann die Steuerung mittels der Teach-in-Funktion (→ *Position anfahren)* programmiert werden.

Im Betriebsmodus Offline ist es nicht möglich einen Befehl direkt auszuführen oder den Antrieb mit der Funktion → *Position anfahren* zu bewegen.

Geben Sie die Position in das Feld für die Achse ein. Die Vorschau zeigt Ihnen die genaue Syntax des Befehls. Sie haben nun drei Alternativen zur Auswahl, die Sie zur Programmierung des FC 300 beliebig mischen können.



ACHTUNG!:
Während des Programmierens werden die eingegebenen Wert grundsätzlich weder getestet noch wird der zulässige Eingabebereich geprüft. Wegen der vielfältigen Anwendungsmöglichkeiten und verschiedenen Motorleistungsklassen ist dies weder möglich noch erwünscht.

Einfügen oder Jetzt ausführen

Stellen Sie den Cursor im Editierfenster an die Stelle, wo Sie ein oder mehrere Befehle einfügen wollen, wählen Sie *Entwicklung* → *Befehlshilfe* und hier den Befehl aus, zum Beispiel POSA, ergänzen den Wert und ggf. einen Kommentar und klicken Sie auf → *Einfügen*.

Oder klicken Sie auf → *Jetzt ausführen* und testen diesen Befehl, bevor Sie ihn in Ihr Programm → *Einfügen.*

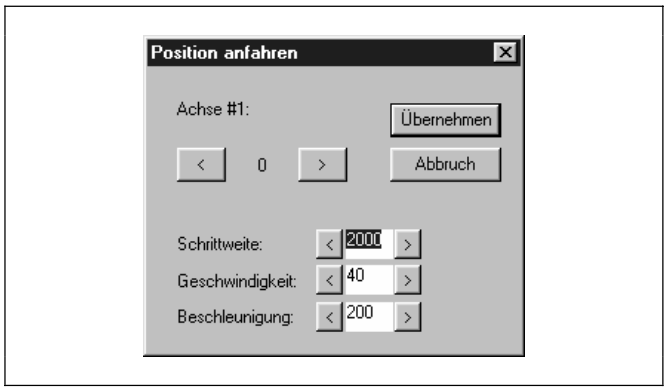
ACHTUNG!:
Freigegebene Antriebe laufen an.

Position anfahren

Oder Sie nutzen die Teach-in-Funktion und klicken auf→ *Position anfahren*: Im Dialogfeld wird die aktuelle Position der Achse angezeigt.

Klicken Sie auf das Vorwärts- > oder Rückwärtssymbol < und fahren Sie den Antrieb an die gewünschte Position: Schrittweise mit einzelnen Mausklicks, Dauerfahrt durch Festhalten der Maustaste.

Wenn der Antrieb die gewünschte Position erreicht hat, klicken Sie auf → *Übernehmen* und der Wert wird in das Dialogfeld zur Achse eingetragen.



☐ Steuerung auswählen

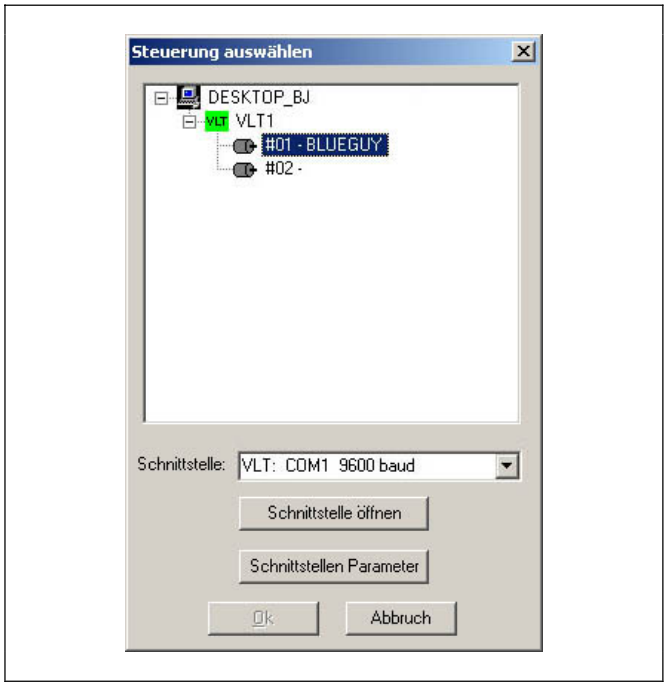
Wenn Sie mehr als einen FC 300 konfiguriert haben, wählen Sie mit *Entwicklung* → *Steuerung auswähle*n den FC 300 aus, in den Sie Programme laden und starten wollen. Alle aktuell verfügbaren Steuerungen werden in einem Verzeichnisbaum dargestellt.

Markieren Sie die gewünschte Steuerung und klicken Sie auf *OK* um die Steuerung zu verbinden. Falls keine Steuerungen gezeigt werden oder die gewünschte Schnittstelle nicht vorhanden ist, dann wählen sie diese im Popup-Menü aus und klicken auf → *Schnittstelle öffnen*.

Programme in mehreren FC 300 ausführen

Wenn Sie das Programm in mehrere Steuerungen laden wollen, verbinden Sie das Programm mit dem jeweiligen FC 300 und klicken auf → *Ausführen [F5]*.

Wenn Sie in jeder Steuerung ein anderes Programm laden wollen, öffnen Sie für jeden FC 300 ein eigenes Editierfenster, öffnen dort die gewünschte Programmdatei und verbinden es mit → *Steuerung auswählen* mit den FC 300. Dann starten Sie nacheinander jedes Programm mit *Entwicklung* → *Ausführen* oder [F5].

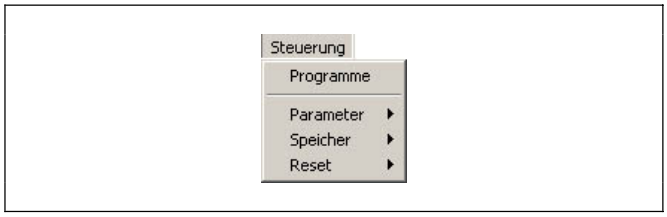


□ Schnittstelle schließen

Wenn Sie diese Funktion auswählen wird eine aktuell offene Schnittstelle zu einer Motorsteuerung geschlossen. Wenn keine Schnittstelle geöffnet ist, hat diese Funktion keine Auswirkung.

□ Menü Steuerung

Mit den Funktionen im Menü *Steuerung* verwalten Sie Ihre Programme: Sie speichern oder löschen die Programme im EEPROM der Steuerung und kennzeichnen ein Programm für einen *Autostart*.



□ Programme

Klicken Sie auf *Steuerung* → *Programme* und das Dialogfeld zeigt alle angeschlossenen Steuerungen. Es ist der FC 300 markiert, mit dem das Programm im Editierfenster gerade verbunden ist. Sie können natürlich auch einen anderen FC 300 markieren und bearbeiten.

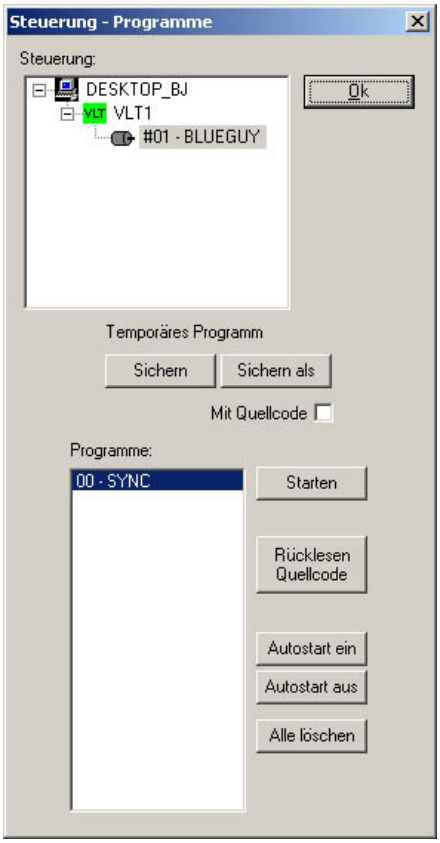
Temporäres Programm sichern

Immer wenn Sie ein Programm ausführen, wird es in einen temporären Bereich im RAM geladen, der mit jedem weiteren Ausführen überschrieben wird. Sie können das zuletzt ausgeführte temporäre Programm jetzt dauerhaft → *Sichern*.

ACHTUNG!:
Es empfiehlt sich, ein noch nicht kompiliertes Programm immer auch auf der Festplatte des PCs zu speichern und zu archivieren, da eine kompilierte Quelldatei im FC 300 nicht mehr bearbeitet werden kann.

Klicken Sie auf → *Sichern* und geben Sie im folgenden Dialogfeld einen Namen ein oder bestätigen Sie den vorgeschlagenen Dateinamen. Die Programmnummer wird automatisch vergeben.

ACHTUNG!:
Wenn versucht wird, ein neues Programm zu sichern, aber schon ein Programm aktiv ist, dann kann das neue Programm nicht gesichert werden.
In diesem Fall bietet ein Dialogfeld die Möglichkeit das aktuell aktive Programm abzubrechen. Danach wird dann das neue Programm gesichert.



Sichern als

Klicken Sie auf → *Sichern als* und Sie können zusätzlich zum Namen auch die Programmnummer (0 bis 90) selbst bestimmen.

Über diese Programmnummer kann ein beliebiges Programm auch über die Eingänge, zum Beispiel von einer SPS aus, gestartet werden. Dazu sind die Eingänge mit *Steuerung* → *Parameter* → *Global* entsprechend zu setzen.

Sichern mit Quellcode

Wenn das Kontrollkästchen aktiviert ist, wird zusätzlich zur kompilierten und direkt ausführbaren Programmdatei der Quellcode im FC 300 gesichert. Sie können diesen bei Bedarf wieder zurücklesen und auf dem PC in einer Datei speichern.

Enthält ein Programm eingebundene Dateien (Include-Dateien), wird der Quellcode um diese erweitert und sie werden auch mit heruntergeladen. Dies erlaubt es, statt Programmteile das komplette Programm in der Steuerung zu speichern.

Klicken Sie auf → *Sichern als* und geben Sie einen Namen in das folgende Dialogfeld ein oder bestätigen Sie den Dateinamen. Der vorgeschlagene Name enthält das Datum und die Uhrzeit, so dass ein unbeabsichtigtes Überschreiben der Datei beim Zurücklesen ausgeschlossen wird.

Der Quellcode wird im Flash-EPROM gespeichert. Falls dort dafür nicht genügend Platz ist, erhalten Sie eine Meldung und müssten dann andere Programmdateien löschen, bevor Sie die neue speichern.

Alle mit Quellcode gesicherten Programme werden mit einem '+' gekennzeichnet

Programm Starten

Sie können in diesem Dialogfenster ein Programm auswählen und direkt → *Starten.*

Rücklesen Quellcode

Alle mit '+' gekennzeichneten Programme können Sie im Quellcode-Format wieder aus der Steuerung auslesen und auf Ihrem PC zur weiteren Verwendung ablegen.

Wählen Sie das gewünschte Programm aus und klicken Sie auf → *Rücklesen Quellcode*. Sie können die Datei dann wie gewohnt bearbeiten oder für andere FC 300 duplizieren.

Autostart

Mit Autostart kennzeichnen Sie ein Programm, das künftig nach dem Einschalten des FC 300 sofort gestartet wird. Markieren Sie das gewünschte Programm und klicken Sie auf *Autostart ein*. Das ausgewählte Programm wird mit einem ***** gekennzeichnet.

Wenn Sie einen gesetzten Autostart aufheben wollen, klicken Sie auf *Autostart aus* oder Sie kennzeichnen gleich ein anderes Programm.

Um mehrere Programme mit Autostart ablaufen zu lassen, nutzen Sie den Par. 33-80 *Aktivierte Programmnummer*. Damit können Sie festlegen, welches Programm nach Ablauf des per Autostart ausgeführten Programms gestartet werden soll.

Wenn in den Parametern 33-80 PRGPAR, 33-5* I_FUNCTION_n_13 oder I_FUNCTION_n_14 nichts anderes festgelegt ist, wird immer wieder das mit *Autostart* gekennzeichnete Programm gestartet.

Ein gesetzter Autostart wirkt sich wie folgt aus:

Wenn bei einem Kaltstart kein Fehler vorliegt (Ausnahme Schleppfehler, Endschalter-Fehler und SW-Endschalter-Fehler) wird das entsprechende Autostart-Programm gestartet.

Wird das Autostart-Programm durch einen Abbruch des Benutzers (APOSS) gestoppt, wird es nicht wieder gestartet, es sei denn es findet ein neuer Kaltstart statt. In diesem Fall wird auch kein Programm auf Grund von Eingängen oder Par. 33-80 PRGPAR gestartet.

Wird das Autostart-Programm durch einen Fehler abgebrochen (weil keine ON ERROR Routine definiert wurde) oder normal beendet, wird anschließend geprüft, ob ein Start durch Eingänge vorgesehen oder ob der Par. 33-80 PRGPAR gesetzt ist. Wenn ja, wird das entsprechende Programm ausgeführt, bzw. auf den Start-Eingang gewartet. Wenn nicht, wird das Autostart-Programm wieder von vorne begonnen. Daraus folgt:

Autostart-Programm einmal ausführen

Wenn prinzipiell ein Start von Programmen über den Par. 33-80 PRGPAR oder über Eingänge vorgesehen ist, wird das Autostart-Programm nur einmal ausgeführt (zum Beispiel für HOME-Aufgaben).

Wiederholende Ausführung des Autostart-Programms

In den anderen Fällen wird das Autostart-Programm immer wieder gestartet.

So kann man auch ein Programm mit einem EXIT Befehl einfach wieder von vorne starten. Dies ist dann nützlich, wenn man in einer Fehlersituation (ON ERROR) nicht mit RETURN fortfahren, sondern zum Beispiel eine erneute Homefahrt erzwingen will. Es sollte allerdings darauf geachtet werden, dass kein Fehler (außer Schleppfehler, Endschalter-Fehler und SW-Endschalter-Fehler) vorliegt, da sonst das Autostart-Programm nicht wieder gestartet wird.

Verkettung von Autostart-Programmen

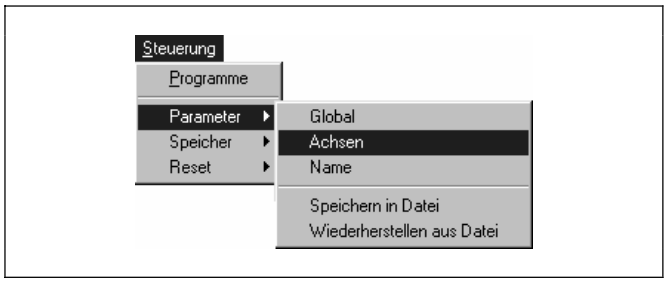
Das Starten über Par. 33-80 PRGPAR kann natürlich auch zur Verkettung benutzt werden: Nachdem ein Programm gestartet wurde, kann die mit Par. 33-80 definierte Programmnummer umgesetzt und so bestimmt werden, welches Programm als Nächstes ausgeführt werden soll.

ACHTUNG!:
Wenn kein Autostart-Programm definiert ist, kann auch kein Programm über Par. 33-80 *Aktivierte Programmnummer* gestartet werden; dies erfordert immer ein beendetes Autostart-Programm.

Alle Programme löschen

Klicken Sie auf *Alle Löschen*, wenn Sie alle Programme im FC 300 löschen wollen. Vergewissern Sie sich zuvor, dass Sie die Programme noch im PC zur Sicherheit oder für das Archiv gespeichert haben.

□ Steuerung > Parameter



Die Parameter im Menü *Steuerung* sind in zwei Gruppen unterteilt: Globale Parameter, die für die gesamte Steuerung gelten und Achsparameter, die für jede Achse unterschiedlich sein können. Alle Details der Parameter und die Parameterkennungen mit den Werkseinstellungen finden Sie in der Parameter-Referenz.
Oder drücken Sie [F1] wenn der Mauszeiger in einem der Eingabefelder steht und Sie erhalten Information zu diesem Parameter.

Globale Parameter und Achsparameter

Zu den globalen Parametern gehören die Funktionen der Ein- und Ausgänge und die Standardparameter, die Sie in den Gruppen 33-5* und 33-8* finden.

Die Achsparameter sind immer für alle Programme, die zu einer Steuerung gehören, gültig.

Markieren Sie den FC 300 den Sie bearbeiten wollen. Sie können jeden voreingestellten Wert einzeln verändern. Klicken Sie auf → *OK* um die Änderungen in den FC 300 zu laden.

Mit *Reset* → *Parameter* im Menü *Steuerung* erhalten Sie wieder die Werkseinstellungen; allerdings werden dabei alle – also auch alle Achsparameter – auf die ab Werk eingestellten Werte zurückgesetzt.

Es gibt zwei Möglichkeiten die Parameter einzustellen oder zu ändern:

Achsparameter online einstellen oder ändern

Klicken Sie auf *Steuerung* → *Parameter* → *Achsen* und markieren Sie im folgenden Dialogfenster die Steuerung, deren Parameter Sie sehen oder ändern wollen. Wählen Sie außerdem im Feld *Parameter* den Typ aus:

Slave Encoder	Par. Gruppe 32-0*
Master Encoder	Par. Gruppe 32-3*
Homefahrt	Par. Gruppe 33-0*
Eingänge/Ausgänge (inkl. SW-Endschalter)	Par. Gruppe 33-4* und 33-5*
PID-Regelung	Par. Gruppe 32-6*
Synchronisierung	Par. Gruppe 33-1*
Geschwindigkeit	Par. Gruppe 32-8*

Sie können jeden Parameter ändern und mit Klicken auf *OK* wieder in den FC 300 laden. Sie können aber auch sofort einen anderen FC 300 auswählen, die Parameter ändern und dann mit *OK* alle Änderungen gleichzeitig in den FC 300 laden.

Parameter einer Konfigurationsdatei CNF ändern

Zusätzlich zu der Möglichkeit die Parameter online zu ändern, können Sie auch alle Parameter-Einstellungen einer gespeicherten Konfigurationsdatei (CNF) ändern. Dazu öffnen Sie den → *CAM-Editor* und ändern die Parameter in den entsprechenden Registerkarten.

ACHTUNG!:
Diese Änderungen betreffen aber nur die Konfigurationsdatei, nicht die Parameter in der Steuerung. Wenn diese geänderten Einstellungen der Konfigurationsdatei auch für die Steuerung gelten sollen, müssen Sie die CNF-Datei mit *Steuerung → Parameter → Wiederherstellen aus Datei* in die Steuerung laden.

Reset Parameter

Wenn Sie für alle Achsparameter wieder die Standardeinstellungen wünschen, klicken Sie auf *Steuerung → Reset → Parameter.*

ACHTUNG!:
Dabei werden aber auch die globalen und I/O-Parameter auf die Werkseinstellungen zurückgesetzt.

Parameter > Name

Sie können zusätzlich zur Nummer für jeden FC 300 einen Namen eingeben oder einen vorhandenen mit dieser Funktion ändern. Klicken Sie auf *Steuerung → Parameter → Name* und wählen Sie im Dialogfeld den FC 300 aus, den Sie in Betrieb nehmen wollen, geben einen maximal 8-stelligen Namen ein oder überschreiben den vorhandenen und klicken auf *OK***.**

Parameter > Speichern in Datei und > Wiederherstellen aus Datei

Mit *Parameter → Speichern in Datei* sichern Sie die Benutzerparameter inklusive der Arrays in eine Datei mit der Extension „.CNF"

So können Sie schnell die Parameter in anderen FC 300 laden oder später jederzeit wieder in den FC 300, zum Beispiel nach EEPROM löschen.

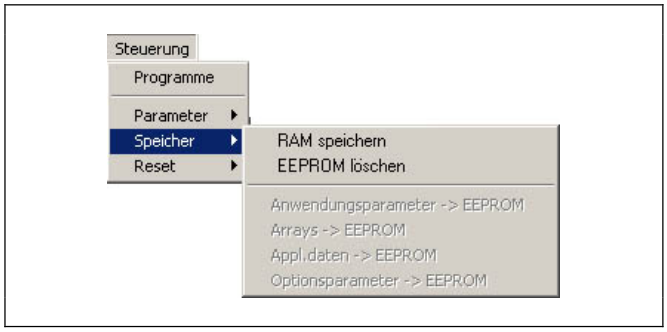
Wählen Sie die Steuerung aus und klicken Sie dann auf → *Sichern*. Geben Sie einen Namen ein oder bestätigen Sie den vorgeschlagenen. Dieser enthält das Datum und die Uhrzeit, so dass ein unbeabsichtigtes Überschreiben der Parameter beim Wiederherstellen ausgeschlossen wird. Falls Parameter von mehreren Steuerungen gesichert werden sollen, dann wählen Sie einfach nacheinander die Steuerungen aus und → *Sichern* wieder.

Klicken Sie auf *Parameter → Wiederherstellen aus Datei* und wählen Sie die Datei aus, die geladen werden soll. Im folgenden Dialogfeld wählen Sie den FC 300 aus, in die die Daten geladen werden sollen und klicken auf → *Wiederherstellen aus Datei*. Sofort werden die gespeicherten Benutzerparameter inklusive der Arrays in die Steuerung geladen.

Falls die gleichen Daten in mehr als einen FC 300 geladen werden sollen, dann wählen Sie einfach einen anderen aus und klicken erneut auf → *Wiederherstellen aus Datei*.

□ Steuerung > Speicher

Zusätzlich zu → *RAM speichern* und → *EEPROM löschen* gibt es Funktionen um individuell Daten im LCP-Speicher zu sichern, zum Beispiel → *Optionsparameter.*



Speicher > RAM speichern

Die Funktion *RAM speichern* wird normalerweise nicht benötigt, da Programme und Parameter automatisch gesichert werden. Aber mit → *RAM speichern* können Sie zusätzlich aktuelle Array-Werte mit in das EEPROM speichern. RAM speichern entspricht dem Befehl SAVEPROM, denn es werden alle Programme, Parameter und Arrays gesichert.

Speicher > EEPROM löschen

Löschen Sie das EEPROM im LCP wenn Sie entweder die Array-Definition rückgängig machen wollen oder wenn Sie alle Parameter auf Werkseinstellungen zurücksetzen wollen.

ACHTUNG!:
Wenn Sie das EEPROM löschen werden alle Parameter auf die Werkseinstellung zurückgesetzt. Allerdings erst nach dem Ausschalten des FC 300.

ACHTUNG!:
Beachten Sie also Folgendes, wenn Sie das EEPROM löschen:

1. Prüfen Sie, ob Sie alle noch benötigten Programme auf dem PC gesichert haben, um diese nach dem Löschen des EEPROMs wieder in den FC 300 laden zu können.
2. Prüfen Sie, ob Sie die Parameter von allen angeschlossenen FC 300 in einer Datei auf dem PC gesichert haben.
3. Klicken Sie auf *Speicher* → *EEPROM löschen*.
4. Laden Sie die Parameter und die benötigten Programme wieder in die Steuerung(en).

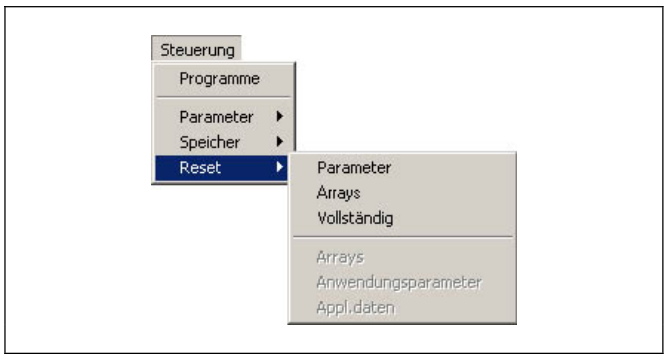
Speicher -> Individuell im LCP-EEPROM speichern

Benutzen Sie die entsprechende Funktion, um → *Anwendungsparameter*, → *Arrays*, → *Alle Applikations-Daten* (diese enthalten zusätzlich zu den Anwendungsparametern und Arrays auch das Applikations-Programm) oder → *Optionsparameter* (MCO 305 Parameter) individuell im LCP-EEPROM zu speichern.

□ Steuerung > Reset Parameter, Arrays oder Vollständig

Mit *Reset* → *Parameter* werden alle globalen Parameter und alle Achsparameter im MCO auf die Werkseinstellungen zurückgesetzt

Mit *Reset* → *Arrays* können Sie alle Arrays im RAM löschen ohne dabei auch die Parameter etc. zu löschen. Dieser Menü-Befehl bewirkt das Gleiche, wie der Befehl DELETE ARRAYS.



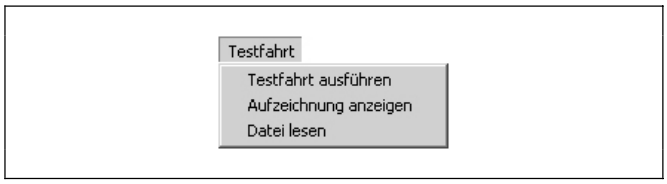
ACHTUNG!:
Wenn Sie anschließend ein SAVE ARRAYS durchführen, werden auch die Arrays im EEPROM überschrieben!

Mit *Reset* → *Vollständig* werden nicht nur die Parameter, sondern auch die Programme und Arrays gelöscht und die MCO 305 Option auf die Werkseinstellungen zurückgesetzt …

ACHTUNG!:
… und zwar sofort und nicht erst nach dem Aus- und Wiedereinschalten der Steuerung wie bei EEPROM löschen.

□ Menü Testfahrt

Das Menü Testfahrt bietet die Funktion → *Testfahrt ausführen* von der Eingabe der Testfahrt-Parameter bis zur grafischen Darstellung der Testfahrt-Ergebnisse.



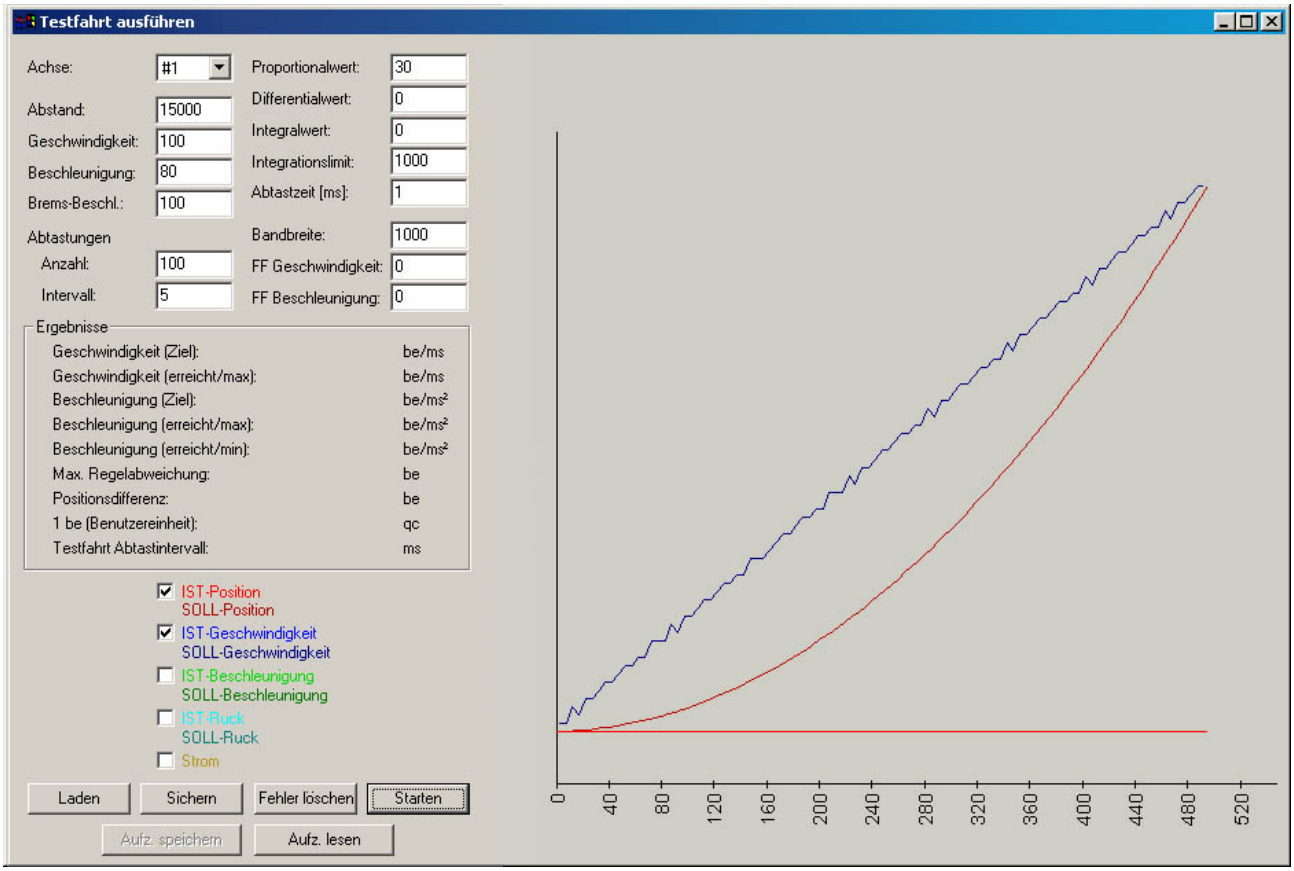
Wenn Sie mit TESTSETP eine Testfahrt mit anderen Parametern definiert haben, können Sie sich diese Ergebnisse nach der Durchführung (TESTSTART) mit *Testfahrt → Aufzeichnung anzeigen* ebenfalls grafisch darstellen lassen.

Entscheidend für eine erfolgreiche Einstellung der Reglerparameter ist die richtige Wahl der Testparameter, wobei verschiedene Punkte beachtet werden sollten

□ Testfahrt-Parameter festlegen

Klicken Sie auf → *Testfahrt ausführen* und geben Sie im Dialogfeld die Testfahrt-Parameter ein:

Verändern Sie nach Möglichkeit von einer Messung zur anderen immer nur einen Parameter und prüfen Sie die Auswirkung.



Abstand

Bestimmen Sie den Fahrweg in Benutzereinheiten und nutzen Sie die gesamte Aufzeichnungsdauer bestmöglich:

Die Zahl der Messpunkte multipliziert mit der Zeitdifferenz zwischen zwei Messungen ergibt die gesamte Aufzeichnungsdauer und bestimmt somit die grafische Darstellung. Für eine möglichst optimale Auswertung der Diagramme sollte der Verfahrweg so gewählt werden, dass die Endposition ungefähr in 80 % der gesamten Aufzeichnungsdauer erreicht wird. So sind auch Überschwinger in der Zielposition noch gut erkennbar.

Beispiel: 50 Messungen in 30 ms Intervallen = 1,5 s Aufzeichnungsdauer

Geschwindigkeit, Beschleunigung und Bremsbeschleunigung

Die Testfahrt-Parameter Geschwindigkeit, Beschleunigung und Brems-Beschleunigung (Verzögerung) werden in Prozent der jeweiligen Maximalwerte eingegeben.

Führen Sie die Messungen mit den für die Steuerung am häufigsten benötigten Werten für Geschwindigkeit, Beschleunigung und Verzögerung durch.

Um das Überschwingungsverhalten bei Erreichen der Endgeschwindigkeit bewerten zu können, ist ein trapezförmiger Geschwindigkeitsverlauf anzustreben. Eventuell muss hierzu das Abtastintervall erhöht oder die Endgeschwindigkeit reduziert werden.

ACHTUNG!:
Prüfen Sie – bevor Sie mit der Optimierung des Regelverhaltens beginnen – ob die Maximalgeschwindigkeit und Maximalbeschleunigung erreicht wird.

Anzahl der Abtastungen

Die Anzahl der Abtastungen sowie das Abtastintervall bestimmen die gesamte Messdauer. 50 bis 100 Messpunkte sind für eine optimale Bildschirmdarstellung ausreichend.

Die Zahl der maximal möglichen Messpunkte ist durch den internen MCO 305 Speicher sowie durch eventuell darin abgelegte Programme begrenzt. Wenn der Speicherplatz für die gewünschten Abtastungen nicht ausreicht, müssen Sie zuvor die in der Positioniersteuerung abgelegten Programme mit *Steuerung → Programme → Alle löschen.*

Abtastintervall

Wählen Sie ein dem System angepasstes Abtastintervall, für Frequenzumrichter z.B. 20 bis 30 ms.

Für dynamische Anwendungen kann das Abtastintervall bis zu 1 ms verkleinert werden. Ein möglichst kurzes Abtastintervall ist für Servomotoren einzustellen.

Für das Aufzeichnen langsamer oder sehr langer Bewegungsvorgänge kann die Zeitdifferenz in Millisekunden natürlich erhöht werden, bis maximal 255 Millisekunden.

ACHTUNG!:
Dieses Abtastintervall ist das interne Intervall zwischen den einzelnen Messungen, nicht das Abtastintervall der Steuerung.

□ Testfahrt ausführen

Beachten Sie bitte die Sicherheitshinweise bevor Sie starten!

Bringen Sie den Antrieb in die Ausgangsstellung, und zwar unbedingt bevor Sie das Testfenster öffnen.

ACHTUNG!:
Bei der Regleroptimierung mit der Funktion *Testfahrt* wird der Antrieb nach dem Erreichen der Zielposition automatisch auf die Startposition zurückgefahren.

Falls Ihr Antrieb nicht reversieren darf, muss der Par. 32-68 *Reversierungsverhalten Slave* auf „Reversieren gesperrt = [2]" eingestellt werden. Klicken Sie auf *Steuerung → Parameter → Achsen* und ändern Sie Einstellung in der Parametergruppe *PID-Regelung*.

ACHTUNG!:
Achten Sie darauf, dass eventuell vorhandene Bremsen gelöst sind und keine Hindernisse innerhalb des Positionierweges sind.

Falsch eingestellte Reglerparameter können den Motor sowie die Mechanik beschädigen. Eine Regleroptimierung darf deshalb nie ohne installierten NOT-AUS-Schalter durchgeführt werden.

Testfahrt Starten

Klicken Sie auf *Testfahrt* → *Testfahrt ausführen* und geben Sie im Dialogfeld die Testfahrt-Parameter ein: Die zuletzt benutzten Testfahrt-Parameter und die aktuellen Achsparameter sind hier bereits eingetragen.

Beginnen Sie die Testreihe mit „stabilen" Reglerparametern. Wenn die standardgemäß vorgegebenen Reglerparameter bereits in der Ausgangsstellung zu einem stark schwingenden Antrieb führen, wählen Sie die Parameter 32-60 *Proportionalfaktor* und 32-61 *Differentialwert* klein (ca. 20) und setzen den Par. 32-62 *Integralfaktor* auf Null. Von diesen Werten ausgehend optimieren Sie dann die Steuerung.

Wenn die Möglichkeit besteht, sollten Sie die Steuerung zunächst mit Motor und Getriebe so optimieren, dass Sie einigermaßen unkritische Werte erhalten. Dann schließen Sie die Mechanik mit der Last an und führen die Feinoptimierung durch.

Klicken Sie auf → *Starten*: Die Testfahrt wird durchgeführt, die aktuellen Positionswerte etc. gespeichert und am Ende der Testfahrt an den PC zur Auswertung übergeben.

ACHTUNG!:
Beobachten Sie das Motorverhalten und die Motortemperatur: Bei starkem Schwingungsverhalten oder übermäßiger Erwärmung des Motors muss der Bewegungsvorgang mit NOT-AUS vorzeitig abgebrochen und die Reglerparameter müssen anders gewählt werden.

Nach einem NOT-AUS müssen Sie den Antrieb für weitere Testfahrten erst wieder in die Ausgangsposition bringen. Reduzieren Sie den Par. 32-60 *Proportionalfaktor* und ggf. zusätzlich den Par. 32-61 *Differentialwert*, bevor Sie die nächste Testfahrt bzw. Messung starten.

Nach Beendigung der Messung werden die Messdaten automatisch an den PC übertragen und der Antrieb kehrt währenddessen mit reduzierter Geschwindigkeit in die Ausgangsposition zurück. Die Diagramme werden automatisch dargestellt.

Fehler löschen

Damit löschen Sie alle aktuell anliegenden Fehler in der Motorsteuerung. Falls Fehler anliegen kann keine *Testfahrt* gestartet werden; die Auswahl *Start* ist daher grau

Testfahrt-Parameter sichern und laden

Klicken Sie auf → *Sichern*, um die Testfahrt-Parameter sowohl im FC 300 als auch auf der Festplatte des PCs zu speichern. Wenn Sie nun bei den weiteren Testfahrten schlechtere Regelungsergebnisse erzielen, können Sie die gespeicherten Parameter wieder → *Laden*.

Aufzeichnung speichern und lesen

Benutzen Sie diese Funktion um die Testfahrtergebnisse als ASCII Textdatei zu speichern. Dies ermöglicht vorhergehende gespeicherte Testfahrt-Ergebnisse wieder einzulesen und darzustellen.

□ Aufzeichnung anzeigen

Wenn Sie mit TESTSETP eine Testfahrt mit anderen Parametern definiert haben, können Sie das Ergebnis nach der Durchführung (TESTSTART) mit *Testfahrt* → *Aufzeichnung anzeigen* auch grafisch darstellen. (Soweit dies bei den von Ihnen gewählten Parametern sinnvoll ist, das heißt dass sich die Ergebnisse mit den vier Grafiken auch darstellen lassen.) Diese vier Grafiken bzw. sieben Kurven werden dazu wie folgt genutzt:

(1) Die Kurve Istposition zeigt die Werte des Index w1 (siehe TESTSETP),
(2) die Kurve Sollposition die Werte des Index w2,
(3) und die Stromkurve die Werte des Index w3.
(4) Die Kurve Ist-Geschwindigkeit zeigt die Differenz der Aufzeichnungswerte zu den Werten von w1. Im Fall der Aufzeichnung von Positionsdaten also die Änderung der Position in ms, das ist die Geschwindigkeit.
(5) Die Kurve Soll-Geschwindigkeit zeigt die Differenz der Aufzeichnungswerte zu w2, im Fall der Aufzeichnung von Positionsdaten also die Änderung der Position in ms = Geschwindigkeit.
(6) Die Kurve Ist-Beschleunigung zeigt die Differenz der Aufzeichnungswerte zur Ist-Geschwindigkeit (s. 4), im Fall der Aufzeichnung von Positionsdaten also die Änderung der Geschwindigkeit in ms = Beschleunigung.
(7) Die Kurve Soll-Beschleunigung zeigt die Differenz der Aufzeichnungswerte zur Soll-Geschwindigkeit (siehe 5), im Fall der Aufzeichnung von Positionsdaten also die Änderung der Geschwindigkeit in ms = Beschleunigung.

□ Fahrdiagramme auswerten

Prüfen Sie die maximale Position, die maximale Geschwindigkeit, die Anzahl der „Überschwinger", sowie die Dauer des Einschwingvorganges.

Zu jeder Grafik werden die wichtigsten Testfahrt-Parameter, die eingestellten Maximalwerte und die tatsächlich erreichten Werte angezeigt:

- Geschwindigkeit in Benutzereinheiten/ms),
- Beschleunigung in Benutzereinheiten/ms²),
- die maximale Soll-Ist-Abweichung (Regelabweichung in Benutzereinheiten), die während der Fahrt herrschte,
- die tatsächliche Positionsabweichung im Ziel,
- Benutzereinheit in qc und das
- Datenabtastintervall in ms.

Klicken Sie in das jeweilige Kontrollkästchen um eine der anderen Grafiken zu sehen, zum Beispiel Geschwindigkeit. Sie können auch zwei oder alle vier Grafiken gleichzeitig darstellen. Dann werden allerdings die Einheiten auf der x-Achse ausgeblendet.

Testfahrt-Grafiken

Die Positionsgrafik zeigt mit der braunen (dunkleren) Kurve die Sollpositionen und mit der roten (helleren) Kurve die tatsächlich erreichten Positionen.

Testfahrt-Grafik Geschwindigkeit: Die gelbe (helle) Kurve zeigt den aufgezeichneten Geschwindigkeitsverlauf, die braune (dunkle) Kurve die gewünschte trapezförmige Sollkurve.

In Sonderfällen kann der trapezförmige Geschwindigkeitsverlauf zu einer Dreiecksform entartet sein. Dieser Effekt tritt auf, wenn die Positionierdistanz zu gering ist, um bei der gewünschten Beschleunigung die Maximalgeschwindigkeit zu erreichen.

Testfahrt-Grafik Beschleunigung: Die hellgrüne Kurve zeigt den aktuellen Beschleunigungsverlauf, die dunkle die gewünschte treppenförmige Sollkurve beim Beschleunigen und Abbremsen.

Testfahrt-Grafik Strom: Die blaue Linie zeigt die aktuelle Stromaufnahme des Motors.

□ Menü CAM-Editor

Mit dem *CAM-Editor* werden die Kurvenprofile für beliebige Kurvenscheibensteuerungen erstellt. Die einzelnen Kurven werden durch Fixpunkte, Parameter für die Ein- und Auskuppelbewegung sowie Parameter für die Synchronisation mit Marker definiert. Für diese Eingabe und für andere Kurven-Informationen gibt es im CAM-Editor *Registerkarten*. In der Grafik werden die Kurven und Parameter visualisiert; außerdem können Sie die Fixpunkte auch interaktiv eingeben und manipulieren.

□ CAM-Editor starten

Sie können Konfigurations-Dateien (*.CNF) mit MCT 10 öffnen. Dies startet auch den APOSS CAM-Editor.

Die Datei muss mindestens eine Kurve enthalten. Falls die Datei keine Kurve enthält wird der Anwender aufgefordert eine Kurve zur Datei hinzuzufügen. Umgekehrt kann der Anwender die letzte Kurve einer Datei nicht löschen.

Mit dem Schließen des CAM-Editors wird auch APOSS beendet und zum MCT 10 zurückgekehrt.

Die Schaltflächen „Neue CNF", „Lade CNF" und „Sichern CNF als" sind deaktiviert, weil die Handhabung der Dateien mit MCT 10 durchgeführt wird.

Bevor Sie beginnen eine Kurve zu editieren

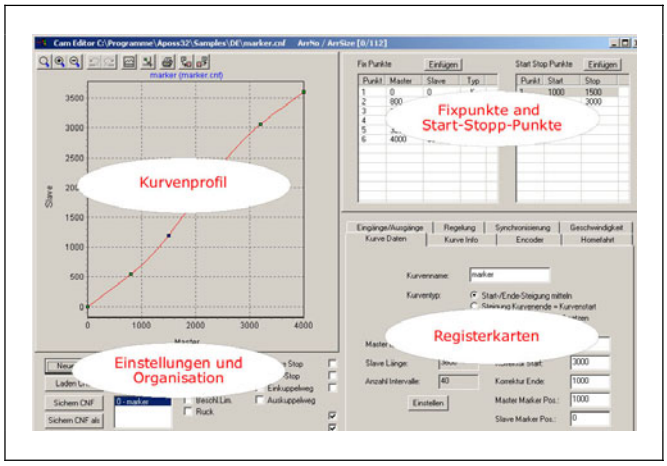
… sollten Sie mit MCT 10 die Parameter der Steuerung als CNF-Datei in den CAM-Editor laden. Denn dann werden die Parameter in der entsprechenden *Registerkarte* schon passend ausgefüllt. Dabei werden auch evtl. vorhandene Arrays mit ausgegeben.

Wenn Sie keine CNF-Datei laden, werden die Werkseinstellungen des FC 300 eingestellt.

□ CAM-Editor Fenster

Das CAM-Editor-Fenster ist in vier Bereiche aufgeteilt:

- Kurvenprofil-Grafik mit einer Symbolleiste.
- Bereich zur Einstellung der verschiedenen Darstellungen des Kurvenprofils und Funktionen zum Organisieren der CNF-Datei.
- Tabelle der *Fixpunkte* und der *Start- und Stopp-Punkte.*
- Registerkarten: *Kurven-Daten, Kurven-Info* und alle Parameter gemäß den Dialogfeldern des APOSS-Programms: *Encoder, Homefahrt, Eingänge/Ausgänge, Regelung, Synchronisation* und *Geschwindigkeit.*



Die Titelleiste zeigt den Namen der CNF-Datei und den vollständigen Pfad. Rechts daneben steht die Nummer (der Reihenfolge) des Arrays in der CNF-Datei und die Anzahl der Array-Elemente, die Sie für die DIM-Anweisung im APOSS-Programm benötigen, zum Beispiel „Arr.Nr/ArrSize [0/112]".

Das gesamte *CAM-Editor*-Fenster kann wie üblich beliebig vergrößert oder verkleinert werden.

Zum Schließen des *CAM-Editor*-Fensters klicken Sie in das *Schließen*-Feld rechts oben.

□ Einstellungen und Organisation

Dieser Bereich des CAM-Editors enthält Kontrollkästchen zum Aktivieren und Deaktivieren der verschiedenen Darstellungen des Kurvenprofils sowie Schaltflächen für die Organisation der CNF-Dateien. Eine detaillierte Beschreibung finden Sie in der Online-Hilfe.

Online-Berechnung, Auto-Skalierung und Ausrichten an Gitter

Online-Berechnung: Wenn Online-Berechnung aktiviert ist, wird das Kurvenprofil kontinuierlich berechnet und dargestellt, und die Fixpunkte werden im Kurvenprofil nachgezogen.

Auto-Skalierung: Wenn Auto-Skalierung aktiviert ist wird das Diagramm automatisch so skaliert, dass immer die gesamte Kurve zu sehen ist.
ANMERKUNG: Wenn das Diagramm manuell gezoomt oder gescrollt wurde, wird die Funktion automatisch deaktiviert, damit der so auswählte Bereich sichtbar bleibt.

Ausrichten an Gitter: Wenn diese Funktion aktiviert ist, rasten die Fixpunkte am nächsten Interpolations-Gitterpunkt ein, sobald sie bewegt werden. Es empfiehlt sich diese Funktion immer zu aktivieren, denn Fixpunkte die nicht auf Interpolationspunkte fallen, liegen nicht im Kurvenprofil.

ANMERKUNG: Der letzte Fixpunkt rastet immer auf dem Interpolations-Gitter ein, unbeachtet ob die Funktion aktiviert ist oder nicht. So kann eine exakte Beziehung zwischen der Master-Intervall-Länge und der Anzahl der Interpolationspunkte sichergestellt werden.

CNF Dateien organisieren: Online-Modus

Benutzen Sie die Funktionen des MCT 10 Motion Control Tools für *Neue*, *Laden* und *Sichern CNF als*.

Zum Antrieb schreiben: Klicken Sie auf → *Zum Antrieb schreiben,* um die neuen CNF-Werte (insbesondere die CAM-Arrays) in den Antrieb zu downloaden. Die neuen Werte werden auch im MCT 10 Datenbestand gespeichert.

CNF Dateien organisieren: Offline-Modus

Benutzen Sie die Funktionen des MCT 10 Motion Control Tools für *Neue CNF*, *Laden* und *Sichern CNF als*.

Sichern CNF: Die aktuellen Daten werden als CNF-Datei gespeichert und überschreiben dabei eine frühere Version der Datei (d.h. speichert diese zurück in den MCT 10 Datenbestand).

□ Kurvenprofil

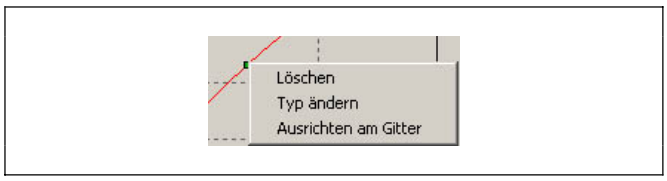
Das Diagramm zeigt eine grafische Darstellung der Kurve, der Parameter und anderer Informationen des Kurvenprofils. Hier können Sie interaktiv mit der Maus die Fixpunkte ändern oder eingeben und das Diagramm vergrößern. Oben links bietet eine Werkzeugleiste weitere kurven-relevante Funktionen.

Eine blaue Überschrift nennt den Dateinamen und die dargestellte Kurve.

Fixpunkte können mit der Maus eingegeben, gelöscht oder interaktiv versetzt werden, entweder durch Doppelklick an der gewünschten Position oder mittels des Kontext-Menüs, das mit der rechten Maustaste geöffnet wird. Bitte benutzen Sie die Online-Hilfe für die Details.

Punktetyp in der Grafik ändern

Im Kurvenprofil werden die Kurvenpunkte in grünen und die Tangentenpunkte in blauen Punkten dargestellt. Bewegen Sie den Mauscursor auf einen Fixpunkt bis das Handsymbol erscheint und klicken Sie auf die rechte Maustaste. Wählen Sie dann im Kontext-Menü die gewünschte Aktion aus:



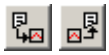
Wenn der ausgewählte Punkt ein Kurvenpunkt ist, werden beide, dieser und der Punkt davor (nach links) zu Tangentenpunkten geändert. Wenn der ausgewählte ein Tangentenpunkt ist, werden beide, dieser und der Punkt am anderen Ende des Tangentensegmentes in Kurvenpunkte geändert.

Kurvenprofil skalieren (Zoom) oder scrollen

Diese Funktionen folgen der Windows-Spezifikation; die Details finden Sie die Online-Hilfe.

Werkzeugleiste Kurvenprofil

Das Kurvenprofil-Fenster enthält folgende Werkzeugleiste: Von links nach rechts:

 Original-Auflösung wiederherstellen,  Vergrößern / Verkleinern,
 Rückgängig / Wiederherstellen,  Grafikanzeige maximieren/Minimieren,
 Fixpunkte am Raster ausrichten,  Kurve drucken,  Import / Export ASCII.
(Mehr Details finden Sie die Online-Hilfe.)

□ Fixpunkte benutzen

Fixpunkte werden benutzt, um die grundlegende Form der Kurve über eine Master-Intervall-Länge festzulegen. Der CAM-Editor errechnet daraus eine mathematische Spline-Funktion die durch diese Fixpunkte führt. Dieser Spline wird durch viele kleine gerade Linien annähernd erreicht; es sind diese Liniensegmente, die letztlich die Steuerung benutzen wird. Die Liniensegmente werden durch Einsatz eines gleichmäßigen Interpolationsrasters erzeugt. Dieses bestimmt der Anwender durch die Anzahl der Interpolationspunkte.

Das Master-Intervall muss exakt mit dem Interpolationsgitter übereinstimmen, so dass ganzzahlige Interpolations-Intervalle benutzt werden können.

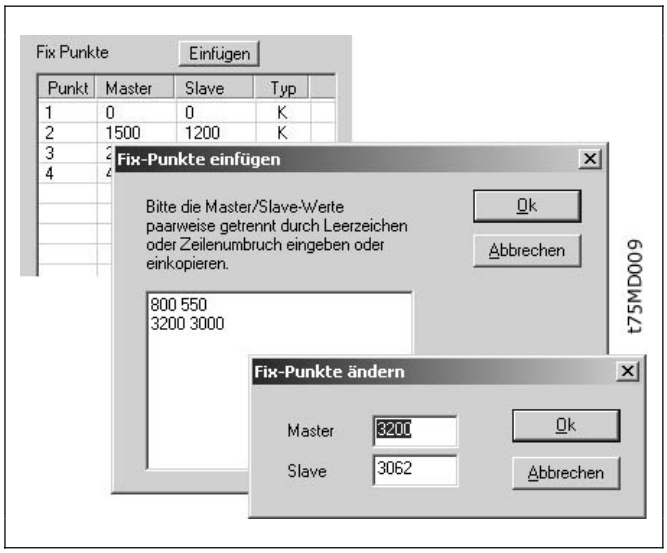
Daher muss die Master-Länge ein Vielfaches der Anzahl der Interpolations-Intervalle sein. Andernfalls können numerische Fehler ein unerwartetes Verhalten verursachen. Es wird empfohlen, dass die Master-Länge mindestens ein vierstelliger Wert ist

Die Anzahl der Interpolations-Intervalle muss groß genug sein, damit die Kurve mit einer vernünftigen Genauigkeit angenähert werden kann. Allerdings darf sie nicht so groß sein, dass die Steuerung wegen zu vielen kleinen Segmenten Leistungsprobleme bekommen könnte. Die Intervall-Dauer (siehe Registerkarte *Kurven Info*) sollte nicht kleiner als 20-30 ms sein.

Eine neue Kurve wird zunächst mit nur zwei Fixpunkten erzeugt. Weitere Fixpunkte können auf verschiedene Art und Weise hinzugefügt werden: Entweder in der Tabelle der Fixpunkte, oder interaktiv mit der Maus im Kurvenprofil. Fixpunkte sollten immer auf dem Interpolationsgitter liegen, daher sollten Sie wenn möglich immer → *Ausrichten am Gitter* aktivieren.

Fixpunkte in der Tabelle einfügen, löschen oder ändern

Klicken Sie in der Tabelle auf → *Einfügen* und geben Sie im folgenden Dialogfenster die Fixpunkte paarweise ein, also je einen Wert für den Master und den Slave. Sie können gleich mehrere Paare eingeben, auch in beliebiger Reihenfolge. Diese werden automatisch geordnet, sobald sie mit → *OK* zur Kurve hinzugefügt werden.



Wenn → *Ausrichten am Gitter* aktiviert ist, werden die Werte für den Master am Interpolations-Raster ausgerichtet.

Fixpunkte können in der Tabelle gelöscht werden, indem Sie einen Fixpunkt in der Tabelle markieren und die [Entf]-Taste drücken.

Durch Doppelklick auf den gewünschten Punkt in der Tabelle können Fixpunkte geändert werden. Dazu werden entsprechende Dialogfenster angeboten.

Typ: Kurven- und Tangentenpunkt

Die Kurve wird als Spline-Interpolation zwischen den Kurvenpunkten berechnet. Für Bereiche, in denen die Geschwindigkeit konstant und die Beschleunigung 0 sein muss, benutzen Sie Tangentenpunkte. Zwischen diesen Punkten wird statt eines Splines eine Gerade gelegt.

Der CAM-Editor wird immer versuchen, einen fließenden Übergang zwischen einem Segment zum nächsten zu erreichen. Daher ist mindestens ein Kurvenpunkt zwischen zwei benachbarten Tangenten erforderlich. So kann der CAM-Editor einen Spline zwischen den beiden Tangenten legen. Dies ist auch der Grund, warum Tangentenpunkte immer als Paar auftreten.

Um ein Segment einer Kurve in eine Tangente zu ändern, doppelklicken Sie in der Fixpunkte-Tabelle in der Spalte → *Typ* auf den zweiten Fixpunkt der Tangente. Dieser Punkt und der vorhergehende werden daraufhin in Tangentenpunkte geändert. Wenn Sie ein Tangenten-Segment zurück in eine Kurve ändern wollen, doppelklicken Sie erneut in die Spalte Typ auf einen der beiden Tangenten-Fixpunkte.

Das Gleiche können Sie in Grafik ausführen: Sie bewegen den Cursor auf den Punkt bis das Handsymbol erscheint. Dann klicken Sie auf die rechte Maustaste und wählen im Kontext-Menü → *Typ ändern*. Auch hier werden sofort beide Punkte geändert.

□ Start- und Stop-Punkte benutzen

Start-Stop-Punkte werden benutzt, um Punktepaare für das Ein- und Auskuppeln des Slaves bei der Synchronisation zu definieren. Ein Punktepaar benötigen Sie, um zu bestimmen, an welcher Master-Position die Synchronisation starten und wo aufsynchronisiert sein soll. Mit einem weiteren Punktepaar legen Sie fest, ab welchem Punkt ausgekuppelt und wo die Synchronisation angehalten werden soll.

Sie können mehrere (maximal 25) Punktepaare definieren, zum Beispiel um mit mehreren Starts und Stopps in einem Zyklus verschiedene Situationen beim Starten zu berücksichtigen. Mit den Befehlen SYNCCSTART *pnum* und SYNCCSTOP *pnum slavepos* bestimmen Sie in Ihrem Programm, welches Punktepaar benutzt werden soll.

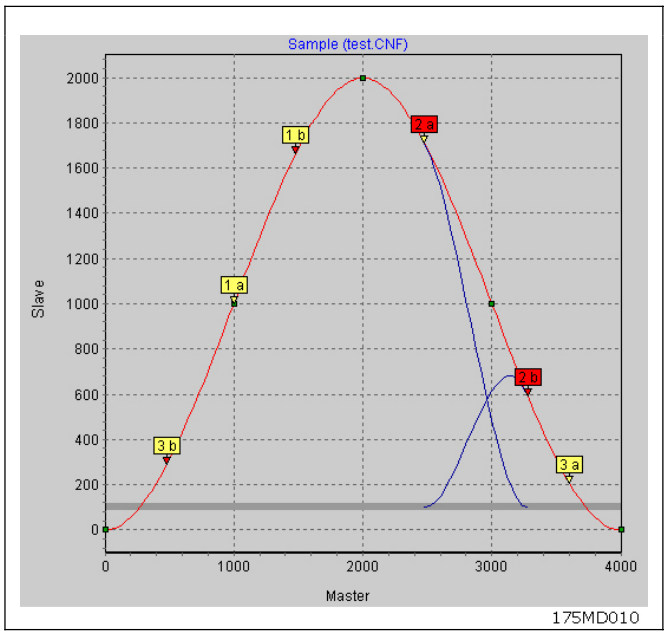
Wenn Start- und Stop-Punkte identisch sind, wird der Slave mit der eingestellten Maximalgeschwindigkeit – also ohne Kurve – eingekuppelt, sobald der Master diesen Punkt erreicht hat.

Wenn der Start-Punkt größer als der Stop-Punkt ist, werden die Wege beim Einkuppeln und Auskuppeln „auf das nächste Master-Intervall" überlaufen.

Wenn keine Start-Stop-Punkte definiert sind, wird der Slave bei SYNCCSTART sofort mit der eingestellten Maximalgeschwindigkeit eingekuppelt.

Die Reihenfolge der Punkte in jedem Paar ist wichtig und wird automatisch bei der Fahrtrichtung berücksichtigt. Beim Vorwärtsfahren startet die Synchronisation am Start-Punkt und wird bis zum Stop-Punkt beendet. Beim Rückwärtsfahren wird sie am Stop-Punkt gestartet und bis zum Start-Punkt beendet.

Wenn das Programm ohne expliziten Befehl SYNCCSTOP *pnum slavepos* verlassen wird, wird immer das zweite Punktepaar für das Auskuppeln benutzt.

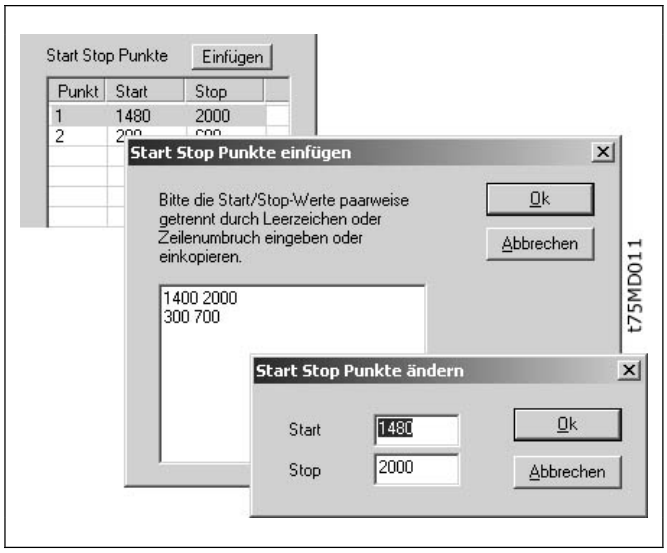


Start-Stop-Punkte hinzufügen, löschen und Verlauf darstellen

Start-Stop-Punkte können eingefügt werden, wenn Sie auf → *Einfügen* in der Start-Stop-Punkte-Tabelle klicken. Im folgenden Dialogfeld geben Sie die Punkte paarweise ein. Dabei können Sie mehrere Paare gleichzeitig eingeben. Auch hier brauchen Sie nicht auf die Reihenfolge achten; Start-Stop-Punktepaare können sogar in der Tabelle in beliebiger Reihenfolge stehen.

Start-Stop-Punkte können Sie löschen, wenn Sie das Paar in der Tabelle markieren und die [Entf]-Taste drücken.

Start-Stop-Punkte können Sie in einem Dialogfenster verändern, das nach einem Doppelklick auf das entsprechende Punktepaar in der Tabelle angeboten wird:



Sie können die Start-Stop-Punkte und die Einkuppel- und Auskuppelwege im Kurvenprofil visualisieren: Aktivieren Sie → *Start-Stop*. Gelbe Flags zeigen das Punktepaar für das Ein- und Auskuppeln während der Synchronisation. Der Startpunkt wird mit „a" und einem gelben Pfeil gekennzeichnet, der Stopppunkt mit „b" und rotem Pfeil. Wenn Sie ein Punktepaar in der Tabelle auswählen, wird dieses rot markiert.

Aktivieren Sie → *Einkuppelweg* oder → *Auskuppelweg* oder beides und markieren Sie das Start-Stop-Punkte-Paar, dessen Weg Sie sehen wollen in der Tabelle.

ACHTUNG!:
Wenn der Master vorwärts fährt, wird das Einkuppeln mit dem Einkuppelweg gezeigt und das Auskuppeln mit dem Auskuppelweg.
Wenn der Master rückwärts fährt, wird das Einkuppeln mit dem Auskuppelweg dargestellt und das Auskuppeln mit dem Einkuppelweg!

□ Registerkarten Kurven-Daten, Kurven-Info und Parameter

Bevor Sie eine Kurve editieren, sollten Sie immer zuerst die Parameter Ihrer Steuerung in den *CAM-Editor* laden. Sie können die Parameter inklusive den Arrays in eine CNF-Datei mit MCT 10 sichern. Öffnen Sie diese Datei.

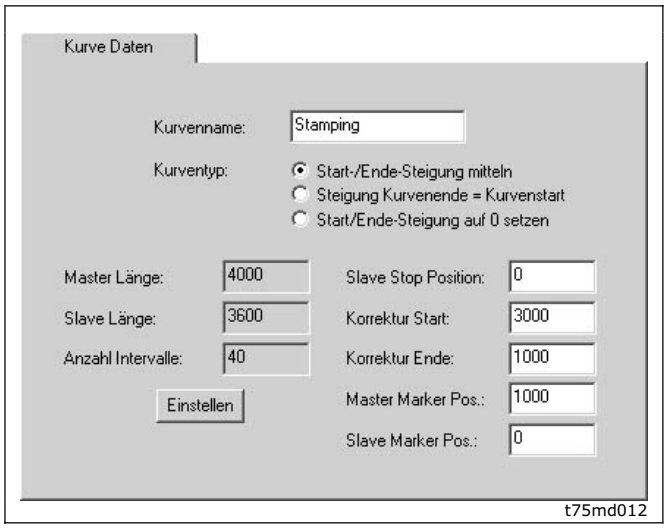
Wenn Sie keine Parameter laden, finden Sie die Werkseinstellungen des FC 300 eingetragen.

Wenn Sie im Verlauf der Kurvenerstellung die Parameter ändern, werden diese mit in der CNF-Datei gespeichert und mit *Parameter* → *Wiederherstellen aus Datei* in die Steuerung geladen und in den entsprechenden Dialogfeldern der Achsparametern eingetragen.

□ Registerkarte Kurven-Daten

In der Registerkarte → *Kurven-Daten* bestimmen Sie wichtige Eckdaten Ihrer Kurve:

Wenn Sie mehrere Kurven editieren, geben Sie hier zu Ihrer eigenen Information den Kurven aussagefähigen → *Kurvennamen*.



Kurventyp

Um Drehzahlsprünge bei mehrmaligem Kurvendurchlauf zu verhindern, können Sie zwischen drei Kurventypen wählen. In jedem Fall berücksichtigt die Interpolation die Steigung der Kurve am Anfang und Ende.

Wählen Sie den Kurventyp aus:

- Die Steigung der Kurve am Anfang und Ende wird gemittelt.
- Die Steigung am Anfang der Kurve wird auch für das Ende der Kurve benutzt.
- Die Steigung am Anfang und Ende der Kurve wird auf 0 gesetzt.

Masterlänge und Anzahl Intervalle definieren

Die Masterlänge und die Anzahl der Intervalle kann auf drei Arten definiert werden:

1. Ändern Sie den letzten Fixpunkt durch Doppelklick in der Tabelle der Fixpunkte.
2. Verschieben Sie den letzten Fixpunkt mit der Maus im Kurvenprofil.
3. Klicken Sie auf → *Einstellen* in der Registerkarte Kurvendaten

Die beiden ersten Methoden ändern die Master-Intervall-Länge direkt. Bei dieser Methode wird die Anzahl der Interpolationspunkte automatisch geändert, so dass das Interpolations-Intervall (der Abstand zwischen den Interpolationspunkten) unverändert bleibt. Der letzte Fixpunkt rastet immer auf dem Interpolationsraster ein, um Nachkommastellen bei den Interpolationspunkten zu vermeiden.

Wenn Sie die Kurvendaten → *Einstellen*, können Sie im Dialogfenster sowohl die Master-Länge, als auch die Anzahl der Intervalle einstellen. Wenn Sie die Änderung auf diese Weise durchführen, wird das Interpolations-Intervall neu berechnet.

Wählen Sie nicht zu kleine Intervalle (das würde nur einen unnötigen Overhead verursachen) und möglichst als ganzzahligen Teiler der Masterlänge. Zum Beispiel bei einer Masterlänge von 3000 ein Intervall von 30 oder 60. Der letzte Fixpunkt kann nicht auf eine Position bewegt werden, die

- vor dem vorhergehenden Fixpunkt liegt,
- vor einem Start-Stop-Punkt liegt,
- entweder vor dem Korrektur Start oder Korrektur Ende liegt oder
- vor der Master-Marker-Position liegt.

Falls es notwendig ist, den letzten Fixpunkt außerhalb dieser Grenzen zu schieben, dann müssen zuerst die begrenzenden Punkte bzw. Positionen geändert werden. Danach kann der letzte Fixpunkt verschoben werden.

Slave-Stop-Position

Bestimmen Sie die Position, zu der der Slave fahren und stoppen soll, wenn im Programm kein SYNCCSTOP *pnum slavepos* Befehl mit der Variablen *slavepos* gesetzt wurde.

Diese Position wird auch verwendet, wenn SYNCC mit einer bestimmten Anzahl Zyklen startet und keinen SYNCCSTOP Befehl benutzt.

Eine graue Linie zeigt diese Position im Kurvenprofil. Aktivieren Sie dazu → ☑ *Slave-Stop-Position*.

Korrektur Start / Ende

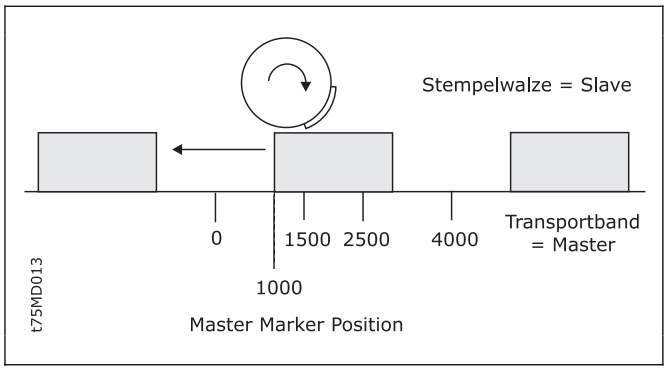
Geben Sie die Master-Positionen ein, bei welcher die Markerkorrektur beginnen und bei welcher sie enden soll. Achten Sie darauf, dass genügend Zeit bleibt, die Synchronisation zu korrigieren, bevor der Bearbeitungspunkt erreicht wird.

Der Korrekturbereich wird im Kurvenprofil blau dargestellt. Aktivieren Sie dazu → ☑ *Korrektur*.

Master-Marker Position und Slave-Marker Position

Tragen Sie die Master-Position (bzw. bei einer Slave-Synchronisation mit Marker die Slave-Position) ein, für die der Marker eingerichtet wurde, hier zum Beispiel der Anfang eines Kartons.

Aus der Master-Markerposition und dem Markerabstand wird die Position der Kurve errechnet, bei der der Marker erkannt wird. Diese Position wird im Kurvenprofil als grüne Linie dargestellt und ermöglicht Ihnen die Festlegung des Korrekturbereiches. Aktivieren Sie dazu → ☑ *Master-Marker* bzw. *Slave-Marker*.



Master Länge und Slave Länge

Information über die in der Tabelle der Fixpunkte festgelegte Zykluslänge des Masters bzw. des Slaves.

ACHTUNG!:
Die Slave-Länge muss positiv sein; dies kann man durch Definition der *Drehrichtung* in Par. 32-10 sicherstellen.

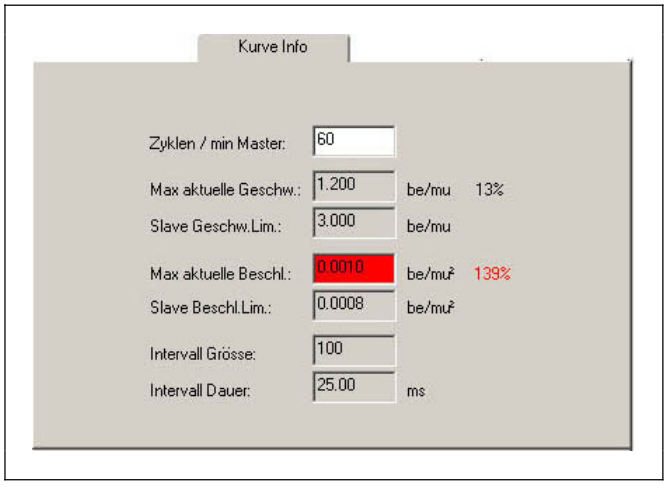
☐ Registerkarte Kurven-Info

In dieser Registerkarte bestimmen Sie im Eingabefeld die Anzahl der → *Zyklen / min Master.* In den anderen Feldern finden Sie Kurven-Informationen, die sich aus den Parametern und der Kurvenanwendung berechnen.

Falls die Geschwindigkeit oder Beschleunigung das Limit erreicht, wird das Feld rot markiert, wie im Beispiel gezeigt.

Sie können den Verlauf der Geschwindigkeit und Beschleunigung im Kurvenprofil grafisch darstellen. Aktivieren Sie dazu das entsprechende Kontrollkästchen, zum Beispiel → ☑ *Velocity.*

Mehr Details finden Sie in der Online-Hilfe.



Zyklen / min Master

Geben Sie die Anzahl der Zyklen des Masters pro Minute ein. In den meisten Fällen ist dies die Anzahl der Produkte, die (maximal) pro Minute verarbeitet werden.

Intervall-Größe und Intervall-Dauer (ms)

Die Intervall-Größe ergibt sich aus der Anzahl der Intervalle pro Masterzykluslänge.

Die Zeit in (ms) für ein Intervall ergibt sich ebenfalls aus der Anzahl der Intervalle pro Masterzykluslänge. Sie sollte nicht kleiner als 30 ms sein. (30 bis 100 ms sind geeignete Werte.) Verändern Sie also in den *Kurven-Daten* die → *Anzahl Intervalle*, damit Sie eine vernünftige Größe erhalten.

□ Registerkarten Parameter

Dies betrifft die Parameter für *Encoder, Homefahrt, Eingänge/Ausgänge, Regelung, Synchronisation* und *Geschwindigkeit*.

Im Folgenden werden nur die Besonderheiten der beiden letztgenannten Registerkarten erläutert. Informieren Sie sich bitte im Kapitel Parameter-Referenz über Inhalt, Einheiten, Eingabebereiche und Werkseinstellung der Parameter oder stellen Sie den Cursor in ein Eingabefeld und tasten [F1].

Registerkarte Synchronisation

Syncfaktor Master und Slave

Die beiden Parameter 33-10 *Syncfaktor Master* und 33-11 *Syncfactor Slave* werden benutzt, um bei der Kurvenscheibensteuerung die MU-Einheiten zu bestimmen.

Marker Abstand

Tragen Sie hier den Abstand des Sensors zum Bearbeitungspunkt ein; bei Master-Marker in Par. 33-17 und bei Slave-Marker in Par. 33-18.

Aus der Master-Markerposition und dem Markerabstand wird die Position der Kurve errechnet, bei der der Marker erkannt wird. Diese Position wird im Kurvenprofil als grüne Linie dargestellt und ermöglicht Ihnen die Festlegung des Korrekturbereiches. Aktivieren Sie dazu → ☑ *Master-Marker* bzw. *Slave-Marker*.

Toleranz

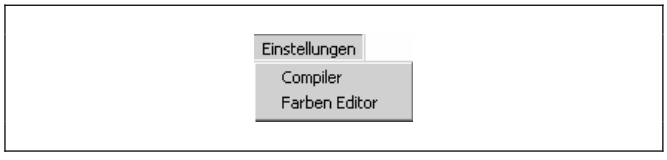
Toleranzfenster für das Auftreten der Master-Marker bzw. der Slave-Marker (Parameter 33-21 und 33-22). Das Toleranzfenster wird im Kurvenprofil als grüner Bereich dargestellt. Aktivieren Sie dazu → ☑ *Master-Marker* bzw. *Slave-Marker.*

Registerkarte Geschwindigkeit

Die in der aktuellen Anwendung erreichte maximale Geschwindigkeit und Beschleunigung wird in qc/Abtastzeit berechnet. Die Darstellung im Kurvenprofil erfolgt in Einheiten. Aktivieren Sie dazu → ☑ *Geschwindigkeit* bzw. *Beschleunigung*.

□ Menü Einstellungen

Dieses Menü bietet abhängig vom Betriebsmodus verschiedene Optionen und Einstellungen.



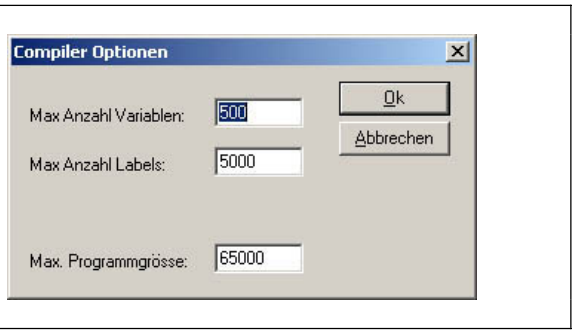
□ Compiler

Die Default-Werte für die Compiler Optionen sind für die meisten Anwendungen passend gesetzt: So belegen sie nicht zu viel Speicherplatz und ermöglichen die notwendigen Eingaben.

Max. Anzahl Variablen

Die Anzahl der Variablen hat direkten Einfluss auf die Menge des Speicherplatzes, der in der Steuerung reserviert wird. Dabei ist zu beachten, dass ein Array zusätzlich den Platz einer Variablen belegt.

Wenn Sie mehr als 92 Variablen (inkl. Arrays) benötigen, erhöhen Sie hier die Anzahl.



Max. Anzahl Labels

Die Maximale Anzahl der Labels bestimmt, wie viel Speicherplatz während der Kompilierung für interne Sprungmarken zur Verfügung gestellt wird. Interne Sprungmarken werden für alle Programmverzweigungen (GOTO, IF, LOOP, REPEAT, WHILE, GOSUB) während der Kompilierung automatisch erzeugt. Der empfohlene Einstellbereich liegt zwischen 100 und 500 interner Labels.

Erhöhen Sie die maximal zulässigen Werte, wenn die Anzahl der Labels zum Beispiel für Texteingaben nicht genügt.

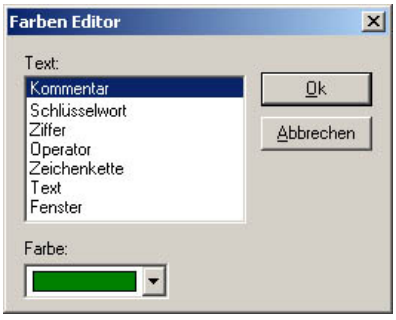
Max. Programmgröße

Die maximale Programmgröße definiert die maximale Größe eines Programms in Bytes nachdem es kompiliert und fertig zum Downloaden in die Steuerung ist.

Beachten Sie, dass einige ältere Steuerungen Programme größer 65.000 Bytes nicht unterstützen.

□ Farben Editor

Zur besseren Übersicht können den verschiedenen Programmteilen wie Kommentar, Schlüsselwort, Ziffer usw. unterschiedliche Farben zugeordnet werden. Markieren Sie den Typ, z.B. Kommentar und wählen Sie dazu die gewünschte Farbe. Mit → *OK* speichern Sie die neuen Einstellungen.



□ Menü Fenster und Hilfe

Die Funktionen dieses Menüs verhalten sich Windows-konform, z.B. → *Vertikal* (nebeneinander) oder *Horizontal* (untereinander).

Menü Hilfe

Die Darstellung und Funktionalität der Online-Hilfe unterscheidet sich in Abhängigkeit vom eingesetzten Betriebssystem, aber sie verhält sich in allen Systemen gemäß der Standard-Spezifikation, zum Beispiel enthält → *Inhalt* eine Volltextsuche.

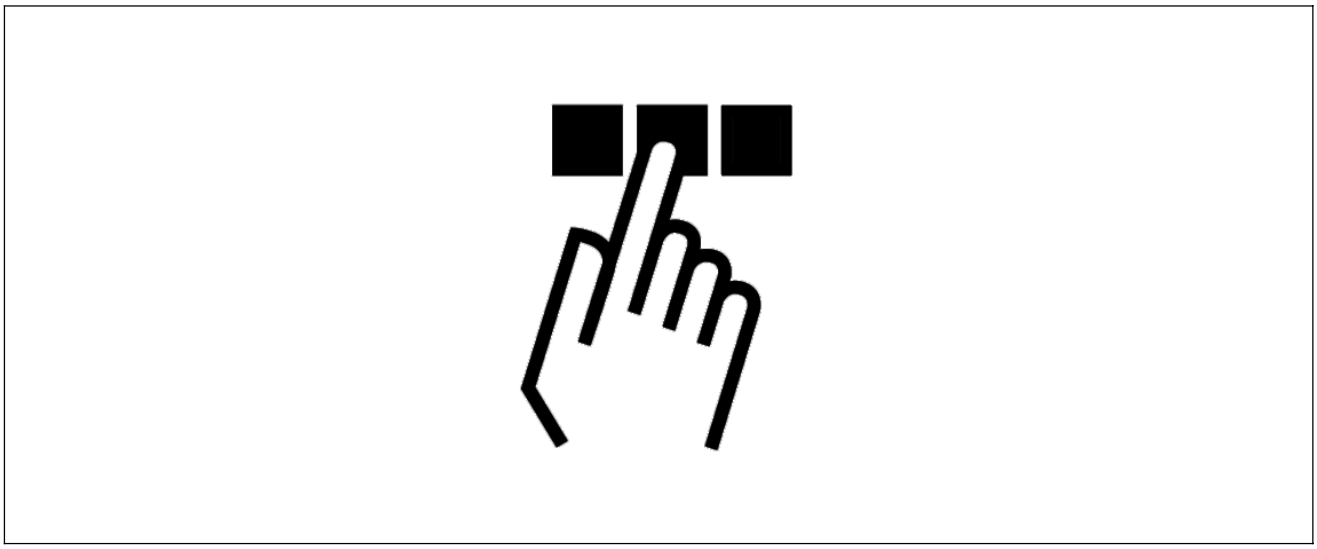
Kontextsensitive Hilfe

Die *Befehlshilfe* und alle Parameter-Dialogfelder im Menü *Steuerung* sowie die Registerkarten im CAM-Editor bieten einen direkten Zugang zur Online-Hilfe. Markieren Sie den Befehl in der → *Befehlshilfe* oder stellen Sie den Mauscursor in das Eingabefeld eines Parameters und drücken Sie [F1]. Sie erhalten dann den entsprechenden Abschnitt der Hilfe direkt angezeigt.

Programminfo

Hier finden Sie die Versionsnummern des APOSS-Programms, der Programmbibliothek und des Compilers.

Programmieren



◻ MCO mit der APOSS Makrosprache programmieren

Die folgenden Kapitel beschreiben, wie Sie den FC 300 mit MCO 305 mit APOSS programmieren. Anfänger lesen bitte die grundsätzlichen Erläuterungen zur Programmiersprache APOSS, also Programmaufbau, Befehlsstruktur, Interrupt, Sprachelemente, Arithmetik und Benutzereinheit. Erfahrene Programmierer informieren sich bitte über die APOSS-spezifischen Grundlagen, zum Beispiel Benutzereinheiten oder Parameter.

Die Funktionen des Menüs *Bearbeiten* werden benutzt, um die Programme zu entwickeln und zu kommentieren. Besonders einfach können Sie ein Programm mit Hilfe des Menüs *Befehlshilfe* schreiben: Wenn Sie den Befehl auswählen, erhalten Sie sofort die notwendigen Eingabefelder eingeblendet. Nach der Eingabe der Werte wird automatisch die Syntax gebildet und Sie können den kompletten Befehl in Ihr Programm übernehmen.

Mit der Teach-in-Programmierung fahren Sie die Achse an die gewünschte Stelle und speichern einfach die erreichte Position. So können Sie schnell die kompliziertesten Verstell- und Bewegungsabläufe programmieren.

Alle Befehle sind in der Software-Referenz zunächst in einer Übersicht und anschließend alphabetisch geordnet ausführlich beschrieben und mit kurzen Beispielen ergänzt. Wie diese genutzt werden, zeigen knapp 50 Programmbeispiele in der Online-Hilfe.

Und in der Parameter-Referenz finden Sie alle Parameter, zuerst in einer Übersicht und im Anschluss daran im Detail beschrieben.

◻ Programmlayout

Üblicherweise beginnt ein Programm mit der Definition der Arrays, Interrupts und Benutzerparameter; zum Beispiel:

```
DIM send[12], receive[12]     // Array

ON ERROR GOSUB errhandle // Interrupts
ON INT -1 GOSUB stopprog
ON PERIOD 500 GOSUB calc
ON TIME 10000 GOSUB break

LINKGPAR 1990 "Offset [qc]" 0 100000 0
                  // Anwendungsparameter
```

Im nächsten Schritt wird die Initialisierung durchgeführt: Parameter setzen, Flags und Variablen.

```
SET POSERR 100000000        // Parameter
SET 1990 10000
SETVLT 205 50

offset = 0                  // Flags/variablen
sync_flag = 0

VEL 100                     // System Parameter
ACC 100
DEC 100
```

Es folgt die Hauptprogrammschleife:

```
main:
…
GOTO main
```

```
main:
IF (IN 3 == 1) THEN
/* Synchronisationsmodus, wenn Eingang 3 = 1 */
   GOSUB syncprog
ELSE    /* Drehzahlmodus, wenn Eingang 3 nicht 1 */
   GOSUB speedprog
GOTO main
```

Unterprogrammbereiche werden wie folgt definiert:

```
SUBMAINPROG
   SUBPROG name
   …
   RETURN
ENDPROG
```

```
SUBMAINPROG
    SUBPROG syncprog
        IF (sync_flag == 0) THEN
        /* synchronisieren, falls nicht schon aktiv */
            SYNCP
            sync_flag = 1
        ENDIF
    RETURN
    SUBPROG errhandle
        WAITI 18 on
        /* auf Eingang 18 warten, Fehler löschen */
            sync_flag = 0
        ERRCLR
    RETURN
ENDPROG
```

☐ Sequentielle Befehlsabarbeitung

Generell wird ein Befehl vollständig abgearbeitet, bevor ein neuer begonnen wird. Das führt dazu, dass bei Positionierbefehlen gewartet wird, bis die Zielposition erreicht ist.

Ausnahme: Wenn NOWAIT ON gesetzt ist.

☐ Befehls-Ausführungszeiten

Die Ausführungszeiten von GETVLT und SETVLT Befehlen hängt von der Lese-/Schreib-Leistung des FC 300 ab. Der GETVLT Befehl dauert typischerweise 20 ms, kann aber auch schneller sein. Der SETVLT Befehl kann ziemlich langsam sein und die Ausführungszeit hängt davon ab, was gerade im FC 300 passiert.

Es ist nicht möglich eine maximale Zeit für das Lesen oder Schreiben eines Befehls bezüglich FC 300 Parameter zu nennen. (Es kann bis zu 100 oder 500 ms oder sogar länger dauern.)

Falls die Ausführungszeit der Befehle in einer Anwendung kritisch sein sollte, können Sie die Ausführungszeiten einer Befehlssequenz mit Hilfe des Befehls TIME unter den verschiedenen Betriebsbedingungen messen.

□ Tipps zur Erhöhung der Programmlesbarkeit

Verwenden Sie Groß- und Kleinschreibung, zum Beispiel alle Befehle groß, Variablen klein.

Fügen Sie Leerzeichen zwischen den Befehlsteilen ein.

Kommentieren Sie Ihr Programm. Die Kommentare stehen zwischen /* … */ oder nach //…
/* Beginn KOMMENTAR Ende */ oder
// Beginn KOMMENTAR Ende
Unzulässig ist die aber Schachtelung von Kommentaren (/* **...** /***...***/ **...** */)

Verwenden Sie Zeileneinzüge innerhalb von Schleifen.

□ Befehlsstruktur

Jeder Befehl besteht aus einem BEFEHLSWORT+ ggf. *Parameter.* Eine Variable kann auch als Parameter statt einer absoluten Zahl benutzt werden.

Beispiel	POSA 10000
oder	pos = 10000 POSA pos

□ Werteeingaben

Wie in anderen Programmiersprachen werden Werteeingaben auch hier nicht geprüft. Es liegt also in der Verantwortung des Programmierers, wenn zu extreme Werte Probleme bereiten. Bei der Suche nach solchen Problemen können Sie den Debug-Modus benutzen.

□ Fehlerhandhabung (Error Handling)

Es können immer Situationen wie Timeout, Positionsfehler (Schleppfehler) oder ein Nothalt auftreten und müssen daher berücksichtig werden; im Allgemeinen benutzt man dafür Unterprogramme. Andernfalls würde das Programm ohne jede Möglichkeit, den Fehler zu löschen abbrechen.

In diesem Programmbeispiel wird der Fehlerstatus mittels einer Fehlernummer ausgewertet und das Unterprogramm „errhandle” aufgerufen, wenn ein Fehler auftritt.

```
ON ERROR GOSUB errhandle
PRINT "Temporären Fehlerzustand erzeugen "
PRINT "durch Betätigen des Endschalters."
endlos:         /* Endlosschleife */
GOTO endlos
/* UNTERPROGRAMMBEREICH  */
SUBMAINPROG
    SUBPROG errhandle
    // Fehlernummer auswerten und
    // Fehlermeldung rücksetzen
        PRINT "Aktuelle Fehlernummer: ",ERRNO
        IF (ERRNO == 25) THEN
        /* Endschalter oder Maschinen-Stopp */
           PRINT "HW Endschalter aktiviert"
        ELSE
          PRINT "ok, obwohl es kein Endschalter war."
        ENDIF
        PRINT "Sie haben 10 Sek. Zeit, ";
        PRINT "den Fehler zu beheben"
        DELAY 10000      /* 10 Sekunden warten */
        PRINT "Fehlerstatus zurücksetzen ..."
        ERRCLR    /* Fehlermeldung zurücksetzen */
        PRINT "... und Programm beenden."
        EXIT                /* Programmabbruch */
    RETURN
ENDPROG
```

□ Debugging

Mit *Entwicklung → Meldungen → Log-Datei* wird die Protokollierung der Meldungen gestartet. → *Logdatei beenden* ist nur anwählbar, wenn die Protokollierung gestartet ist.

Klicken Sie auf *Entwicklung → Syntaxprüfung.* Sobald ein falscher Befehl gefunden ist, wird das Programm abgebrochen. Die Zeilennummer und eine Fehlerbeschreibung werden im Kommunikationsfenster ausgegeben. Der Cursor steht automatisch auf der Position des Syntaxfehlers und das Programm stoppt genau an diesem Punkt.

Die *Syntaxprüfung* erzeugt zusätzlich eine Debug-Datei namens „temp.ad$".

Klicken Sie auf *Entwicklung → Vorbereiten Einzelschritt* und das geöffnete Programm wird für den Debug-Modus vorbereitet. Es wird kompiliert und eine Debug-Datei erzeugt; außerdem wird das Programm in den FC 300 geladen und alle ausführbaren Programmzeilen werden mit blauen Punkten markiert. Nun sind auch alle entsprechenden Menüpunkte aktiviert. Mehr Details finden Sie auf Seite 58 im Abschnitt Debug-Modus, beginnend mit *Vorbereiten Einzelschritt*.

□ Interrupts

Generell gibt es diese Sorten von Interrupts:

ON INT	Interrupt bei Flanken eines Eingangs
ON PERIOD / ON TIME	Interrupt nach Ablauf einer Zeitspanne
ON COMBIT / ON STATBIT	Interrupt wenn Bit n gesetzt wird
ON PARAM	Interrupt wenn sich ein Parameter n ändert

□ Generelle Abarbeitung von Interrupt-Prozeduren

Nach jedem internen APOSS Befehl wird abgefragt, ob ein Interrupt-Ereignis vorliegt. Dabei ist zu beachten, dass mit jedem internen APOSS Befehl der Compiler eine Anweisung im APOSS-Maschinencode erzeugt.

So wird zum Beispiel eine einfache Anweisung wie:

POSA (ziel + 1000)

in folgenden APOSS-Maschinencode zerlegt:

MOVE *ziel* nach Register 101
MOVE Immediate 1000 nach Register 102
ADDREG Register 102 plus Register 101 nach Register 101
POSA Achse 0 nach Register 101

Außerdem wird bei länger dauernden Befehlen (wie DELAY oder WAITAX) ständig geprüft, ob ein Interrupt-Ereignis aufgetreten ist. In diesem Fall wird der Befehl unterbrochen und nach Abarbeitung des Interrupts wieder fortgesetzt.

ACHTUNG!:
Verwenden Sie nicht WAITT in Verbindung mit Interrupts, da dabei der Wartevorgang nach der Unterbrechung erneut von vorn beginnt.

□ Benutzung von Variablen innerhalb von Interrupt-Prozeduren

Das obige Beispiel mit dem „APOSS-Maschinencode" zeigt auch deutlich, dass bei der Zuweisung von Variablen innerhalb von Interrupt-Prozeduren mit größter Sorgfalt vorgegangen werden muss.

Wird zum Beispiel im Hauptprogramm eine Zuweisung der Art:

ziel = *ziel* + wert – 1000

vorgenommen, wird diese in eine Folge von APOSS-Maschinencode-Befehlen zerlegt, wobei die Zwischenergebnisse in temporären Registern gespeichert werden. Erst am Ende der Folge wird das Ergebnis nach *ziel* zurückgespeichert.

Wird nun während der Ausführung dieser Befehle ein Interrupt ausgelöst und in der entsprechenden Prozedur ein Befehl

ziel = 0

ausgeführt, wird es in diesem Fall Probleme geben. Denn nach der Abarbeitung der Interrupt-Prozedur wird in das Hauptprogramm zurückgesprungen und dann das immer noch vorhandene Zwischenergebnis nach *ziel* gespeichert: Somit wird die 0 in *ziel* wieder überschrieben.

□ ON PERIOD innerhalb von Interrupt-Prozeduren

Bei ON PERIOD Funktionen wird dagegen beim Start einer solchen Funktion die Zeit berechnet, wann der nächste Aufruf erfolgen soll, also

START_TIME = TIME + PERIOD.

Sobald diese Zeit erreicht ist, wird die Funktion ausgeführt und anschließend die nächste Startzeit berechnet mit der Formel

START_TIME = START_TIME + PERIOD.

Dies sorgt dafür, dass die Abstände des Aufrufens wirklich gleich sind, da die Ausführungszeit die Berechnung nicht beeinflusst. Dies bedeutet aber auch, dass der Anwender darauf achten muss, dass die Periode wirklich länger als die Ausführungszeit ist, da sonst ein „Stau" entsteht, das heißt es würde eigentlich nur noch die ON PERIOD Funktion ausgeführt.

□ Reaktionszeiten

Das Vorhandensein eines Interrupts wird in einer speziellen Funktion geprüft, die auch zur Watch-Dog-Überwachung verwendet wird. Deshalb wird diese generell in jeder Prozedur die etwas länger dauern könnte und in allen Schleifen etc. aufgerufen.

In dieser Prozedur wird immer nach 1 ms geprüft ob ein solches Ereignis vorliegt und gegebenenfalls ein entsprechendes Flag gesetzt. Dieses Flag wird dann spätestens nach Abarbeitung des gerade aktuellen APOSS-Maschinencodes erkannt und ausgewertet.

Die Reaktionszeit ist die maximale Ausführungszeit eines Maschinencodes oder 1 ms, je nachdem was größer ist.

Eine Ausnahme bildet der Zeit-Interrupt (ON TIME / ON PERIOD). Hier wird nur alle 20 ms geprüft ob die Zeit abgelaufen ist. Daher macht es auch keinen Sinn, ON PERIOD mit weniger als 20 ms zu definieren.

ACHTUNG!:
Außerdem ist generell darauf zu achten, dass Interrupt-Funktionen nicht zu lange dauern. Vor allem bei ON PERIOD Funktionen ist dringend darauf zu achten, dass die Funktion nicht länger dauert als die Periode, weil sonst ein Stau von Funktionsaufrufen entsteht.

□ Prioritäten

Falls zwei Interrupt Ereignisse gleichzeitig auftreten, werden sie in folgender Reihenfolge abgearbeitet:

ON INT geht vor

ON APOS, ON MAPOS, ON MCPOS vor

ON COMBIT vor

ON STATBIT vor

ON PARAM vor

ON TIME / PERIOD, wobei die anderen Ereignisse aber nicht verloren gehen.

Innerhalb der einzelnen Interrupt-Typen gilt wiederum Folgendes:

ON INT / ON COMBIT / ON STATBIT

Sollten zwei (Eingangs)-Interrupts gleichzeitig kommen, wird der mit der niedrigeren Nummer zuerst ausgeführt, wobei der andere aber nicht verloren geht. Die anderen werden dann nach Beendigung der Interrupt-Prozedur entsprechend aufgerufen.

Sollte derselbe Eingang oder Interrupt während der Ausführung der Prozedur noch einmal kommen, wird auch dieser wieder vermerkt und anschließend ausgeführt.

Ein Interrupt kann also nur verloren gehen, wenn er während der Ausführung einer Interrupt-Prozedur zweimal kommt.

ON TIME / ON PERIOD

Wie bereits oben beschrieben, wird in einer internen Struktur für jede Zeitfunktion die nächste Ausführungszeit vermerkt. Bei gleichzeitiger Ausführungszeit wird die Prozedur zuerst ausgeführt, die zuerst in der Liste steht. Die Priorität ergibt sich also aus der Reihenfolge der ON PERIOD Befehle.

ON PARAM

Wenn mehrere dieser Interrupts gleichzeitig auftreten, werden sie in der Reihenfolge der ON PARAM Befehle im Programm abgearbeitet.

□ Interrupt Schachtelung

Es ist nicht möglich, dass ein Interrupt von einem anderen Interrupt unterbrochen wird. Während ein Interrupt behandelt wird, kann demnach kein zweiter behandelt werden. Einzige Ausnahme ist die ON ERROR Funktion, die auch während der Abarbeitung von Interrupts möglich ist.

Eine ON ERROR Funktion kann aber von keinem Interrupt unterbrochen werden.

□ NOWAIT in Interrupts

Generell ist während eines Interrupts NOWAIT auf ON gesetzt, das heißt dass nicht auf die Beendigung von POSA Befehlen gewartet wird.

Dies ist nötig, da sonst ein POSA Befehl von einer Interrupt-Prozedur nicht unterbrochen werden kann, denn es würde sofort darauf gewartet werden, dass die Achse die Zielposition erreicht. Will man also innerhalb einer Interrupt-Prozedur auf die Beendigung einer Positionierung warten, muss man dies explizit mit WAITAX tun.

□ Sprachelemente

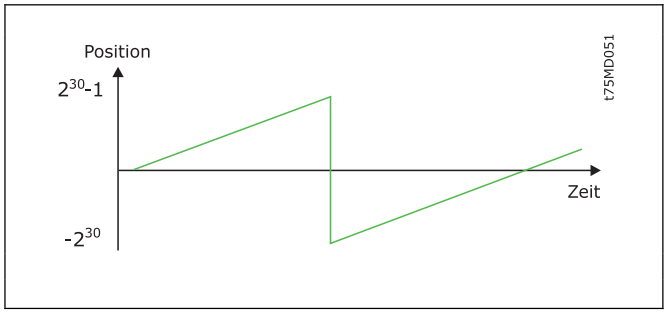
□ APOSS Zahlenformate

Word (16 Bit): 2^{16} - 1 = 65535

Long Word (32 Bit): -2^{31} to $+2^{31}-1$ = -2147483648 to + 2147483647

Positionen (32 Bit, wobei 1 Bit für den Überlauf benutzt wird):
-2^{30} to $+2^{30}-1$ entspricht -1073741823 to + 1073741823

Beim Positionieren und Synchronisieren springt die Steuerung reibungslos von Position 1 Milliarde zur Position –1 Milliarde; es gehen keine Impulse verloren.



☐ Konstanten

Überall dort, wo Parameter oder Werte erwartet werden, können Konstanten stehen, die typischerweise in Ganzzahlenwerte eingegeben werden, zum Beispiel: wert = 5000

Konstanten
- sind Ganzzahlenwerte im Bereich von –2 bis +2 Mrd.,
- gelten innerhalb des gesamten Programms (sie sind global),
- können dezimal, hexadezimal (0x + Hexadezimalzahl), oktal (0 + Oktalzahl) oder in ASCII (zwischen Apostroph) eingegeben werden, zum Beispiel
  wert = 5000 = Dezimal 5000
  wert = 0x7F = Dezimal 127
  wert = 0100 = Dezimal 64
  wert = 'A' = Dezimal 65

Besonders die Hexadezimal- und ASCII-Eingaben vermeiden manche Umrechnung und das Programm wird lesbarer, zum Beispiel: taste = 'A'

Der Vorteil von Konstanten ist, dass sie keinen eigenen Speicherplatz benötigen.

☐ Variablen

Variablen
- können zur Zwischenspeicherung von Abfrage- und Rechenergebnissen verwendet werden,
- entstehen durch die Zuweisung eines Wertes,
- müssen nicht separat definiert werden,
- gelten innerhalb des gesamten Programms (sie sind global),
- enthalten Ganzzahlenwerte im Bereich von –2 bis +2 Mrd.,
- können innerhalb von Befehlen statt fester Werte verwendet werden,
- müssen vor der Verwendung in einem Befehl einen Wert zugewiesen bekommen.

Variablennamen
- können beliebig lang sein,
- können aus Buchstaben, Ziffern und dem Unterstrich bestehen,
- dürfen keine länderspezifischen Zeichen, wie Umlaute enthalten,
- müssen mit einem Buchstaben beginnen,
- können groß oder klein geschrieben werden (keine Unterscheidung!),
- dürfen nicht mit einem Befehlsnamen identisch sein.

Spezielle Variablen
ERRNO = Systemvariable, die die aktuelle Fehlernummer enthält.

☐ Arrays

Die Programmierung von Programmen mit Dialog erfordert die Speicherung von Benutzereingaben oder Positionen über längere Zeit, also auch nach dem Ausschalten der Steuerung. Meistens sind dies mehrere Werte, die am besten in Feldern bzw. Arrays abgelegt werden.

Die Arrays werden im Speicherbereich der Benutzerprogramme abgelegt und sind global definiert, das heißt unabhängig vom aktuellen Programm. Der Benutzer kann selbst festlegen, wie viele Arrays er definiert und wie groß die einzelnen Arrays sein sollen. Die Festlegung erfolgt durch die Anweisung DIM und ist danach fest und kann nicht mehr geändert werden (außer durch Speicher löschen). In jedem Programm, das Arrays benutzen soll, muss eine entsprechende DIM Anweisung stehen, die mit der ursprünglichen Definition übereinstimmt; andernfalls wird ein Fehler gemeldet.

DIM Anweisung

DIM muss die erste Anweisung in einem Programm sein und noch vor dem Unterprogrammbereich erscheinen.

Die DIM Anweisung vereinbart die später verwendbaren Arrays. Sollten bis dahin noch keine Arrays angelegt gewesen sein, werden sie neu angelegt. Waren bereits Arrays definiert, müssen die Angaben mit der ursprünglichen Definition übereinstimmen.

Beispiel DIM ziel1[20], ziel2[20], ziel3[20], werkoffset[50]
DIM parameter[10]

Mit diesen Befehlen werden insgesamt 5 Arrays mit den entsprechenden Größen definiert. Wenn dieses Programm einmal ausgeführt wurde, sind die obigen Arrays in der Steuerung angelegt. Wird bei einem erneuten Start eines Programms festgestellt, dass die Array-Definition von den Arrays in der Steuerung abweicht, wird dies als Fehler angezeigt. Allerdings ist es korrekt, wenn ein zweites Programm nur folgende Zeile enthält:

Beispiel DIM ziel1[20], ziel2[20], ziel3[20]

Die Reihenfolge der Definition muss aber immer gleich sein, da die Steuerung nicht die Namen der Arrays speichert, sondern nur deren Position in der DIM Anweisung. So ist auch die folgende Programmzeile korrekt und das Array xpos ist dann identisch mit dem Array ziel1.

Beispiel DIM xpos[20], ypos[20], zpos[20], offs[50]

Indizes

Die Elemente eines Arrays werden über einen entsprechenden Index in eckigen Klammern bezeichnet: xpos[5]. Dabei sind Indizes von 1 bis zur Größe des definierten Arrays erlaubt. Also im obigen Fall bei xpos von 1 bis 20. Wird versucht, auf Elemente davor oder dahinter zuzugreifen, erfolgt eine Fehlermeldung, da dies zu Datenverlust und Zerstörung der Arrays führen könnten.

Arrays schreiben und lesen

Der Zugriff auf die so definierten Arrays erfolgt analog zu der Benutzung von Variablen. So sind im folgenden Beispiel alle Anweisungen korrekt:

```
xpos [1] = 10000
xpos [2] = 20000
xpos [3] = 30000
i = 1
WHILE (i<20) DO
    ypos [i] = i*1000
    i = i+1
ENDWHILE
zpos [1] = APOS
POSA xpos [1]
offs [1] = (xpos[2]) % 20
```

Arrays versus Variablen

Grundsätzlich können Arrays überall dort verwendet werden, wo auch Variablen zulässig sind. Auch belegt ein Array nur den Platz einer internen Variablen und vermindert somit die Zahl der maximal zulässigen Variablen lediglich um eins. Die maximale Anzahl von Variablen ist im Menü *Einstellungen → Compiler* einstellbar.

□ Arithmetik, Operatoren

Der Compiler bietet folgende Befehle und Parameter:

Operatoren	plus, minus, mal, geteilt, XOR, Modulo, Division, Absolutbetrag
Bitoperatoren	und, oder, invertieren, linksschieben, rechtsschieben, Bit, Byte, Word, Long
Vergleichsoperationen	größer als, kleiner als, größer, gleich als, kleiner, gleich als, gleich wie, ungleich
logische Verknüpfungen	und, oder, nicht

Informieren Sie sich im Anschluss an die Art der Zuweisung (Assignment Operation), die entsprechend den Bit-/Byte-Befehlen aufgebaut ist und über die Prioritäten der Operatoren und Operationen.

ACHTUNG!:
Alle Arithmetikoperationen sind Ganzzahloperationen!

□ Operatoren

Symbol	Bedeutung	Syntax / Beispiel	Beschreibung
+	plus	3 + 3= 6	Addition
–	minus	9 – 3 = 6	Subtraktion
*	mal	2 * 3 = 6	Multiplikation
%	geteilt	19 % 3 = 6	Division (Ergebnis abgeschnitten)
^	XOR	expr1 ^ expr2 127 ^ 255 = 128	Exklusiv Oder (binäre Operation)
mod	Modulo	expr1 mod expr2 250 mod 16 = 10	Mathematisches Modulo (Rest einer Integerdivision)
rnd	Division	expr1 rnd expr2 250 rnd 16 = 16	Division mit Runden, im Gegensatz zur Division (%) mit Abschneiden
abs	Absolutbetrag	Abs(expr) abs (-5) = 5	Absolutbetrag des Ausdrucks

□ Bitoperatoren

Symbol	Bedeutung	Syntax / Beispiel	Beschreibung	Wertebereich
&	und	7 & 6 = 6	bitweise Verknüpfung	
|	oder	2 | 4 = 6	bitweise Verknüpfung	
~	invertieren	~(–7) = 6	bitweises Invertieren	
<<	linksschieben	3 << 1 = 6	bitweises Linksschieben	
>>	rechtsschieben	12 >> 1 = 6	bitweises Rechtsschieben	
.	Bit	expr1.expr2 7.1 = 1 7.3 = 1 7.4 = 0	Liefert das Bit expr2 von expr1 zurück	1 - 32
.b	Byte	expr1.b expr2 0x027F.b1 = 127 0x027F.b2 = 2	Liefert das Byte expr2 von expr1 zurück	1 - 4
.w	Word	expr1.w expr2 0x0010FFFF.w2 = 16	Liefert das Wort expr2 von expr1 zurück	1 - 2
.l	Long	expr1.l expr2	Liefert das Long expr2 von expr1 zurück (Standard)	

☐ Vergleichsoperationen und logische Verknüpfungen

Vergleichsoperationen	
>	größer als
<	kleiner als
>=	größer, gleich als
<=	kleiner, gleich als
==	gleich wie
!=	ungleich

Logische Verknüpfungen	
AND	und
OR	oder
NOT	nicht

☐ Zuweisung (Assignment Operation)

Zuweisung		Beschreibung	Wertebereich
Wert	= 0	Standard Zuweisung zu einer Variablen	
Feld[1]	= 0	Standard Zuweisung zu einem Array Wert	
Wert.3	= 1	Bit 3 wird auf 1 gesetzt, Wert = 4	1 - 32
Feld[1].8	= 1	Bit 8 wird auf 1 gesetzt, Feld[1] = 128	1 - 32
Wert.b1	= 72	Unterstes Byte von Wert wird auf 72 gesetzt Wert = 72	1 - 4
Wert.b2	= 128	Zweites Byte von Wert wird auf 128 gesetzt Wert = 0x00008048	1 - 4
Wert.w2	= 15	Zweites Wort von Wert wird auf den Wert 15 gesetzt. Wert = 0x000F8048	1 - 2

☐ Priorität der Operatoren und Operationen

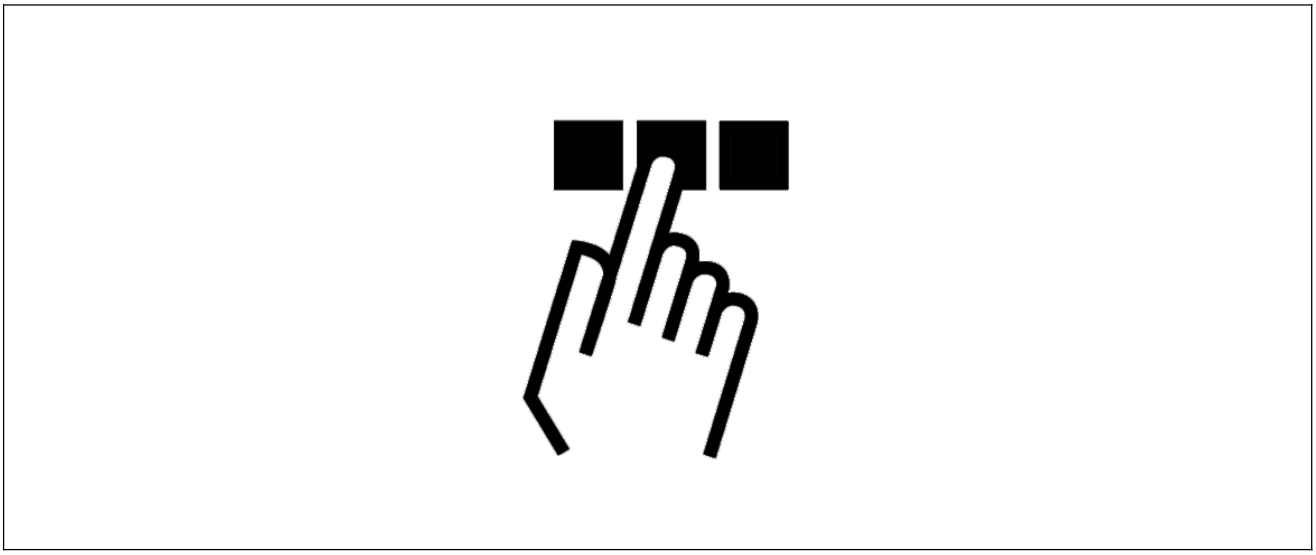
Operatoren in derselben Zeile haben die gleiche Priorität, werden also nacheinander abgearbeitet.

Die Prioritäten sind in absteigender Folge erläutert:

*	%	(multiplikativ)
+	–	(additiv)
>>	<<	(bitweises Schieben / shiften)
>=	<= > <	(Relation)
==	!=	(Gleichheit/Equality)

&	(bitweises und)
|	(bitweises inklusive oder)
AND	(logisches und)
OR	(logisches oder)

Software-Referenz



Befehlsübersicht

Alle Befehle werden zuerst in einem allgemeinen Überblick in Gruppen genannt und im folgenden Abschnitt dann alphabetisch geordnet detailliert beschrieben sowie mit kurzen Programmbeispielen ergänzt.

Befehle zum Initialisieren

Befehle zum Initialisieren der Achse und der MCO 305 sowie zum Anfahren und Definieren der/des Nullpunkte(s). (Gruppe INI)

Command	Beschreibung	Syntax	Parameter
DEFMORIGIN	Aktuelle Master-Position als Nullpunkt für den Master setzen	DEFMORIGIN	–
DEF ORIGIN	Istposition als Nullpunkt setzen	DEF ORIGIN	–
DELETE ARRAYS	Alle Arrays im RAM löschen.	DELETE ARRAYS	–
ERRCLR	Fehlermeldung löschen	ERRCLR	–
HOME	Maschinennullpunkt anfahren	HOME	–
INDEX	nächste Indexposition anfahren	INDEX	–
MOTOR OFF	Motorregelung ausschalten	MOTOR OFF	–
MOTOR ON	Motorregelung einschalten	MOTOR ON	–
RST ORIGIN	Temporärnullpunkt löschen	RST ORIGIN	–
SETMORIGIN	Aktuelle Position als Nullpunkt für den Master setzen	SETMORIGIN wert	wert = absolute Position
SET ORIGIN	Temporärnullpunkt setzen	SET ORIGIN p	p = absolute Position
SAVE ARRAYS	Arrays im EEPROM sichern	SAVE ARRAYS	–
SAVE AXPARS	Aktuelle Achsparameter im EEPROM sichern	SAVE AXPARS	–
SAVE GLBPARS	Aktuelle globale Parameter im EEPROM sichern	SAVE GLBPARS	–
SAVEPROM	Speicher in EEPROM sichern	SAVEPROM	
SWAPMENC	Master- und Slave-Drehgeber intern tauschen	SWAPMENC s	s = Bedingung ON / OFF

□ Steuerungsbefehle

Befehle zur Steuerung des Programmablaufs und zum Strukturieren von Programmen. (Gruppe CON)

Befehl	Beschreibung	Syntax	Parameter
CONTINUE	Abgebrochene Positionier- und Drehzahl-befehle fortsetzen, zum Beispiel nach einem MOTOR STOP	CONTINUE	–
DELAY	Zeitverzögerung	DELAY t	t = Verzögerung in ms
DIM	Definition eines Arrays	DIM array [n]	array = Name n = Anzahl der Elemente
EXIT	Vorzeitiger Programmabbruch	EXIT	–
GOSUB	Aufruf eines Unterprogramms	GOSUB name	name = Name des Unter-programms
GOTO	Sprung zu einem Programmlabel	GOTO label	label = Zielposition
IF THEN	Bedingte einfache Programmverzweigung	IF Bedingung THEN Befehl	Bedingung = Verzweigungs-kriterium
… ELSE IF THEN	Bedingte mehrfache Programmverzweigung	ELSEIF Bedingung THEN Befehl	Befehl = ein oder mehrere Programmbefehle
… ELSE	Alternative Programmverzweigung	ELSE Befehl	
… ENDIF	Ende der Programmverzweigung	ENDIF	
LOOP	Definierte Schleifenwiederholung	LOOP n label	n = Anzahl der Schleifenwieder-holungen label = Zielposition
MOTOR STOP	Stoppen des Antriebs	MOTOR STOP	–
NOWAIT	Wartemodus ein-/ausschalten	NOWAIT s	s = Zustand ON / OFF
REPEAT	Bedingte Schleife Anfang	REPEAT	
REPEAT… UNTIL	Bedingte Schleife Ende	UNTIL Bedingung	Bedingung = Abbruchkriterium
SUBMAINPROG	Beginn der Definition des Unterprogramms	SUBMAINPROG	–
… ENDPROG	Ende der Definition des Unterprogramms	ENDPROG	–
SUBPROG	Beginn eines Unterprogramms	SUBPROG name	name = Name des Unter-programms
… RETURN	Ende eines Unterprogramms	RETURN	–
SYSVAR	Systemvariable (Pseudo-Array) liest Systemwerte	SYSVAR [n]	n = Index
VLTALARMSTAT	Gibt an, ob ein Alarm vorliegt oder nicht.	VLTALARMSTAT	–
VLTCONTROL	Setzt das VLT Steuerwort im Status MOTOR OFF.	VLTCONTROL Wert Steuerwort	Wert
VLTERRCLR	Löscht einen VLT-Alarm	VLTERRCLR	–
WAITAX	Warten bis Zielposition erreicht ist	WAITAX	–
WAITI	Warten auf bestimmten Eingangszustand	WAITI n s	n = Eingangsnummer s = erwarteter Zustand ON / OFF
WAITNDX	Warten auf Index	WAITNDX t	t = Timeout in ms
WAITP	Warten bis Position erreicht	WAITP p	p = absolute Position
WAITT	Zeitverzögerung	WAITT t	t = Verzögerung in ms
WHILE … DO	While-Schleife Anfang	WHILE Bedingung DO	Bedingung = Abbruchkriterium
ENDWHILE	While-Schleife Ende	ENDWHILE	–

Befehl	Beschreibung	Syntax	Parameter
#INCLUDE	Einfügen des Inhalts einer Datei	#INCLUDE file	file = Name der Datei, die eingefügt wird

□ Ein-/Ausgabe-Befehle (I/O)

Befehle zum Setzen und Rücksetzen der Ausgänge, Abfragen der Eingänge, Abfragen von Bewegungsinfos, Abfragen von Systemdaten und zum Ein- und Ausgeben von Benutzerinformationen. (Gruppe I/O)

Befehl	Beschreibung	Syntax	Parameter
APOS	Istposition lesen	erg = APOS	–
AVEL	Aktuelle Geschwindigkeit der Achse abfragen	erg = AVEL	–
AXEND	Status der Programmausführung abfragen	erg = AXEND	–
CPOS	Sollposition lesen	erg = CPOS	–
ERRNO	Fehlernummer lesen	erg = ERRNO	–
IN	Eingänge bitweise lesen (einzeln)	erg = IN n	n = Nummer des Eingangs
INAD	Analogeingang lesen	erg = INAD n	n = Nummer des analogen Eingangs
INB	Eingänge byteweise lesen (8 Stück).	erg = INB n	n = Eingangsnummer
INKEY	Tastencodes des FC 300 lesen.	INKEY p	p = 0 (auf Zeichen warten) p > 0 (max. p ms warten) p < 0 (nicht warten)
IPOS	Letzte Index- bzw. Markerposition des Slaves abfragen.	erg = IPOS	–
MAPOS	Aktuelle Istposition des Masters abfragen.	erg = MAPOS	–
MAVEL	Aktuelle Geschwindigkeit des Masters abfragen.	erg = MAVEL	–
MIPOS	Letzte Index- bzw. Markerposition des Masters abfragen.	erg = MIPOS	–
OUT	Digitale Ausgänge bitweise setzen (einzeln).	OUT n s	n = Nummer des Ausgangs s = Zustand ON / OFF
OUTAN	FC 300 Bus-Sollwert setzen.	OUTAN w	w = Bus-Sollwert
OUTB	Digitale Ausgänge byteweise setzen (8 Stück).	OUTB n w	n = Ausgangsbyte w = Wert
OUTDA	FC 300 analoge Ausgänge setzen	OUTDA n w	n = Nummer des Ausgangs w = Wert
PID	PID-Berechnung durchführen	u(n) = PID e(n)	e(n) = aktuelle Abweichung
PRINT	Text und Variablen im Display ausgeben.	PRINT i oder PRINT i;	i = Information
PRINT DEV	Stoppt die Ausgabe von Informationen.	PRINT DEV nn printlist	nn = Ausgabeschnittstelle 0 = Standard -1 = danach keine Ausgabe printlist = Argument
STAT	Status der Achse lesen	erg = STAT	–
SYNCERR	Aktuellen Synchronisationsfehler des Slaves abfragen.	erg = SYNCERR	–

Befehl	Beschreibung	Syntax	Parameter
TESTSETP	Aufzeichnungsdaten für Testfahrt festlegen	TESTSETP ms vi1 vi2 vi3 arrayname	ms = Interval in ms vi 1 … 3 = Indizes der Werte, die aufgezeichnet werden sollen arrayname = Array, das für die Aufzeichnung benutzt wird
TESTSTART	Aufzeichnung der Testfahrt starten	TESTSTART nr	nr = Anzahl der durchzuführenden Messungen
TIME	Systemzeit auslesen	erg = TIME	–
TRACKERR	Aktuellen Schleppabstand einer Achse abfragen	erg = TRACKERR	–
_GETVEL	Abtastzeit für AVEL und MAVEL ändern	var = _GETVEL t	t = Abtastzeit in ms

□ Interrupt-Funktionen

Befehl	Beschreibung (Gruppe INT)	Syntax	Parameter
DISABLE …	Sperrt die Ausführung von Interrupts.	DISABLE inttyp	inttyp = INT, COMBIT, …
ENABLE …	Gibt gesperrte Interrupts wieder frei.	ENABLE inttyp	inttyp = INT, COMBIT, …
ON APOS ..	Unterprogramm aufrufen, wenn die Slave-Position xxx passiert wurde.	ON sign APOS xxx GOSUB name	sign = Fahrtrichtung xxx = Slave-Position [BE] name = Unterprogramm
ON COMBIT ..	Unterprogramm aufrufen, wenn Bit n des Kommunikationspuffers gesetzt ist	ON COMBIT n GOSUB name	n = Bit n des Kommunikationspuffers name = Unterprogramm
ON DELETE ..	Löscht einen Positions-Interrupt: ON APOS, ON MCPOS oder ON MAPOS.	ON DELETE pos GOSUB name	pos = Wert name = Unterprogramm
ON ERROR ..	Unterprogramm bei Fehler aufrufen	ON ERROR GOSUB name	name = Unterprogramm
ON INT ..	Unterprogramm bei Flanke eines Eingangs aufrufen	ON INT n GOSUB name	n = zu überwachender Eingang name = Unterprogramm
ON MAPOS ..	Unterprogramm aufrufen, wenn die Master-Position xxx [qc] passiert ist	ON sign MAPOS xxx GOSUB name	sign = Fahrtrichtung xxx = Master-Position name = Unterprogramm
ON MCPOS ..	Unterprogramm aufrufen, wenn die Master-Position xxx (MU) passiert ist	ON sign MCPOS xxx GOSUB name	sign = Fahrtrichtung xxx = Master-Position name = Unterprogramm
ON PARAM ..	Unterprogramm aufrufen, wenn sich ein Parameter ändert.	ON PARAM n GOSUB name	n = Parameternummer name = Unterprogramm
ON PERIOD ..	Unterprogramm in regelmäßigen Zeitabständen aufrufen.	ON PERIOD n GOSUB name	n > 20 ms (Zeit für Wiederaufruf) n = 0 (Funktion abschalten) name = Unterprogramm
ON STATBIT..	Unterprogramm aufrufen, wenn Bit n des Statuswortes gesetzt ist.	ON STATBIT n GOSUB name	n = Bit n des FU Status name = Unterprogramm
ON TIME..	Unterprogramm nach einmaligem Zeitablauf aufrufen.	ON TIME n GOSUB name	n = Zeit bis Wiederaufruf name = Unterprogramm

□ Befehle für die Handhabung der Parameter

Alle mit einer Parameterkennung versehenen globalen und Achsparameter können mit den folgenden Befehlen gesetzt und gelesen werden. (Parameterkennungen siehe Übersicht der Parameter-Referenz.) (Gruppe PAR)

Befehl	Beschreibung	Syntax	Parameter
GET	Parameterwerte lesen (MCO 305 und Anwendungsparameter)	erg = GET par	par = Parameterkennung
GETVLT	FC 300 Parameterwerte lesen	erg = GETVLT par	par = Parameternummer
GETVLTSUB	FC 300 Parameterwerte mittels Indexnummer lesen	erg = GETVLTSUB par indxno	par = Parameternummer indxno = Indexnummer
LINKGPAR	Globalen Parameter mit dem LCP-Display verknüpfen.	LINKGPAR parno "text" min max type	parno = LCP Par. Nummer text = ASCII Text min = min. Wert max = max. Wert type = Online / Offline Par.
LINKSYSVAR	Systemvariable mit dem LCP-Display verknüpfen.	LINKSYSVAR indx parno "text"	indx = SYSVAR Index parno = LCP Par. Nummer text = Anzeigentext
SET	Parameterwerte setzen (MCO 305 und Anwendungsparameter).	SET par v	par = Par. Identifikation v = Parameterwert
SETVLT	FC 300 Parameterwerte setzen.	SETVLT par v	par = Parameternummer v = Parameterwert
SETVLTSUB	Setzt FC 300 Parameterwerte mit Indexnummer.	SETVLTSUB par indxno v	par = Parameternummer indxno = Indexnummer v = Parameterwert

□ Befehle der Kommunikationsoption

Befehl	Beschreibung	Syntax	Parameter
COMOPTSEND	Schreibt in den Puffer der Kommunikationsoption	COMOPTSEND nr array	nr = Anzahl der Wörter (Senden) array = Name des Arrays (Mindestgröße nr)
COMOPTGET	Liest ein Telegramm der Kommunikationsoption.	COMOPTGET nr array	nr = Anzahl der Wörter (Lesen) array = Name des Arrays (Mindestgröße nr)
PCD	Pseudo-Array für den direkten Zugriff auf den Feldbus-Datenbereich.	PCD[n]	n = Index

□ Befehle zur Drehzahlregelung

Befehle zum permanenten Verfahren der Achse mit konstanter Geschwindigkeit. (Gruppe DRE)

Befehl	Beschreibung	Syntax	Parameter
CSTART	Permanentes Verfahren im Drehzahlmodus starten.	CSTART	–
CSTOP	Antrieb im Drehzahlmodus stoppen.	CSTOP	–
CVEL	Geschwindigkeit für die Drehzahlregelung setzen.	CVEL v	v = Geschwindigkeitswert

□ Positionierbefehle

Befehle zum absoluten und relativen Positionieren der Achse. (Gruppe ABS und REL)

Befehle	Beschreibung	Syntax	Parameter
Absolute Positionierung (ABS)			
ACC	Beschleunigung setzen.	ACC a	a = Beschleunigung
DEC	Negative Beschleunigung setzen.	DEC a	a = Verzögerung
POSA	Achse absolut positionieren.	POSA p	p = Position in BE
VEL	Geschwindigkeit setzen.	VEL v	v = normierter Geschwindigkeitswert
Relative Positionierung (REL)			
ACC	Beschleunigung setzen	ACC a	a = Beschleunigung
DEC	Negative Beschleunigung setzen.	DEC a	a = Verzögerung
POSR	Achse relativ zur Istposition positionieren	POSR d	d = Abstand zur Istposition in BE
VEL	Geschwindigkeit setzen	VEL v	v = normierter Geschwindigkeitswert

□ Synchronisationsbefehle

Befehle zum Synchronisieren der Slaves mit dem Master oder mit der Master-Simulation. (Gruppe SYN)

Befehle	Beschreibung	Syntax	Parameter
DEF SYNCORIGIN	Definiert das Verhältnis Master:Slave für den nächsten SYNCP oder SYNCM Befehl.	DEF SYNCORIGIN master slave	master = Sollposition in qc slave = Sollposition
MOVE SYNCORIGIN	Synchronisationsursprung relativ verschieben.	MOVE SYNCORIGIN mwert	mwert = relativer Offset
PULSACC	Beschleunigung für den virtuellen Master setzen.	PULSACC a	a = Beschleunigung in Hz/s
PULSVEL	Geschwindigkeit für den virtuellen Master setzen.	PULSVEL v	v = Geschwindigkeit in Pulsen pro Sekunde (Hz)
SYNCM	Winkel-/Positionssynchronisation mit Markerkorrektur.	SYNCM	–
SYNCP	Winkel-/Positionssynchronisation.	SYNCP	–
SYNCV	Geschwindigkeitssynchronisation.	SYNCV	–
SYNCSTAT	Flag für Synchronisationsstatus abfragen.	erg = SYNCSTAT	
SYNCSTATCLR	Zurücksetzen der Flags MERR und MHIT.	SYNCSTATCLR value	value = 8 = SYNCMMHIT 16 = SYNCSMHIT 32 = SYNCMMERR 64 = SYNCSMERR

□ CAM-Befehle

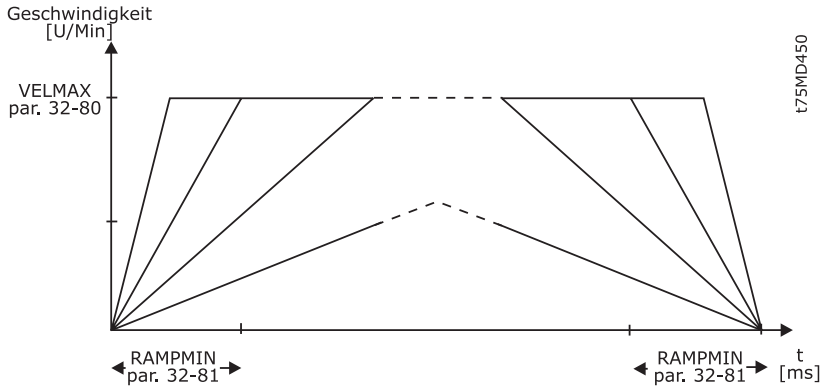
Befehle für die Synchronisation im CAM-Modus (Kurvenscheibensteuerung).

Befehle	Beschreibung	Syntax	Parameter
CURVEPOS	Slave-Position, die der aktuellen Master-Position der Kurve entspricht, abfragen.	erg = CURVEPOS	_
DEFMCPOS	Anfangsposition des Masters definieren.	DEFMCPOS p	p = Position in MU
POSA CURVEPOS	Slave auf die, der Master-Position entsprechenden Kurvenposition fahren.	POSA CURVEPOS	_
SETCURVE	CAM-Kurve setzen	SETCURVE array	array = Array oder Kurvenname
SYNCC	Synchronisation im CAM-Modus.	SYNCC num	num = Anzahl der Kurven, die ausgeführt werden (0 = Antrieb bleibt im CAM-Modus)
SYNCCMM	Synchronisation im CAM-Modus mit Markerkorrektur des Masters.	SYNCCMM num	wie oben
SYNCCMS	Synchronisation im CAM-Modus mit Markerkorrektur des Slaves.	SYNCCMS num	wie oben
SYNCCSTART	Slave zur Synchronisation im CAM-Modus starten.	SYNCCSTART pnum	pnum = Start-Stop-Punktepaar Nummer
SYNCCSTOP	Slave nach der CAM-Synchronisation anhalten.	SYNCCSTOP pnum slavepos	pnum = Start-Stop-Punktepaar Nummer slavepos = Slave-Position nach dem Auskuppeln

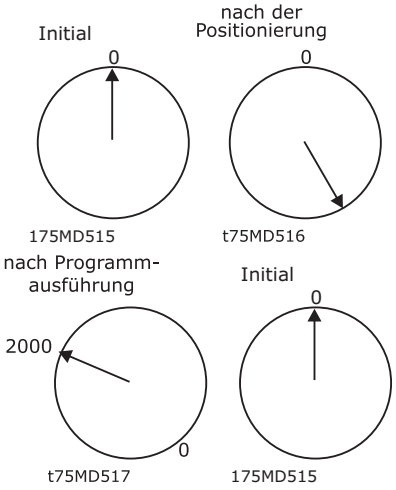
□ Alle Befehle von ACC to #INCLUDE

Im folgenden Abschnitt finden Sie alle Befehle in alphabetischer Reihenfolge ausführliche beschrieben mit Syntax-Beispielen sowie kurzen Programmbeispielen.

□ ACC

Kurzinfo	Beschleunigung für Fahrbefehle setzen
Syntax	ACC a
Parameter	a = Beschleunigung
Beschreibung	Der Befehl ACC bestimmt die Beschleunigung für die nächsten Fahrbefehle im Drehzahl-, Positionier- oder Synchronisationsmodus. Der Wert bleibt solange gültig, bis mit einem weiteren ACC Befehl eine neue Beschleunigung gesetzt wird. Der Wert bezieht sich auf die Parameter 32-81 *Kürzeste Rampe* und 32-80 *Maximalgeschwindigkeit* sowie 32-83 *Geschwindigkeitsteiler*.
	ACHTUNG!: Wurde vor einem Fahrbefehl noch keine Beschleunigung definiert, wird mit dem Default-Wert aus Par. 32-85 *Default-Beschleunigung* beschleunigt.
	ACHTUNG!: Wenn die MCO 305 zum Steuern des FC 300 benutzt wird, sollten die Rampen immer über die Optionskarte eingestellt werden und nicht im FC 300. Die FC 300-Rampen müssen dabei immer auf Minimum stehen.
Befehlsgruppe	REL, ABS
Querverweise	DEC, VEL, POSA, POSR, Parameter: 32-81 *Kürzeste Rampe*, 32-80 *Maximalgeschwindigkeit*, 32-83 *Geschwindigkeitsteiler*
Syntax-Beispiel	ACC 10 /* Beschleunigung 10 */
Beispiel	Minimale Beschleunigungszeit: 1000 ms Maximale Geschwindigkeit: 1500 U/Min (25 U/s) Geschwindigkeitsteiler: 100
	
Programmbeispiel	ACC_01.M

☐ APOS

Kurzinfo	Aktuelle Position einer Achse abfragen
Syntax	erg = APOS
Rückgabewert	erg = Istposition in Benutzereinheiten (BE) absolut zum aktuellen Nullpunkt Wegangaben in Fahrbefehlen erfolgen immer in Benutzereinheiten und werden intern in Quadcounts umgerechnet. (Siehe auch Benutzerfaktor Zähler und Nenner in Parameter 32-12 und 32-11.) Die Benutzereinheit (BE) entspricht in der Standardeinstellung der Anzahl Quadcounts: $\text{Parameter} = \frac{\text{Par. 32 - 12 Benutzereinheit Zähler}}{\text{Par. 32 - 11 Benutzereinheit Nenner}} = 1$
Beschreibung	Der Befehl APOS kann die Position der Achse absolut zum aktuellen Nullpunkt abfragen. **ACHTUNG!:** Wenn ein mit SET ORIGIN gesetzter und aktiver Temporärnullpunkt existiert, bezieht sich der Positionswert auf diesen Nullpunkt. **ACHTUNG!:** Das Ergebnis muss nicht der Ziel- oder Sollposition entsprechen, wenn man mit APOS die Position abfragt. Es können sich Fehler oder Abweichungen durch die Mechanik und den abgerundeten Dezimalstellen in den Benutzereinheiten ergeben. APOS wird von den Parametern 32-12 und 32-11 sowie den Befehlen SET ORIGIN p und DEF ORIGIN beeinflusst. Beispiel: POSA 2000 PRINT "Istposition erreicht", APOS Ausgabe: Istposition erreicht 2000 (abhängig von den PID Einstellungen könnte eine kleine Abweichung auftreten) Beispiel mit SET ORIGIN SET ORIGIN 2000 POSA 2000 PRINT "Istposition", APOS Ausgabe: Istposition 2000  Beim Programmstart wird in diesem Beispiel die absolute Position 2000 qc als Startposition festgelegt; dann wird der Antrieb um 2000 qc entsprechend dem Positionierbefehl weiter gefahren.
Befehlsgruppe	I/O
Querverweise	CPOS, DEF ORIGIN, SET ORIGIN, POSA, POSR Parameter: 32-12 *Benutzerfaktor Zähler*, 32-11 *Benutzerfaktor Nenner*
Syntax-Beispiel	PRINT APOS /* Istposition der Achse am PC ausgeben */
Programmbeispiel	APOS_01.M, GOSUB_01.M, MOTOR_01.M

☐ AVEL

Kurzinfo	Aktuelle Geschwindigkeit der Achse abfragen.
Syntax	erg = AVEL
Rückgabewert	erg = aktuelle Geschwindigkeit der Achse in BE/s; Wert mit Vorzeichen
Beschreibung	Diese Funktion liefert die aktuelle Geschwindigkeit der Achse in Benutzereinheiten pro Sekunde (BE/s) zurück. Die Genauigkeit der Werte hängt von der Messdauer (Mittelung) ab. Diese ist standardgemäß auf 20 ms eingestellt, kann aber vom Anwender mit dem _GETVEL Befehl verändert werden. Es genügt den Befehl einmal aufzurufen, um von da an mit einer anderen Messzeit zu arbeiten. So stellt der Befehl: var = _GETVEL 100 die Messdauer auf 100 ms ein, so dass man bei AVEL und MAVEL eine wesentlich bessere Auflösung der Geschwindigkeit erhält, schnelle Änderungen dagegen erst mit einer Verzögerung von maximal 100 ms.
Befehlsgruppe	I/O
Querverweise	MAVEL, APOS, _GETVEL
Syntax-Beispiel	PRINT AVEL /* aktuelle Geschwindigkeit der Achse am PC ausgeben */

☐ AXEND

Kurzinfo Status der Programmausführung abfragen.

Syntax erg = AXEND

Rückgabewert erg = Achsstatus mit folgender Bedeutung:

Wert	Bit	
128	7	1 = Motor ist zurückgesetzt (reset), d.h. er ist startbereit und regelt wieder, z.B. nach ERRCLR, MOTOR STOP, MOTOR ON
64	6	1 = Lageregelung ist abgeschaltet, Motor ist aus
	4 - 5	nicht verwendet
8	3	1 = Motor ist im Zustand STOP
4	Bit 2	1 = Drehzahlmodus ist aktiv
2	Bit 1	1 = Positioniervorgang ist aktiv
1	Bit 0	1 = Zielposition erreicht; Motor im Stillstand

Beschreibung Der Befehl AXEND liefert den aktuellen Status der Achse bzw. den Stand der Programmausführung.

Damit können Sie zum Beispiel abfragen, wann die „Position erreicht" ist und ein Positionierbefehl (POSA, POSR) wirklich abgeschlossen ist. Wenn Bit 1 auf [0] gesetzt ist, ist der Positioniervorgang abgeschlossen und die Position erreicht.

Wenn aber der Positionierbefehl mit MOTOR STOP unterbrochen wurde und später mit CONTINUE fortgesetzt wird, dann würden folgende Bits auf [1] gesetzt sein:

das Bit 0 für „Motor ist im Stillstand"
das Bit 1 für „Positioniervorgang aktiv"
das Bit 3 für „Motor ist im Zustand STOP"
das Bit 6 für „Lageregelung abgeschaltet"

Der Befehl AXEND eignet sich besonders um im NOWAIT ON Zustand festzustellen, ob eine Bewegung abgeschlossen ist.

Befehlsgruppe I/0

Querverweise WAITAX, STAT, NOWAIT

Syntax-Beispiel

```
NOWAIT ON              // nicht warten bis Position erreicht ist
POSA 100000
WHILE (AXEND&2) DO
// Solange Positioniervorgang aktiv, Schleife wiederholen
    IF IN1 THEN        // wenn Eingang 1 gesetzt ist
       VEL 100         // Geschwindigkeit erhöhen
       POSA 100000
       WAIT IN1 OFF    // warten, bis Taste losgelassen
    ENDIF
ENDWHILE               // Position erreicht
```

Syntax-Beispiel

```
IF (AXEND&64) THEN
    OUT 1 1        // Ausgang 01 setzen, wenn Lageregelung abgeschaltet
ELSE
    OUT 1 0
ENDIF
```

Programmbeispiel AXEND_01.M

□ COMOPTGET

Kurzinfo	Liest ein Telegramm der Kommunikationsoption.
Syntax	COMOPTGET anz array
Parameter	array = der Name eines Arrays, das mindestens die Größe 'anz' haben muss anz = Anzahl Datenworte, die gelesen werden sollen
Beschreibung	COMOPTGET liest aus dem Puffer der Kommunikationsoption 'anz' Datenworte aus und schreibt sie in das Array 'array' beim ersten Element beginnend.
Kompatibilität	Mit eingebauter Kommunikationsoption.
Kommunikations-option	Funktion der Kommunikationsoption: Parameter: Die Lese- und Schreibparameter werden von der Optionskarte nicht verändert.
	ACHTUNG!: Die Parameter 9-15 und 9-16 müssen zusätzlich mit den richtigen Werten gesetzt werden.
Kontrolldaten	Die Funktion des Steuerwortes (STW) und der Hauptsollwertes (HSW) hängt davon ab, wie der Par. 33-82 *Statusüberwachung Antrieb gesetzt ist*; das Zustandswort (ZSW) und der Hauptistwert (HIW) sind immer aktiv.

	Parameter 33-82 „MCO 305 EIN”	Parameter 33-82 „MCO 305 AUS”
STW/HSW	nicht aktiv	aktiv
ZSW/HIW	aktiv	aktiv

Prozessdaten	PCD's 1 – 4 von PPO Typ 2/ 4 und PCD's 1 – 8 von PPO Typ 5 sind nicht mit einer Parameternummer 9-15 und 9-16 festgelegt, sondern können frei in einem APOSS-Programm benutzt werden. Der Befehl COMOPTGET kopiert die empfangenen Daten der Kommunikationsoption in ein Array, in dem jedes Array-Element ein Datenwort (16 Bit) enthält. Der Befehl COMOPTSEND kopiert die Daten von einem Array, in dem jedes Array Element ein Datenwort (16 Bit) enthält, in einen Sendepuffer der Kommunikations-option, von dem es via Netzwerk zum Master gesendet wird.
Befehlsgruppe	Kommunikationsoption
Querverweis	COMOPTSEND
Programmbeispiel	COM_OPT

□ COMOPTSEND

Kurzinfo	Schreibt in den Puffer der Kommunikationsoption.
Syntax	COMOPTSEND anz array
Parameter	array = Name eines Arrays, das mindestens die Größe 'anz' haben muss anz = Anzahl der Datenworte, die gesendet werden sollen
Beschreibung	COMOPTSEND schreibt in den Puffer der Kommunikationsoption. Dabei werden aus 'array' die ersten 'anz' Datenworte gesendet.
Kompatibilität	Mit eingebauter Kommunikationsoption.
Kommunikations-option	Funktion der Kommunikationsoption: Siehe COMOPTGET Befehl
Befehlsgruppe	Kommunikationsoption
Querverweis	COMOPTGET
Programmbeispiel	COM_OPT

□ CONTINUE

Kurzinfo	Abgebrochene Positionier- und Drehzahlbefehle fortsetzen.
Syntax	CONTINUE
Beschreibung	Mit dem Befehl CONTINUE können Positionier- und Drehzahlbefehle, die durch den Befehl MOTOR STOP oder einen Fehlerzustand abgebrochen oder mit MOTOR OFF angehalten wurden, fortgesetzt werden. CONTINUE kann besonders in einem Fehlerunterprogramm in Verbindung mit dem Befehl ERRCLR eingesetzt werden, um nach einem Fehlerabbruch den Bewegungsablauf korrekt weiterzuführen.
	**ACHTUNG!:** CONTINUE setzt aber nicht abgebrochene Synchronisationsbefehle fort.
Befehlsgruppe	CON
Querverweise	MOTOR STOP, ERRCLR, ON ERROR GOSUB
Syntax-Beispiel	CONTINUE /* Unterbrochene Bewegungsvorgänge fortsetzen */
Programmbeispiel	MSTOP_01.M

□ CPOS

Kurzinfo	Aktuelle Sollposition einer Achse abfragen.
Syntax	erg = CPOS
Rückgabewert	erg = Absolute Sollposition in Benutzereinheiten (BE) bezogen auf den aktuellen Nullpunkt
Beschreibung	Mit dem Befehl CPOS kann die aktuelle Sollposition einer Achse absolut zum aktuellen Nullpunkt abgefragt werden. Unter der Sollposition versteht man die temporäre Sollposition, die durch die Lageregelung während eines Positioniervorgangs oder einer Bewegung im Drehzahlmodus jede ms neu berechnet wird. Die Sollposition kann unabhängig vom Betriebszustand (Lageregelung im Stillstand, Positioniervorgang, Drehzahlregelung oder Synchronisation) abgefragt werden.
	**ACHTUNG!:** Wenn ein mit SET ORIGIN gesetzter und aktiver Temporärnullpunkt existiert, ist der Positionswert auf diesen Nullpunkt bezogen.
Befehlsgruppe	I/O
Querverweise	APOS, DEF ORIGIN, SET ORIGIN, POSA, POSR, Parameter: 32-12 *Benutzerfaktor Zähler*, 32-11 *Benutzerfaktor Nenner*
Syntax-Beispiel	PRINT CPOS /* aktuelle Sollposition der Achse */
Programmbeispiel	CPOS_01.M, GOSUB_01.M

□ CSTART

Kurzinfo	Starten des Drehzahlmodus.
Syntax	CSTART
Beschreibung	Mit dem Befehl CSTART wird ein drehzahlgeregelter Fahrbefehl gestartet. Die Beschleunigungsrampe sowie die Drehzahl sollte vor dem Starten des Drehzahlmodus mit den Befehlen ACC, DEC und CVEL festgelegt werden. CSTART enthält nicht den Befehl MOTOR ON der die Motorregelung einschaltet. Nach vorangegangenem MOTOR OFF ist bei Verwendung von CSTART also ein explizites Aufrufen von MOTOR ON notwendig.  **ACHTUNG!:** Wenn zum Zeitpunkt des CSTART noch kein Drehzahlwert mit CVEL definiert wurde, wird die Default-Geschwindigkeit 0 verwendet. Der Motor dreht sich nicht, die Lageregelung ist aber aktiv. Alle nach dem Start des Drehzahlmodus folgenden CVEL Befehle werden sofort ausgeführt: Es wird sofort eine entsprechende Drehzahlanpassung mit der durch ACC bzw. DEC definierten Beschleunigungs- bzw. Bremsrampe vorgenommen.
Befehlsgruppe	DRE
Querverweise	ACC, DEC, CVEL, CSTOP
Syntax-Beispiel	CSTART /* Drehzahlmodus starten */
Programmbeispiel	CMODE_01.M

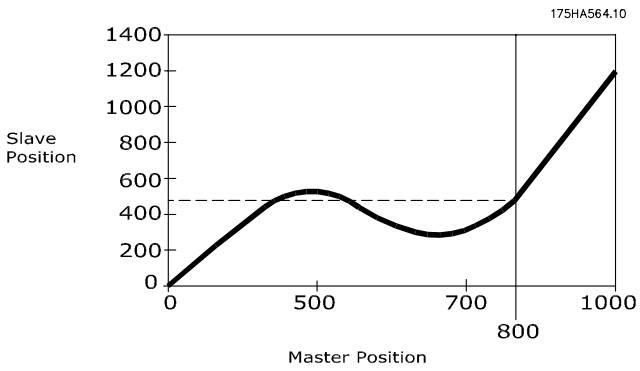
□ CSTOP

Kurzinfo	Stoppen des Antriebs im Drehzahlmodus.
Syntax	CSTOP
Beschreibung	Mit dem CSTOP Befehl wird der Modus Drehzahlregelung verlassen und in den Positioniermodus geschaltet. Dabei wird eine noch drehende Achse mit der durch DEC definierten Verzögerung abgebremst und der Motor in der Stopp-Position angehalten.  **ACHTUNG!:** Ein im Positioniermodus ausgeführter CSTOP Befehl führt ebenfalls zu einem abrupten Abbrechen des Positioniervorgangs.
Befehlsgruppe	DRE
Querverweise	ACC, DEC, CVEL, CSTART
Syntax-Beispiel	CSTOP /* Drehzahlmodus stoppen */
Programmbeispiel	CMODE_01.M

□ CURVEPOS

Kurzinfo	Der aktuellen Master-Position entsprechende Slave-Kurvenposition abfragen.
Syntax	erg = CURVEPOS
Rückgabewert	erg = Slave-Position in CAM-Einheiten (BE) absolut zum aktuellen Nullpunkt.
Beschreibung	Mit dem Befehl CURVEPOS kann die Slave-Kurvenposition, die der aktuellen Master-Position entspricht, abgefragt werden. Die Position kann unabhängig vom Betriebszustand (Lageregelung im Stillstand, Positioniervorgang, Drehzahlregelung oder Synchronisation) abgefragt werden. CMASTERCPOS (SYSVAR) und CURVEPOS werden aktualisiert, auch wenn SYNCC nicht weiter aktiv ist. Die Aktualisierung dieser Werte beginnt nach einem SETCURVE Befehl (wenn Par. 33-23 *Startverhalten für Sync* ist < 2000) oder nach SYNCC und den ersten Master-Marker (wenn Par. 33-23 = 2000). Die Aktualisierung wird also auch nach dem Stoppen des SYNCC Befehls fortgeführt, wenn Par. 33-23 *Startverhalten für Sync.* < 2000. **ACHTUNG!:** Die Position ist nur definiert, wenn zuvor ein SETCURVE gesetzt wurde. **ACHTUNG!:** Wenn ein mit SET ORIGIN gesetzter und aktiver Temporärnullpunkt existiert, bezieht sich der Positionswert auf diesen Nullpunkt. **ACHTUNG!:** DEFMCPOS und DEFMORIGIN können diese Position noch verändern.
Befehlsgruppe	CAM
Querverweise	APOS, DEF ORIGIN, SET ORIGIN, POSA, POSR, DEFMCPOS, Parameter: 33-10 *Sync-Faktor Master*, 33-11 *Sync-Faktor Slave*
Syntax-Beispiel	PRINT CURVEPOS // aktuelle Slave-Position der Kurve ausgeben
Beispiel	Fixpunkte einer Kurve:

Master	Slave
0	0
500	500
700	300
1000	1200



Wir nehmen an, dass die aktuelle Master-Position 800 sei. Dann gibt CURVEPOS die entsprechende Slave-Position von 450 aus.

Fall 1: Istposition Master ist 800 und Istposition Slave ist 200.
CURVEPOS gibt den Wert 450 aus.

Fall 2: Istposition Master ist 800 und Istposition Slave ist 700.
CURVEPOS gibt den Wert 450 aus.

Also ist CURVEPOS unabhängig von der Slave-Position.

□ CVEL

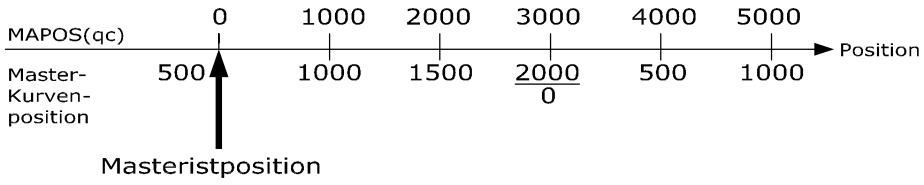
Kurzinfo	Geschwindigkeit für drehzahlgeregelte Motorbewegungen setzen.
Syntax	CVEL v
Parameter	v = Normierter Geschwindigkeitswert (negativ für andere Drehrichtung) $\text{Sollgeschwindigkeit [U/Min]} = V * \frac{\text{Par. 32-80 Maximalgeschwindigkeit}}{\text{Par. 32-83 Geschwindigkeitsteiler}}$
Beschreibung	Mit dem CVEL Befehl wird die Geschwindigkeit für die nächsten drehzahlgeregelten Motorbewegungen gesetzt. Der Wert bleibt solange gültig bis mit einem weiteren CVEL Befehl eine neue Geschwindigkeit gesetzt wird. Der zu übergebende Geschwindigkeitswert bezieht sich auf die Parameter 32-80 *Maximalgeschwindigkeit* und 32-83 *Geschwindigkeitsteiler*.  **ACHTUNG!:** CVEL Befehle, die nach einem CSTART folgen, werden sofort ausgeführt, das heißt die Geschwindigkeit wird mit der durch ACC/DEC vorgegebenen Beschleunigung bzw. Verzögerung auf den mit CVEL übergebenen Wert angepasst. Wurde vor dem Starten des Drehzahlmodus (CSTART) noch keine Geschwindigkeit definiert, beträgt die Standardvorgabe 0. Der Motor dreht sich nicht und erst eine Geschwindigkeitsvorgabe mit CVEL startet die Bewegung im Drehzahlmodus.
Befehlsgruppe	DRE
Querverweise	ACC, DEC, CSTART, CSTOP; Parameter: 32-80 *Maximalgeschwindigkeit*
Syntax-Beispiel	CVEL 100
Programmbeispiel	CMODE_01.M

□ DEC

Kurzinfo	Verzögerung (negative Beschleunigung) setzen.
Syntax	DEC a
Parameter	a =Verzögerung
Beschreibung	Mit dem Befehl DEC bestimmen Sie die Verzögerung (negative Beschleunigung) für die nächsten Fahrbefehle im Drehzahl-, Positionier- oder Synchronisationsmodus. Der Wert bleibt solange gültig, bis mit einem weiteren Befehl DEC eine neue Verzögerung gesetzt wird. Der Wert bezieht sich auf die Parameter 32-81 *Kürzeste Rampe* und 32-80 *Maximalgeschwindigkeit* sowie 32-83 *Geschwindigkeitsteiler*.  **ACHTUNG!:** Wurde vor einem Positionierbefehl noch keine Verzögerung definiert, wird mit dem in Parameter 32-85 *Default-Beschleunigung* vorgegebenen Wert abgebremst.  **ACHTUNG!:** Wenn Sie mit MCO 305 arbeiten, dann sollten Sie immer die Rampen mittels der Optionskarte setzen und nicht im FC 300. Die FC Rampen müssen dabei immer auf Minimum stehen.
Befehlsgruppe	REL, ABS
Querverweise	ACC; Parameter: 32-81 *Kürzeste Rampe*, 32-80 *Maximalgeschwindigkeit*, 32-83 *Geschwindigkeitsteiler*
Syntax-Beispiel	ACC 50 /* Beschleunigung: 50, beim Bremsen 10 */ DEC 10
Beispiel	Kürzeste Rampe: 1000 ms Maximale Geschwindigkeit: 1500 U/Min Geschwindigkeitsteiler: 100

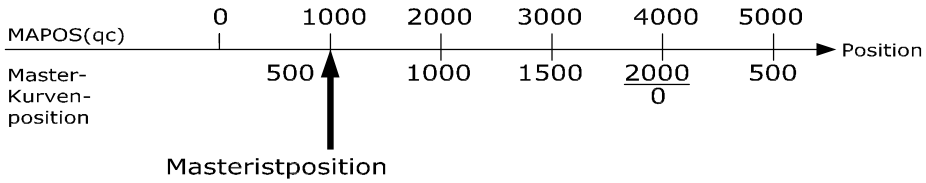
□ DEFMCPOS

Kurzinfo	Anfangsposition des Masters definieren.
Syntax	DEFMCPOS p
Parameter	p = Position in Benutzereinheiten (MU)
Beschreibung	DEFMCPOS definiert die Anfangsposition des Masters (in MU) im CAM-Modus und somit, wo die Kurve startet, sobald die Masterpulse gezählt werden.
Befehlsgruppe	CAM
Querverweise	DEFMORIGIN, SETMORIGIN, SYNCC, Parameter: 33-23 *Startverhalten für Sync.*
Syntax-Beispiel	DEFMCPOS 1000 // internen MU-Zähler auf 1000 setzen
Beispiel	DEFMCPOS positioniert die physikalische Ist-Master-Position auf die angegebene Master-Kurvenposition ungeachtet dessen, was im MAPOS Befehl steht.



175HA560.10

Wenn ein DEFMCPOS 500 begonnen ist, wird die physikalische Position des Masters als Position 500 der Kurve definiert.



175HA561.10

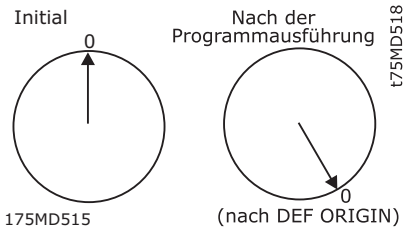
Wenn ein DEFMCPOS 500 begonnen ist, wird die physikalische Position des Masters als Position 500 der Kurve definiert.

□ DEFMORIGIN

Kurzinfo	Aktuelle Master-Position als Nullpunkt für den Master setzen.
Syntax	DEFMORIGIN
Beschreibung	DEFMORIGIN definiert die aktuelle Master-Position als Nullpunkt für den Master. Die Master-Position (MAPOS) bezieht sich bis zu einer erneuten Definition mit DEFMORIGIN oder SETMORIGIN auf diesen Nullpunkt.
	ACHTUNG!: Der Befehl DEFMORIGIN kann bei Einsatz von Absolutgebern (siehe Par. 32-30 *Inkrementalgeber Signaltyp*) nicht verwendet werden.
Befehlsgruppe	INI
Querverweise	MAPOS, SETMORIGIN
Syntax-Beispiel	DEFMORIGIN /* Nullpunkt für Master definieren. */

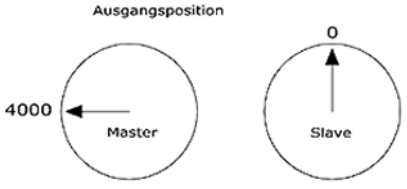
□ DEF ORIGIN

Kurzinfo	Istposition als Nullpunkt setzen.
Syntax	DEF ORIGIN
Beschreibung	Mit dem DEF ORIGIN Befehl wird die Istposition als Nullpunkt gesetzt. Alle absoluten Positionierbefehle (POSA etc.) beziehen sich fortan auf diesen Nullpunkt. Die Istposition, die in einem Positionierbefehl erreicht wird, ist die Zielposition plus möglicher Fehler, die nicht automatisch kompensiert werden, während DEF ORIGIN ausgeführt wird.
	ACHTUNG!: Der Befehl DEF ORIGIN kann bei Einsatz von Absolutgebern (siehe Par. 32-00 *Inkrementalgeber Signaltyp*) nicht verwendet werden.
Befehlsgruppe	INI
Querverweise	POSA
Syntax-Beispiel	POSA 80000 /* Absolut positionieren */ DEF ORIGIN /* Istposition als Nullpunkt definieren */
Beispiel	POSA 2000 PRINT "Position vor neuem Nullpunkt", APOS DEF ORIGIN PRINT "Position nachher", APOS Output Position vor neuem Nullpunkt 2000, Position danach 0
Programmbeispiel	DORIG_01.M, ORIG_01.M



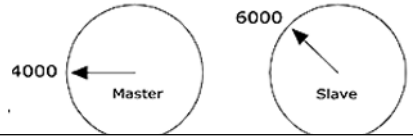
□ DEF SYNCORIGIN

Kurzinfo	Definiert das Verhältnis Master:Slave für den nächsten SYNCP oder SYNCM Befehl.
Syntax	DEFSYNCORIGIN master slave
Parameter	master = Sollposition in qc slave = Sollposition
Beschreibung	Dieser Befehl definiert, wie viel Abstand vor oder nach dem Slave im Verhältnis zur Masterposition sein soll. Damit kann das Verhältnis zwischen Master und Slave für den nächsten SYNCP oder SYNCM Befehl definiert werden. Er setzt die interne Slave-Sollposition auf den Wert des Slaves. Der Wert des Masters wird für einen internen MOVE SYNCORIGIN benutzt. Dafür wird ein MOVESYNCORIGN durch diesen Befehl überschrieben. Beide Aktionen werden in dem Moment ausgeführt, wenn der SYNC Befehl aktiviert wird. Das garantiert, dass Master und Slave auf die o.g. Master-Slave-Position synchronisiert werden.
Befehlsgruppe	SYN
Querverweise	MOVESYNCORIGIN
Beispiel	In diesem Beispiel soll der Slave auf Position 4000 qc sein, wenn der Master auf 2000 qc ist, d.h. der Slave sollte einen Vorsprung von 2000 qc auf den Master haben. Ebenso soll der Slave auf Position 5000 qc sein, wenn der Master auf 3000 qc ist.



Befehl; DEFSYNCORIGIN 2000 4000

Der Slave wird auf eine Position um 2000 qc vor dem Master korrigiert.



□ DELAY

Kurzinfo	Zeitverzögerung
Syntax	DELAY t
Parameter	t = Verzögerungszeit in Millisekunden (maximal MLONG)
Beschreibung	Der DELAY Befehl führt zu einer definierten Programmverzögerung. Der Übergabeparameter gibt dabei die Verzögerungszeit in Millisekunden an. Wenn während der Verzögerungszeit ein Interrupt auftritt, wird nach dem Abarbeiten der Interrupt-Prozedur der Wartevorgang mit der restlichen Verzögerungszeit fortgesetzt. Der DELAY Befehl führt somit zu einer konstanten Wartezeit unabhängig davon, ob verschiedene Interrupts während der Verweilzeit behandelt werden mussten. Nimmt der Interrupt mehr Verarbeitungszeit in Anspruch als restliche Verweilzeit zur Verfügung steht, wird die Interrupt-Prozedur zu Ende abgearbeitet, bevor mit dem auf die DELAY Anweisung folgenden Befehl das Programm fortgesetzt wird.
Befehlsgruppe	CON
Querverweise	WAITT, WAITI, WAITAX
Syntax-Beispiel	DELAY 1000 /* 1 Sekunde verzögern */
Programmbeispiel	DELAY_01.M

□ DELETE ARRAYS

Kurzinfo	Alle Arrays im RAM löschen.
Syntax	DELETE ARRAYS
Beschreibung	Mit DELETE ARRAYS können Sie alle Arrays im RAM löschen, ohne auch die Parameter etc. zu löschen. Dieser Befehl bewirkt das Gleiche, wie der Menübefehl *Steuerung* → *Reset* → *Arrays*. **ACHTUNG!:** Wenn Sie anschließend ein SAVE ARRAYS durchführen, werden auch die Arrays im EEPROM überschrieben! **ACHTUNG!:** Falls DELETE ARRAYS nach einer DIM Anweisung im Programm durchgeführt wird, darf danach nicht mehr auf die Array-Elemente zugegriffen werden. **ACHTUNG!:** Wenn ein Programm einen DELETE ARRAYS Befehl enthält, gibt es nach Verlassen des Programms im RAM keine Arrays mehr.
Befehlsgruppe	INI

□ DIM

Kurzinfo	Definition eines Arrays
Syntax	DIM array [n]
Parameter	array = Name des Arrays n = Anzahl der Array-Elemente
Beschreibung	Mit einer DIM Anweisung am Programmanfang vereinbaren Sie die Verwendung von ein oder mehreren Arrays (= Variablenfeldern). Arrays besitzen Gültigkeit für alle in der Steuerung abgelegten Programme. Sollten noch keine Arrays im Speicher der Steuerung vorhanden sein, werden durch die DIM Anweisung die Arrays angelegt. Bei bereits im Speicher vorhandenen Arrays wird überprüft, ob deren Größe mit der aktuellen DIM Anweisung übereinstimmt. Sollten hierbei Unterschiede auftreten, wird eine Fehlermeldung ausgegeben. Wenn zusätzlich zu den übereinstimmenden Arrays noch weitere neue Arrays erklärt sind, müssen diese an das Ende der DIM Anweisung angefügt werden. Auf jedes Array-Element kann später ähnlich wie auf eine Variable zugegriffen und es können Rechenergebnisse, Zeichen oder andere Informationen abgelegt werden. Ein Array-Element wird über den Array-Namen und einen Index angesprochen. Die Indizes sind dabei von 1 bis zu der in der DIM Anweisung definierten Größe zulässig. Ein wesentlicher Unterschied zwischen Variablen und Array-Elementen besteht jedoch darin, dass Arrays im nicht flüchtigen Speicherbereich abgelegt sind und ihr Inhalt – sofern mit SAVEPROM oder SAVE ARRAYS gesichert – auch beim Abschalten der Versorgungsspannung erhalten bleibt. Im Gegensatz zu Variablen besitzen Arrays nicht nur für ein Programm, sondern für alle in der Steuerung abgelegten Programme Gültigkeit. Einzige Voraussetzung dafür ist, dass die Arrays mit einer DIM Anweisung in den gewünschten Programmen zugänglich gemacht werden, wodurch ein Datenaustausch zwischen mehreren Programmen möglich wird. Es spielt hierbei keine Rolle, ob das Array in allen Programmen durch den gleichen Namen gekennzeichnet ist. Entscheidend ist lediglich die Reihenfolge der Array-Definitionen. Dadurch greift das erste definierte Array in allen Programmen immer auf das erste im Speicher abgelegte Array zu, unabhängig vom Array-Namen.  **ACHTUNG!:** Die DIM Anweisung muss die erste Anweisung in einem Programm sein und noch vor dem Unterprogrammbereich stehen! Indizes sind von 1 bis zur Größe des definierten Arrays erlaubt. Eine einmal definierte Array-Größe gilt für alle Programme und kann nicht geändert werden. Einzig die Reihenfolge der Array-Definition (und nicht der Namen) bestimmt, auf welche Datenfelder zugegriffen wird. Array-Definitionen können nur durch das Löschen des gesamten Speichers rückgängig gemacht werden.
Befehlsgruppe	CON
Syntax-Beispiel	DIM xpos[100], ypos[100] /* Array xpos und ypos mit je 100 Elementen definieren */
Programmbeispiel	DIM_01.M

☐ DISABLE ... interrupts

Kurzinfo	Sperrt die Ausführung von Interrupts.
Syntax	DISABLE inttyp
Parameter	inttyp = ALL, INT, COMBIT, STATBIT, PARAM, PERIOD, TIME, POSINT

ACHTUNG!:
Die Ausführung der Fehlerbehandlung (ON ERROR) kann mit DISABLE nicht gesperrt werden. Der Fehler-Interrupt hat höchste Priorität und unterbricht auch andere aktive Interrupts.

Beschreibung

DISABLE schaltet alle oder explizit genannte Interrupts – außer ON ERROR – ab. Wenn die Funktion DISABLE … im Hauptprogramm verwendet wird, kann sie Interrupts der entsprechenden Art verhindern.

Dies ist insbesondere nützlich, wenn eine Variable, die in einer Interrupt-Prozedur gesetzt ist, im Hauptprogramm verwendet wird. Dazu sollten Sie im Hauptprogramm zunächst die entsprechenden (oder alle) Interrupts mit DISABLE … abschalten, die Variable ändern und anschließend die entsprechenden (oder alle) Interrupts mit ENABLE … wieder einschalten.

ACHTUNG!:
Wird ein Interrupt disabled (d.h. gesperrt) existiert er weiterhin, wird aber nicht mehr ausgeführt. (Ausnahme: DISABLE ALL).

Die Erkennung läuft weiter im Hintergrund und die Interrupt-Anforderung wird im Fall eines nicht (!) flankengetriggerten oder nachrichtenorientierten Interrupts (ON PERIOD, ON APOS, ON PARAM, etc.) gespeichert. Wenn der Interrupt dann wieder enabled (d.h. freigegeben) wird und es zuvor einen noch nicht ausgeführten, gespeicherten (nicht flankengetriggerten) Interrupt gab, wird dieser Interrupt sofort ausgeführt.

Im Fall eines flankengetriggerten Interrupts (z.B. ON INT, ON COMBIT, ON STATBIT), werden alle Interrupts, die während der DISABLE-Phase stattgefunden haben, nicht ausgeführt, auch dann nicht, wenn wieder auf ENABLE umgeschaltet wird. Diese Interrupts werden im Status DISABLE nicht gespeichert. Flankengetriggerte Interrupts, die nach dem erneuten ENABLE stattfinden, werden weiterhin wieder ausgeführt.

ACHTUNG!:
Ausnahme: DISABLE ALL

Während bei dem selektiven Sperren flankengetriggerter Interrupts (z.B. DISABLE INT) diese Interrupts, wie beschrieben, ignoriert und auch nach der Freigabe nicht mehr ausgeführt werden, wird bei DISABLE ALL die Anforderung (auch von flankengetriggerten Interrupts) gespeichert und der Interrupt nach der Freigabe (ENABLE ALL) noch ausgeführt!

<u>DISABLE ALL in Kombination mit selektivem DISABLE</u>

Hierbei ist zu beachten, dass das ENABLE ALL keine Auswirkung auf gleichzeitig noch gültige selektive Sperrungen hat (z.B. durch DISABLE INT). Eine selektive Sperrung muss somit auch wieder durch das entsprechende selektive ENABLE aufgehoben werden!

Interrupt-Behandlung im Interrupt

Während der Ausführung eines Interrupt-Unterprogramms wird automatisch intern zuerst ein DISABLE ALL ausgeführt. Dies sperrt die Ausführung aller weiterer Interrupts, speichert deren Anforderung jedoch. Am Ende des „aktuellen" Interrupt-Unterprogramms wird wiederum automatisch ein ENABLE ALL ausgeführt. Mit dem Abschluss des „aktuellen" Interrupts werden dann die anstehenden, gespeicherten Interrupts noch ausgeführt. Die Ausführung der Befehle DISABLE ALL und ENABLE ALL ist somit innerhalb eines Interrupts nicht notwendig und nicht sinnvoll.

Das selektive Sperren einzelner Interrupts innerhalb eines Interrupt-Unterprogramms kann jedoch in Abhängigkeit von der Anwendung sinnvoll und erforderlich sein. Falls zum Beispiel während der Ausführung eines Interrupts keine weiteren flankengetriggerten Interrupts akzeptiert und auch nicht gespeichert werden sollen, ist ein gezieltes Sperren der Interrupt-Quelle (z.B. mit DISABLE INT) möglich. In diesem Fall muss der selektive Interrupt später (z.B. mit ENABLE INT) wieder durch das Applikationsprogramm (z.B. am Ende des aktuellen Interrupt-Unterprogramms) freigegeben werden, um die Ausführung entsprechender Interrupt-Anforderungen künftig wieder zu ermöglichen. Alle flankengetriggerten Interrupts, die zwischen dem entsprechenden selektiven DISABLE und ENABLE eingetroffen sind, werden ignoriert und (auch später) nicht mehr ausgeführt. Alle Interrupts, die vor der selektiven Sperrung (z.B. DISABLE INT) oder nach der erneuten selektiven Freigabe (z.B. ENABLE INT) eingetroffen sind, werden nach Abschluss des „ersten" Interrupts abgearbeitet.

Befehlsgruppe INT

Querverweis ON INT, ON COMBIT, ON STATBIT, ON PARAM, ON PERIOD, ON TIME, ENABLE .. Interrupts

Syntax-Beispiel

```
DISABLE ALL          /* Alle Interrupts abschalten */
DISABLE STATBIT      /* Interrupt für Statusbit abschalten */
```

☐ ENABLE ... interrupts

Kurzinfo	Gibt gesperrte Interrupts wieder frei.
Syntax	ENABLE inttyp
Parameter	inttyp = ALL INT COMBIT STATBIT PARAM PERIOD TIME POSINT (ON APOS, ON MAPOS, ON MCPOS)
Beschreibung	ENABLE schaltet alle oder explizit genannte Interrupts wieder ein.
	**ACHTUNG!:** Während der Ausführung eines Interrupt-Unterprogramms wird automatisch intern zuerst ein DISABLE ALL und am Ende ein ENABLE ALL ausgeführt. Die Ausführung der Befehle DISABLE ALL und ENABLE ALL ist somit innerhalb eines Interrupts nicht notwendig und nicht sinnvoll.
	Weitere Informationen zu Interrupt-Sperrungen und der typ-abhängigen Behandlung nach erneuter Freigabe finden Sie bei dem Befehl DISABLE…
Befehlsgruppe	INT
Querverweis	ON INT, ON COMBIT, ON STATBIT, ON PARAM, ON PERIOD, ON TIME, DISABLE …interrupts
Syntax-Beispiel	ENABLE ALL /* Alle Interrupts einschalten */ ENABLE COMBIT /* Interrupt für Kommunikationsbit einschalten */

☐ ERRCLR

Kurzinfo	Löschen einer Fehlermeldung.
Syntax	ERRCLR Der ERRCLR Befehl sollte nur in einem Unterprogramm zur Fehlerbehandlung eingesetzt werden (siehe ON ERROR GOSUB).
	**ACHTUNG!:** ERRCLR beinhaltet den Befehl MOTOR ON, der die Regelung automatisch wieder einschaltet. (Der Motor wird auf aktueller Position lagegeregelt.)
Beschreibung	Ein Fehler der Optionskarte kann durch einen ERRCLR Befehl gelöscht werden. Voraussetzung ist jedoch, dass die Fehlerursache auch tatsächlich beseitigt wurde, da ansonsten die gleiche Fehlermeldung noch mal auftritt. Wenn zwischenzeitlich ein weiterer, noch nicht behobener Fehler aufgetreten ist, wird nur die erste Fehlermeldung gelöscht. ERRCLR setzt auch FC 300 Meldungen mittels Bit 7 des Steuerworts zurück.
Befehlsgruppe	INI, CON
Querverweise	ON ERROR GOSUB, ERRNO, CONTINUE, MOTOR ON, Warnungen und Fehlermeldungen
Syntax-Beispiel	ERRCLR /* aktuelle Fehlermeldung löschen */
Programmbeispiel	ERROR_01.M, IF_01.M, INDEX_01.M

ERRNO

Kurzinfo	Systemvariable mit der aktuellen Fehlernummer.
Syntax	erg = ERRNO
Beschreibung	ERRNO ist eine Systemvariable, die in allen Programmen verfügbar ist und die aktuelle Fehlernummer enthält. Alle Fehlernummern sind im Abschnitt Warnungen und Fehlermeldungen erläutert. Für den Fall, dass zum Zeitpunkt der Abfrage kein Fehler aufgetreten ist, enthält ERRNO eine 0.
Portabilität	Standardvariable
Befehlsgruppe	I/O
Querverweise	ON ERROR GOSUB, ERRCLR, Warnungen und Fehlermeldungen
Syntax-Beispiel	PRINT ERRNO /* aktuelle Fehlernummer ausgeben */
Programmbeispiel	ERROR_01.M, IF_01.M, INDEX_01.M

EXIT

Kurzinfo	Vorzeitiger Programmabbruch.
Syntax	EXIT
Beschreibung	Der EXIT Befehl beendet ein Programm, wobei aktive Positionierprozesse noch zu Ende ausgeführt werden. Der EXIT Befehl ist besonders für den Einsatz in einer Routine zur Fehlerbehandlung vorgesehen und ermöglicht zum Beispiel bei nicht behebbaren Fehlern einen gezielten Programmabbruch. Ein mit *Autostart* gekennzeichnetes Programm wird nach einem Abbruch mit EXIT automatisch wieder anlaufen, wenn SET PRGPAR = –1.
	ACHTUNG!: Normalerweise sollte ein Programm nur bei schwerwiegenden Fehlern, wie zum Beispiel beim Ansprechen eines Endschalters, abgebrochen werden.
Befehlsgruppe	CON
Querverweise	ON ERROR GOSUB, SET, Parameter: 33-80 *Aktivierte Programmnummer* PRGPAR, Autostart
Syntax-Beispiel	EXIT /* Programmabbruch */
Programmbeispiel	EXIT_01.M, ERROR_01.M

☐ GET

Kurzinfo	Liest einen Parameter.
Syntax	erg = GET par
Parameter	par = Parameterkennung
Rückgabewert	erg = Parameterwert
Beschreibung	GET liest den Wert eines MCO 305 Parameters oder eines Anwendungsparameters. Die Parameter werden mit einer Kennung adressiert, zum Beispiel KPROP für den *Proportionalfaktor* oder POSERR für den *Tolerierten Positionsfehler.* Eine vollständige Liste aller Parameterkennungen finden Sie in der Parameter-Referenz. Anwendungsparameter werden mit einer Nummer der Gruppe 19-** adressiert. Siehe auch Parameter-Referenz für die Details.
Befehlsgruppe	PAR
Querverweise	SET, GETVLT, SETVLT, LINKGPAR, Parameter-Referenz
Syntax-Beispiel	PRINT GET POSLIMIT /* Positive Wegbegrenzung ausgeben */ posdiff = GET POSERR /* Aktuelle Einstellung Schleppabstand lesen */ PRINT GET I_FUNCTION_9_4 /* Eingang für Abbruch lesen */
Programmbeispiel	GETP_01.M

☐ GETVLT

Kurzinfo	Liest einen VLT-Parameter.
Syntax	erg = GETVLT par
Parameter	par = Parameternummer
Rückgabewert	erg = Parameterwert
Beschreibung	GETVLT liest einen VLT-Parameter und liefert den entsprechenden Wert zurück. Mit GETVLT haben Sie somit Zugriff auf Betriebsdaten (z.B. Motorstrom 1-24) oder auf Konfigurationen (z.B. max. Sollwert Par. 3-03) des FC 300. Da ausschließlich Ganzzahlenwerte übertragen werden, muss bei der Auswertung des Rückgabewertes der Umwandlungsindex beachtet werden. So ist ein LCP Wert von 50,0 Hz (Par. 16-13 Umwandlungsindex = –1) gleichbedeutend mit einem Rückgabewert von 500. Die Liste der FC 300 Parameter mit dem zugehörigen Umwandlungsindex finden Sie im FC 300 Produkthandbuch.  **ACHTUNG!:** Benutzen Sie GETVLTSUB um Parameter mit Indexnummern zu lesen, z.B. den FC 300 Parameter 5-40.
Befehlsgruppe	PAR
Querverweise	SETVLT
Syntax-Beispiel	PRINT GETVLT 4-13 /* Lese Par. 4-13 Motordrehzahl-Obergrenze */

□ GETVLTSUB

Kurzinfo	Liest einen VLT Parameter mit Indexnummer.
Syntax	erg = GETVLTSUB par indxnr
Parameter	par = Parameternummer indxnr = Indexnummer
Rückgabewert	erg = Parameterwert
Beschreibung	GETVLTSUB liest einen VLT Parameter inklusive der Indexnummer, z.B. den FC 300 Parameter 5-40 und gibt den entsprechenden Wert zurück. Da ausschließlich Ganzzahlenwerte übertragen werden, muss bei der Auswertung des Rückgabewertes der Umwandlungsindex beachtet werden. So ist ein LCP Wert von 50,0 Hz (Parameter 16-13 Umwandlungsindex = –1) gleichbedeutend mit einem Rückgabewert von 500. Die Liste der FC 300 Parameter mit dem zugehörigen Umwandlungsindex finden Sie im FC 300 Produkthandbuch.
Befehlsgruppe	PAR
Querverweise	SETVLTSUB
Syntax-Beispiel	PRINT GETVLTSUB 540 0 // Index 01 des Parameters 5-40 "Relaisfunktion" lesen

□ GOSUB

Kurzinfo	Aufruf eines Unterprogramms.
Syntax	GOSUB name
Parameter	name = Name des Unterprogramms
Beschreibung	Der GOSUB Befehl ruft ein Unterprogramm auf und der zugehörige Programmbereich wird abgearbeitet. Nach dem letzten Unterprogrammbefehl (RETURN) wird im Hauptprogramm mit dem auf die GOSUB Anweisung folgenden Befehl fortgefahren. **ACHTUNG!:** Unterprogramme müssen am Anfang oder Ende des Programms innerhalb des SUBMAINPROG Bereichs definiert sein.
Befehlsgruppe	CON
Querverweise	SUBMAINPROG .. ENDPROG, SUBPROG .. RETURN, ON ERROR GOSUB .., ON INT n GOSUB
Syntax-Beispiel	GOSUB testup /* Aufruf des Unterprogramms testup */ Befehlszeile 1 Befehlszeile n SUBMAINPROG /* Unterprogramm testup muss definiert sein */ SUBPROG testup Befehlszeile 1 Befehlszeile n RETURN ENDPROG
Programmbeispiel	GOSUB_01.M, AXEND_01.M, INCL_01.M, STAT_01.M

☐ GOTO

Kurzinfo	Sprung zu einem Programmlabel.
Syntax	GOTO label
Parameter	label = Kennung der Programmzielposition
Beschreibung	Mit dem GOTO Befehl wird unbedingt zu der angegebenen Programmposition gesprungen und die Abarbeitung des Programms an dieser Position fortgesetzt. Die Programmposition, zu der gesprungen werden soll, ist durch ein Label gekennzeichnet. Ein Label kann aus einem oder mehreren Zeichen bestehen und darf nicht mit einem Variablennamen oder einem Befehlswort identisch sein. Ein Label muss zudem eindeutig sein, es darf nicht mehrfach an unterschiedlichen Programmpositionen verwendet werden. Mit dem GOTO Befehl ist es zum Beispiel möglich, eine Endlosschleife zu programmieren.  **ACHTUNG!:** Das Label an der Programmzielposition muss mit einem Doppelpunkt (:) versehen sein.
Befehlsgruppe	CON
Querverweise	LOOP
Syntax-Beispiel	endlos: /* Label zu dem gesprungen wird */ Befehlszeile 1 Befehlszeile n GOTO endlos /* Sprungbefehl zu Label endlos */
Programmbeispiel	GOTO_01.M, EXIT_01.M, IF_01.M

☐ HOME

Kurzinfo	Maschinennullpunkt (Referenzschalter) anfahren und als Realnullpunkt setzen.
Syntax	HOME
Beschreibung	Der HOME Befehl fährt den Antrieb zum Referenzschalter, der am Maschinennullpunkt oder an der Sollposition angebracht sein muss. Die Geschwindigkeit und Beschleunigung/Verzögerung für die Homefahrt wird in den Parametern 33-03 *Homefahrt-Geschwindigkeit* und 33-02 *Homefahrt-Rampe* festgelegt. Um eine exakte Positionierung zu erreichen, sollte die *Homefahrt-Geschwindigkeit* in Par. 33-03 nicht höher sein als 10 % der Maximaldrehzahl. Das Vorzeichen in Par. 33-03 bestimmt, in welcher Richtung nach dem Referenzschalter gesucht wird. Wenn die HOME-Position erreicht ist, wird diese als Nullpunkt definiert. Der Referenzschalter kann in vier verschiedenen Arten anfahren werden. Welche Art Homefahrt durchgeführt wird, wird in Par. 33-04 *Homefahrt-Verhalten* festgelegt: 0 = Fahren bis zum Endschalter, Reversieren und den Referenzschalter verlassen und beim nächsten Indeximpuls (Drehgeber Nullimpulse oder externes Markersignal) halten. 1 = Wie 0, aber ohne Suchen des Indeximpulses. 2 = Wie 0, aber ohne Reversieren, sondern in gleicher Richtung weiter aus dem Schalter heraus. 3 = Wie 2, aber ohne Suchen des Indeximpulses. Wird die Homefahrt durch einen Interrupt abgebrochen, wird HOME nicht automatisch weitergeführt wenn die Interrupt-Routine wieder verlassen wird. Stattdessen wird mit dem nächsten Befehl fortgefahren. Dies dient dazu, dass nach einem Störfall HOME auch abgebrochen werden kann.  **ACHTUNG!:** Die Anlage muss mit einem Referenzschalter sowie nach Möglichkeit mit einem Drehgeber mit Indexpuls ausgestattet sein.  **ACHTUNG!:** Der HOME Befehl wird auch bei NOWAIT ON zu Ende ausgeführt, bevor mit der weiteren Abarbeitung des Programms begonnen wird. Bitte beachten Sie, dass ON PERIOD xx GOSUB xx während der Homefahrt deaktiviert sein muss. Zum Beispiel ON PERIOD n GOSUB x und dann Reset, nachdem die Homefahrt beendet ist.  **ACHTUNG!:** Der Befehl HOME kann bei Einsatz von Absolutgebern (siehe Par. 32-00 *Inkrementalgeber Signaltyp*) nicht verwendet werden.
Befehlsgruppe	INI
Querverweise	INDEX, NOWAIT Parameter: 33-03 *Homefahrt-Geschwindigkeit*, 33-02 *Homefahrt-Rampe*, 33-00 *Homefahrt erzwingen?*
Syntax-Beispiel	HOME /* Referenzschalter und Index anfahren */
Programmbeispiel	HOME_01.M

☐ IF ..THEN .., ELSEIF .. THEN .. ELSE .. ENDIF

Kurzinfo	Bedingte ein- oder mehrfache Programmverzweigung; (wenn Bedingung erfüllt, dann führe aus …, sonst …)
Syntax	IF Bedingung THEN Befehl ELSEIF Bedingung THEN Befehl ELSE Befehl ENDIF
Parameter	Bedingung = Verzweigungskriterium Befehl = ein oder mehrere Programmbefehle
Beschreibung	Mit der IF..ENDIF Konstruktion können bedingte Programmverzweigungen realisiert werden. Ist die hinter IF bzw. ELSEIF stehende Bedingung erfüllt, werden die Befehle bis zur nächsten ELSEIF, ELSE oder ENDIF Anweisung ausgeführt und dann mit den nach der ENDIF Anweisung stehenden Befehlen das Programm fortgesetzt. Ist die Bedingung nicht erfüllt, werden die nachfolgenden ELSEIF Verzweigungen überprüft und es wird, sofern die Bedingung erfüllt ist, der entsprechende Programmteil ausgeführt und das Programm nach ENDIF fortgesetzt. Die Verzweigungsbedingung, die nach einer IF oder ELSEIF Anweisung überprüft wird, kann sich aus einer oder mehreren Vergleichsoperationen zusammensetzen. Innerhalb der IF..ENDIF Konstruktion können beliebig viele ELSEIF Verzweigungen auftreten, es darf jedoch nur eine ELSE Anweisung vorhanden sein. Hinter der ELSE Anweisung steht der Programmteil, der abgearbeitet wird, sofern keine der Bedingungen erfüllt wurde. Die Anweisungen ELSEIF und ELSE können, müssen aber nicht innerhalb einer IF … ENDIF Konstruktion enthalten sein.
	**ACHTUNG!:** Nachdem eine Bedingung erfüllt wurde, wird der zugehörige Programmteil ausgeführt und das Programm nach der ENDIF Anweisung fortgesetzt. Weitere Bedingungen werden nicht mehr überprüft.
Befehlsgruppe	CON
Querverweise	REPEA T .. UNTIL, WHILE **..** ENDWHILE

Syntax-Beispiel	/*** Einfachverzweigung ***/
	IF (a == 1) THEN /* Variable a = 1, dann */
	Befehlszeile 1
	Befehlszeile n
	ENDIF
	/*** Mehrfachverzweigung ***/
	IF (a == 1 AND b != 1) THEN
	Befehlszeilen
	ELSEIF (a == 2 AND b != 1) THEN
	Befehlszeilen
	ELSEIF (a == 3) THEN
	Befehlszeilen
	ELSE
	Befehlszeilen
	ENDIF

Programmbeispiel	IF_01.M, ERROR_01.M, EXIT_01.M, HOME_01.M, IN_01.M, …

☐ IN

Kurzinfo	Zustand eines digitalen Eingangs abfragen.
Syntax	erg = IN n
Parameter	n = Nummer des Eingangs 1 – 10 oder 1 – 12 (Optionale Eingänge) 18, 19, 27, 29, 32, 33
Rückgabewert	erg = Zustand des Eingangs 0 = Low-Pegel oder undefiniert 1 = High-Pegel
Beschreibung	Mit dem IN Befehl können Sie den Zustand eines digitalen Eingangs abfragen. Es wird abhängig vom anliegenden Signalpegel eine 0 oder 1 zurückgeliefert. Der Modus Eingang 11,12 wird in Par. 33-60 IOMODE ausgewählt. Die Definition des High- und Low-Pegels sowie die Eingangsbeschaltung sind in den Produkthandbüchern MCO 305 und FC 300 beschrieben. Die Eingänge 5 und 6 werden auch als Marker-Eingänge für die Master- und Slave-Drehgeber benutzt.
Befehlsgruppe	I/O
Querverweise	INB, OUT, OUTB Parameter: 33-60 *Klemme X59/1 und X59/2 Modus*, IOMODE, 33-50...59,61,62 *Klemme X57/n Digitale Eingänge*, I_FUNCTION_n

Syntax-Beispiel	in4 = IN 4 /* Zustand Eingang 4 in Variable ein4 speichern */
	IF (IN 2) THEN /* Bei High-Pegel an Klemme 2, Ausgang 1 setzen */
	OUT 1 1
	ELSE
	OUT 1 0
	ENDIF

Programmbeispiel	IN_01.M

☐ INAD

Kurzinfo	Analogen Eingang lesen.
Syntax	erg = INAD n
Parameter	n = Nummer des Analogeingangs: 53,54
Rückgabewert	erg = Analogwert Klemme 53/54: -1000 – 1000 = -10 V – 10 V Klemme 53/54: 0 – 10 V erg = 0 – 100
Beschreibung	Der INAD Befehl liest den Analogwert des entsprechenden Eingangs.
Befehlsgruppe	I/O
Syntax-Beispiel	an1 = INAD 53 PRINT "Analogeingang 53 " ,an1

□ INB

Kurzinfo	Zustand der digitalen Eingänge byteweise abfragen.
Syntax	erg = INB n
Parameter	n = Eingangsbyte: 0 = Eingang 1 (LSB) - 8 (MSB) 1 = Eingang 33 (LSB) - 18 (MSB) 2 = Eingang 9 - 10 (12)
Rückgabewert	erg = Wert des Eingangsbytes (0 - 255) Das niederwertigste Bit entspricht dabei dem Zustand des Eingangs 1/33.
Beschreibung	Mit dem INB Befehl kann der Zustand der digitalen Eingänge byteweise abgefragt werden. Der zurückgelieferte Wert spiegelt den Zustand der einzelnen Eingänge wieder. Die Definition des High- und Low-Pegels sowie die Eingangsbeschaltung ist im FC 300 Produkthandbuch beschrieben.
Befehlsgruppe	I/O
Querverweise	IN, OUT, OUTB
Syntax-Beispiel	in = INB 0 /* Zustand der ersten 8 Eingänge speichern */
Beispiel	IN1 = low, IN2 = high, IN3 = high, alle anderen Eingänge low erg = 2^1 + 2^2 = 6
Programmbeispiel	INB_01.M, INB_02.M, OUTB_01.M

□ INDEX

Kurzinfo	Indexposition des Drehgebers anfahren.
Syntax	INDEX
Beschreibung	Der INDEX Befehl startet eine Fahrt zur Indexposition des Drehgebers. Die Indexsuche erfolgt mit der *Homefahrt-Geschwindigkeit*, die in Par. 33-03 festgelegt ist. Das Vorzeichen der *Homefahrt-Geschwindigkeit* bestimmt in welcher Drehrichtung nach dem Indexsignal gesucht wird.  **ACHTUNG!:** Der verwendete Drehgeber muss einen Indexkanal haben.  **ACHTUNG!:** Es können nur Drehgeber mit low-aktivem Indexpuls verwendet werden. Wird innerhalb einer kompletten Umdrehung kein Indexpuls gefunden, erfolgt eine Fehlermeldung. Der INDEX Befehl wird auch bei NOWAIT ON zu Ende ausgeführt, bevor mit der weiteren Abarbeitung des Programms begonnen wird.  **ACHTUNG!:** Der Befehl INDEX kann bei Einsatz von Absolutgebern (siehe Par. 32-00 *Inkrementalgeber Signaltyp*) nicht verwendet werden.
Befehlsgruppe	INI
Querverweise	HOME, POSA, DEF ORIGIN, NOWAIT
Syntax-Beispiel	INDEX /* Index anfahren */
Programmbeispiel	INDEX_01.M

☐ INKEY

Kurzinfo	Einlesen eines Zeichens der Tastatur.
Syntax	INKEY (p)
Parameter	p maximale Wartezeit, definiert in **...** p = 0 es wird gewartet bis Zeichen kommt p > 0 es wird maximal p Millisekunden gewartet p < 0 es wird nicht auf Zeichen gewartet (ein negativer Parameter muss in Klammern angegeben werden)
Rückgabewert	ASCII-Code des empfangenen Zeichens bzw. –1 falls kein Zeichen vorhanden ist. Folgende Tasten-Codes werden zurückgesendet, solange die Taste gedrückt wird. Werden mehr als eine Taste gleichzeitig gedrückt, wird die entsprechende Summe der Werte zurückgesendet:

Taste:	Wert:
[Main Menu]	1
[Quick Menu]	2
[Alarm Log]	4
[Status]	8
[OK]	16
[Cancel]	32
[Info]	64
[Back]	128
[→]-Taste / rechts	256
[↑]-Taste / nach oben	512
[↓]-Taste / nach unten	1024
[←]-Taste / links	2048
[Auto on]	4096
[Reset]	8192
[Hand on]	16384
[Off]	32768

Kombinationen senden folgende Werte:

[OK] und [Cancel]	48
[Auto on] und [↑]-Taste	4608

ACHTUNG!:
Die Tasten behalten ihre FC 300-Funktionen, wenn sie nicht in Parameter 0-4* deaktiviert werden.

ACHTUNG!:
NLCP (LCP 101 Numerical Local Control Panel) ist derzeit nicht enthalten.

Beschreibung	Mit dem INKEY Befehl kann ein Tastensignal vom FC 300 LCP-Tastenfeld eingelesen werden. Der mit INKEY übergebene Parameter bestimmt dabei, ob auf ein Tastensignal ohne Bedingung, eine gewisse Zeitspanne oder gar nicht gewartet wird. Pro erfolgreichen INKEY Befehl wird jeweils ein Tastensignal eingelesen. Für die Eingabe von Zeichenketten muss der INKEY Befehl (p<>0) in einer Schleife so oft wiederholt werden, bis keine weiteren Tastensignale mehr vorliegen.
Befehlsgruppe	I/O
Querverweise	PRINT
Syntax-Beispiel	input = INKEY 0 /* Warten bis Tastensignal gelesen wird */ character = INKEY 5000 /* max. 5 Sek. auf Eingabe warten */ character = INKEY (-1) /* nicht auf Eingabe warten */
Programmbeispiel	INKEY_01.M, EXIT_01.M, WHILE_01.M

☐ IPOS

Kurzinfo	Letzte Index- bzw. Markerposition des Slaves abfragen.
Syntax	erg = IPOS
Rückgabewert	erg = letzte Slave-Position (Index oder Marker) absolut zum aktuellen Nullpunkt Die Positionsangabe wird in Benutzereinheiten [BE] zurückgeliefert und entspricht in der Standardeinstellung der Anzahl Quadcounts. (Parameter 32-12 *Benutzerfaktor Zähler* und 32-11 *Benutzerfaktor Nenner* = 1)
Beschreibung	Der Befehl IPOS liefert die letzte Index- bzw. Markerposition des Slaves absolut zum aktuellen Nullpunkt zurück.  **ACHTUNG!:** Wenn ein mit SET ORIGIN gesetzter und aktiver Temporärnullpunkt existiert, bezieht sich der Positionswert auf diesen Nullpunkt. Die Konfiguration von IPOS, d.h. ob die Index- oder Markerposition des Slave (= geregelter Antrieb) zurückgeliefert wird, erfolgt über Par. 33-20 *Markertyp Slave*.  **ACHTUNG!:** Das Triggersignal für die Markerposition muss dabei zwingend an den Eingang 6 angeschlossen werden. Der Positionswert in IPOS ist auf ±1 qc genau. Im Gegensatz zu der Positionsinformation in APOS, welche nur im Reglerzyklus von typisch 1 ms aktualisiert wird, wird der aktuelle Positionswert hardwaremäßig beim Auftreten des konfigurierten Signals (in einem internen Prozessorregister) in Echtzeit zwischengespeichert und dann in die Systemvariable IPOS kopiert. Falls gleichzeitig zur Markerposition ein Interrupt ausgelöst wird (ON INT 6 GOSUB ...) und in diesem Interrupt mit IPOS gearbeitet wird, sollte im Interrupt-Unterprogramm eine Verzögerung von 2 Millisekunden (DELAY 2) vor dem Lesen von IPOS verwendet werden. So kann sichergestellt werden, dass der gelatchte Positionswert bereits vollständig in die Systemvariable IPOS kopiert ist und nicht noch auf einen veralteten Wert zurückgegriffen wird. Siehe Beispiel.  **ACHTUNG!:** Der Befehl IPOS kann nicht verwendet werden: – Bei Einsatz von Absolutgebern (siehe Par. 32-00 *Inkrementalgeber Signaltyp*). – Wenn der Parameter 32-50 auf [3] - Motor Steuerung steht.
Befehlsgruppe	I/O
Querverweise	CPOS, DEF ORIGIN, SET ORIGIN, POSA, POSR, MIPOS, ON INT; Parameter: 32-12 *und* 32-11 *Benutzerfaktor Zähler und Nenner*, 33-20 *Markertyp Slave*
Syntax-Beispiel	PRINT IPOS /* letzte Indexposition am PC ausgeben */

Beispiel

```
ON INT 6 GOSUB slave_int  // Definition Interrupt-Handler
SET SYNCMTYPS 2    // Definition von IPOS-Latching auf positive Flanke an
Eingang 6
CVEL 10            // Bewegung starten
CSTART x(1)        // Endlos-Schleife
mainloop:                  // …
GOTO mainloop
SUBMAINPROG
SUBPROG slave_int
   int_pos = APOS
   // APOS zwischenspeichern um zu testen, wie genau dies wäre ...
   DELAY 2         // 2 ms warten, damit IPOS sicher aktualisiert ist
   triggered_pos = IPOS    // IPOS für spätere Bearbeitung etc. zwischenspeichern
      // …
      // …
   PRINT "Interrupt Position: ",int_pos
   PRINT "Triggered Position: ",triggered_pos
   RETURN
ENDPROG
```

□ LINKGPAR

Kurzinfo	Globalen Parameter oder Parametergruppen mit LCP-Display verknüpfen.
Syntax	LINKGPAR parnr "text" min max option
Parameter	parnr = LCP Parameternummer (Gruppe 19-00 bis 19-99) text = beschreibender Text für das Display; nur ASCII Text (8-Bit) wird unterstützt. min = minimaler Wert, den der Parameter annehmen darf max = maximaler Wert, den der Parameter annehmen darf option = Parametertyp 0 = offline, d.h. Änderungen werden erst durch die Bestätigung mit [OK] aktiv. 1 = online, d.h. Änderungen über das LCP-Display sind sofort aktiv.
Beschreibung	Mit LINKGPAR können Sie freie interne Anwendungsparameter mit dem LCP verknüpfen. Danach können Sie über das LCP den Parameter verändern oder den gesetzten Wert auslesen. Wird ein verknüpfter Parameter mit einem SET Befehl verändert, wird er automatisch auch an das LCP übergeben; er wirkt jedoch nur temporär, weil die Werkseinstellungen nicht geändert werden. Ändert der Anwender einen verknüpften Parameter am LCP, wird der neue Wert ausgeführt. Aber erst, wenn dieser Wert mit [OK] bestätigt wird, wird er permanent als Benutzerparameter im EEPROM gespeichert. Der Befehl LINKGPAR prüft, ob der Wert des Anwendungsparameters innerhalb des vorgegebenen Bereiches liegt. Falls nicht, wird das entsprechende Limit verwendet und dieser Wert gespeichert. Auf diese Weise wird sichergestellt, dass eine Anzeige erscheint.
Befehlsgruppe	PAR
Querverweise	SET, GET, Anwendungsparameter, Parameter-Referenz
Syntax-Beispiel	LINKGPAR 1901 "name" 0 100000 0 /* Par. 19-01 mit LCP-Display verknüpfen */

□ LINKSYSVAR

Kurzinfo	Systemvariable mit LCP-Display verknüpfen.
Syntax	LINKSYSVAR indx parnr "text"
Parameter	indx = Index der Systemvariable SYSVAR parnr = LCP-Parameternummer 19-00 bis 19-99 text = beschreibender Text für Display
Beschreibung	Der Befehl LINKSYSVAR verknüpft die Systemvariable SYSVAR[indx] mit dem FC 300 Parameter (19-00 to 19-99) und dem Anzeigetext "text". Auf diese Weise können Sie interne Werte ohne Umweg über LINKGPAR auf dem Display darstellen. Wenn Sie zum Beispiel mit #DEBUG NOSTOP kompilieren, verknüpfen Sie die interne Zeilennummer mit dem FC 300 Parameter 19-90. Dann können Sie die Programmausführung ganz gezielt beobachten.  **ACHTUNG!:** Alle 40 ms wird der Parameter aktualisiert. Wenn also auf diese Weise fünf Parameter verknüpft werden, dauert es mindestens 200 ms, bis derselbe Parameter wieder erneuert wird.
Befehlsgruppe	PAR
Querverweise	LINKGPAR, SYSVAR, Anwendungsparameter, Parameter-Referenz
Syntax-Beispiel	LINKSYSVAR 33 19-90 "interne Zeilennummer" LINKSYSVAR 30 19-91 "Motorspannung"

□ LOOP

Kurzinfo	Definierte Schleifenwiederholung.
Syntax	LOOP n label
Parameter	n = Anzahl der Schleifenwiederholungen label = Kennung der Programmzielposition
Beschreibung	Mit dem LOOP Befehl kann die ein- oder mehrmalige Wiederholung eines bestimmten Programmbereichs realisiert werden. Die Anzahl der Schleifenwiederholungen kann dabei als absoluter Wert oder auch in Form einer Variablen angegeben werden. Die Programmposition, zu der gesprungen werden soll, ist durch ein Label gekennzeichnet. Ein Label kann aus einem oder mehreren Zeichen bestehen und darf nicht mit einem Variablennamen oder einem Befehlswort identisch sein. Ein Label muss zudem eindeutig sein, das heißt das gleiche Label darf nicht mehrfach für unterschiedliche Programmpositionen verwendet werden.  Das Label an der Programmzielposition muss mit einem Doppelpunkt (:) versehen sein. Da der interne Schleifenzähler erst am Schleifenende überprüft und danach verringert wird, werden die Befehle innerhalb der Schleife insgesamt einmal mehr als in dem entsprechenden Übergabewert angegeben ausgeführt.
Befehlsgruppe	CON
Querverweise	GOTO, WHILE .. ENDWHILE, REPEAT .. UNTIL
Syntax-Beispiel	schleife: /* Label zu dem gesprungen wird */ Befehlszeile 1 Befehlszeile n LOOP 9 schleife /* Schleifeninhalt 10-mal wiederholen */
Programmbeispiel	LOOP_01.M, APOS_01.M, IN_01.M, MOTOR_01.M, NOWAI_01.M

☐ MAPOS

Kurzinfo	Aktuelle Istposition des Masters abfragen.
Syntax	erg = MAPOS
Rückgabewert	erg = Master-Position absolut zum aktuellen Nullpunkt in qc
Beschreibung	Mit MAPOS können Sie die aktuelle Master-Position (absolut zum aktuellen Nullpunkt) abfragen.
Befehlsgruppe	I/O
Querverweise	CPOS, DEF ORIGIN, SET ORIGIN, POSA, POSR, Parameter: 32-12 *Benutzerfaktor Zähler*, 32-11 *Benutzerfaktor Nenner*
Syntax-Beispiel	PRINT MAPOS /* aktuelle Master-Position abfragen und ausgeben */

☐ MAVEL

Kurzinfo	Aktuelle Geschwindigkeit des Masters abfragen.
Syntax	erg = MAVEL
Rückgabewert	erg = aktuelle Geschwindigkeit des Master-Antriebs in qc/s; Wert mit Vorzeichen
Beschreibung	Diese Funktion liefert die aktuelle Geschwindigkeit des Master-Antriebes in qc/s zurück, wobei sich qc auf den Master-Drehgeber bezieht. Die Genauigkeit der Werte hängt von der Messdauer (Mittelung) ab. Diese ist standardgemäß auf 20 ms eingestellt, kann aber vom Anwender mit dem _GETVEL Befehl verändert werden. Es genügt den Befehl einmal aufzurufen, um von da an mit einer anderen Messzeit zu arbeiten. So stellt der Befehl var = _GETVEL 100 die Messdauer auf 100 ms ein, so dass man bei MAVEL eine wesentlich bessere Auflösung der Geschwindigkeit erhält, schnelle Änderungen dagegen erst mit einer Verzögerung von maximal 100 ms.
Befehlsgruppe	I/O
Querverweis	AVEL
Syntax-Beispiel	PRINT MAVEL /* aktuelle Master-Geschwindigkeit am PC ausgeben */

☐ MIPOS

Kurzinfo	Letzte Index- bzw. Markerposition des Masters abfragen.
Syntax	erg = MIPOS
Rückgabewert	erg = letzte Index- bzw. Markerposition des Masters absolut zum aktuellen Nullpunkt in qc
Beschreibung	Diese Funktion liefert die letzte Index- bzw. Markerposition des Masters absolut zum aktuellen Nullpunkt in qc zurück. Die Konfiguration von MIPOS, d.h. ob die Index- oder Markerposition des Master-Drehgebers (= geregelter Antrieb) zurückgeliefert wird, erfolgt über den Parameter 33-19 *Markertyp Master*.  **ACHTUNG!:** Das Triggersignal für die Markerposition muss dabei zwingend an den Eingang 5 angeschlossen werden. Der Positionswert in MIPOS ist auf ±1 qc genau. Im Gegensatz zu der Positionsinformation in MAPOS, welche nur im Reglerzyklus von typisch 1 ms aktualisiert wird, wird der aktuelle Positionswert hardwaremäßig beim Auftreten des konfigurierten Signals (in einem internen Prozessorregister) in Echtzeit zwischengespeichert und dann in die Systemvariable MIPOS kopiert. Falls gleichzeitig zur Markerposition ein Interrupt ausgelöst wird (ON INT 5 GOSUB ...) und in diesem Interrupt mit MIPOS gearbeitet wird, sollte im Interrupt-Unterprogramm eine Verzögerung von 2 Millisekunden (DELAY 2) vor dem Lesen von MIPOS verwendet werden. So kann sichergestellt werden, dass der gelatchte Positionswert bereits vollständig in die Systemvariable MIPOS kopiert ist und nicht noch auf einen veralteten Wert zurückgegriffen wird.  **ACHTUNG!:** Der Befehl MIPOS kann nicht verwendet werden: – Bei Einsatz von Absolutgebern (siehe Par. 32-30 *Inkrementalgeber Signaltyp*). – Wenn der Parameter 32-50 auf [3] – Motor Steuerung steht.
Befehlsgruppe	I/O
Querverweise	CPOS, DEF ORIGIN, SET ORIGIN, POSA, POSR, ON INT Parameter: 32-12 *Benutzerfaktor Zähler*, 32-11 *Benutzerfaktor Nenner*, 33-19 *Markertyp Master*
Syntax-Beispiel	PRINT MIPOS /* letzte Indexposition des Masters am PC ausgeben */

Beispiel	// Definition Interrupt-Handler
	ON INT 5 GOSUB master_int
	// Definition IPOS-Latching auf positive Flanke an Eingang 5
	SET SYNCMTYPM 2
	CVEL 10 // Bewegung starten
	CSTART // Endlos-Schleife
	mainloop: // ...
	GOTO mainloop
	SUBMAINPROG

```
SUBPROG master_int
    int_mpos = MAPOS
    // MAPOS zwischenspeichern um zu testen, wie genau dies wäre ...
    DELAY 2                 // 2 ms warten, damit MIPOS sicher aktualisiert ist
    triggered_mpos = MIPOS
        // IPOS für spätere Bearbeitung etc. zwischenspeichern.
        // ....
        // ...
    PRINT "Interrupt Master-Position: ",int_mpos
    PRINT "Master-Position erreicht: ",triggered_mpos
    RETURN
ENDPROG
```

☐ MOTOR OFF

Kurzinfo	Motorregelung ausschalten
Syntax	MOTOR OFF
Beschreibung	Die Motorregelung kann mit dem MOTOR OFF Befehl ausgeschaltet werden. Nach einem MOTOR OFF kann, sofern keine Motorbremse vorhanden ist, die Antriebsachse frei bewegt werden. Die aktuelle Position wird weiterhin überwacht, das heißt, auch nach einem MOTOR OFF kann die Istposition (APOS) abgefragt werden.
	ACHTUNG!: Zum erneuten Starten eines Bewegungsvorganges nach MOTOR OFF muss der Befehl MOTOR ON verwendet werden. Nur der Befehl ERRCLR aktiviert MOTOR ON automatisch.
Befehlsgruppe	INI
Querverweis	MOTOR ON
Syntax-Beispiel	MOTOR OFF /* Lageregelung der Achse abschalten */
Programmbeispiel	MOTOR_01.M, POS_01.M

☐ MOTOR ON

Kurzinfo	Motorregelung einschalten
Syntax	MOTOR ON
Beschreibung	Der MOTOR ON Befehl schaltet die Motorregelung nach einem vorausgegangenen MOTOR OFF wieder ein. Beim Ausführen des MOTOR ON wird die Sollposition als Istposition gesetzt, das heißt der Motor verharrt lagegeregelt auf der Istposition. Dabei wird der Schleppfehler automatisch bei der Ausführung von MOTOR ON zurückgesetzt.
	ACHTUNG!: Der MOTOR ON Befehl ist nicht geeignet, die nach einem Fehler abgeschaltete Lageregelung wieder zu aktivieren. Hierzu muss der ERRCLR Befehl verwendet werden.
Befehlsgruppe	INI
Querverweis	MOTOR OFF
Syntax-Beispiel	MOTOR ON /* Lageregelung der Achse einschalten */
Programmbeispiel	MOTOR_01.M, POS_01.M

□ MOTOR STOP

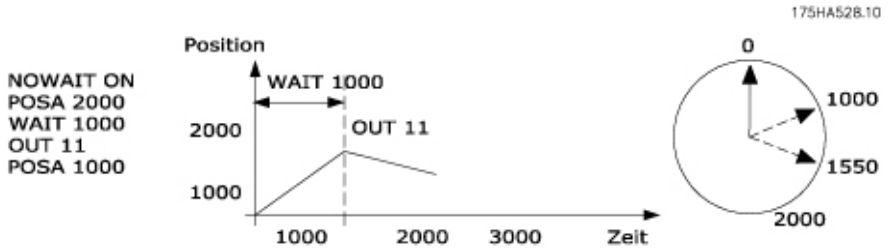
Kurzinfo	Antrieb anhalten.
Syntax	MOTOR STOP
Beschreibung	Mit dem MOTOR STOP Befehl wird ein im Positionier-, Drehzahl- oder Synchronisationsmodus fahrender Antrieb mit der zuletzt programmierten Beschleunigung abgebremst und auf der Istposition lagegeregelt. Ein mit MOTOR STOP abgebremster Antrieb kann zu einem späteren Zeitpunkt mit dem CONTINUE Befehl seinen ursprünglichen Bewegungsablauf wieder aufnehmen. (Ausnahme: CONTINUE setzt keine abgebrochenen Synchronisationsbefehle fort.)  **ACHTUNG!:** Wenn MOTOR STOP in einem Unterprogramm ausgeführt wird oder wenn NOWAIT ON gesetzt ist, werden während der Abarbeitung von MOTOR STOP schon die nächsten Programmzeilen abgearbeitet; der Bremsvorgang läuft im Hintergrund. Um den Antrieb auf Drehzahl Null abzubremsen ist daher sicherzustellen, dass während des Bremsens kein neuer Fahrbefehl gesetzt wird.
Befehlsgruppe	CON
Querverweise	POSA, POSR, CSTART, CONTINUE, CSTOP, NOWAIT
Syntax-Beispiel	MOTOR STOP /* Bewegungsvorgang der Achse unterbrechen */
Programmbeispiel	MSTOP_01.M

□ MOVESYNCORIGIN

Kurzinfo	Synchronisationsursprung relativ verschieben.
Syntax	MOVESYNCORIGIN mwert
Parameter	mwert = Relativer Offset in Bezug zum Master in qc Wertebereich: (–MLONG / Par. 33-11 SYNCFACTS) – (MLONG / Par. 33-11 SYNCFACTS)
Beschreibung	Der Befehl verschiebt den Synchronisationsursprung bezüglich des Masters. Während SET SYNCPOSOFFS den *Positionsoffset* absolut setzt, bezieht sich MOVE SYNCORIGIN immer auf den letzten und verschiebt den *Positionsoffset* relativ. Wenn Sie den *Positionsoffset* ständig verschieben müssen, können Sie so zu große Zahlen bzw. einen Überlauf verhindern.  **ACHTUNG!:** Gültig für Positionssynchronisation SYNCP und Positionssynchronisation mit Markerkorrektur SYNCM.
Befehlsgruppe	SYN
Querverweise	SET, Parameter: 33-11 *Synchronisationsfaktor Slave,* SYNCFACTS, 33-12 *Positionsoffset für Synchronisation,* SYNCPOSOFFS
Syntax-Beispiel	MOVESYNCORIGIN 1000

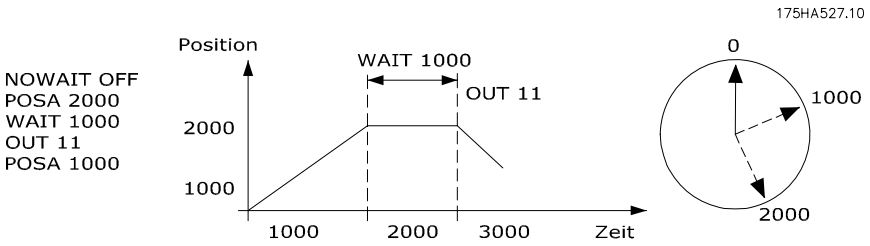
☐ NOWAIT

Kurzinfo	Nach Positionierbefehlen warten / nicht warten.
Syntax	NOWAIT s
Parameter	s = Bedingung: ON = Programmausführung fortsetzen, bis die Zielposition erreicht ist. OFF = Programmausführung anhalten bis die Zielposition erreicht ist.
Beschreibung	Ein NOWAIT Befehl definiert das Verhalten nach Positionierbefehlen mit den beiden Zuständen NOWAIT ON und NOWAIT OFF:
NOWAIT ON	Der Befehl ermöglicht sowohl die Position anzufahren als auch die darauf folgenden Anweisungen auszuführen.



Bei NOWAIT ON wird nach dem Starten eines Positionierbefehls sofort die weitere Abarbeitung der Befehle fortgesetzt und der Positioniervorgang läuft quasi im Hintergrund ab. Im Zustand NOWAIT ON ist es somit auch möglich, während eines Positioniervorgangs die Istposition abzufragen, die Geschwindigkeit oder die Zielposition zu ändern.

NOWAIT OFF — Der Befehl erlaubt die zeilenweise Ausführung des Programms. Positionierbefehle werden vollständig, das heißt bis zur Zielposition ausgeführt, bevor mit der Abarbeitung der folgenden Befehle begonnen wird.



ANMERKUNG: Die Pfeile mit gestrichelter Linie zeigen die Positionen der Bewegung.

ACHTUNG!:
Der Default-Zustand ist NOWAIT OFF. Wenn also innerhalb eines Programms keine NOWAIT ON Anweisung enthalten ist, werden Positioniervorgänge immer vollständig ausgeführt, bevor mit der Abarbeitung des nächsten Befehls begonnen wird.

Wenn im Zustand NOWAIT ON während eines noch aktiven Positioniervorgangs ein weiterer Positionierbefehl erfolgt, wird ohne Unterbrechung und ohne voriges Anfahren der ersten Zielposition sofort auf die neue Endposition verfahren.

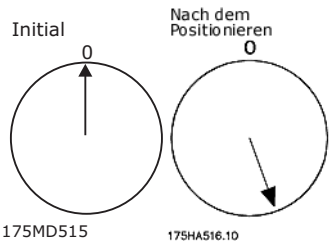
Sowohl der HOME, wie auch der INDEX Befehl werden im Zustand NOWAIT ON zu Ende abgearbeitet, bevor mit der Ausführung des nächsten Befehls begonnen wird.

Befehlsgruppe	CON
Querverweise	WAITAX, AXEND, POSA, POSR, HOME, INDEX
Syntax-Beispiel	NOWAIT ON /* nach Positionierbefehlen nicht warten */ NOWAIT OFF /* nach Positionierbefehlen warten bis Ziel erreicht */
Programmbeispiel	NOWAI_01.M, MSTOP_01.M, OUT_01.M, VEL_01.M

☐ ON APOS .. GOSUB

Kurzinfo	Unterprogramm aufrufen, wenn die Slave-Position xxx passiert wurde.
Syntax	ON sign APOS xxx GOSUB name
Parameter	sign = + = wenn die Slave-Position in positiver Richtung passiert wurde – = wenn die Slave-Position in negativer Richtung passiert wurde xxx = Slave-Position (BE) name = Name des Unterprogramms 100 xxx 0 <–––––– positive Richtung ––––––> negative Richtung
Beschreibung	Mit der Anweisung ON APOS kann man ein Unterprogramm aufrufen, wenn eine bestimmte Slave-Position (BE) in positiver bzw. negativer Richtung passiert wurde. Die Anweisung kann für Positionier- und Synchronisiersteuerungen wie auch für Kurvenscheibensteuerungen und Nockenschaltwerke nützlich sein. Zum Beispiel um bei offenen Kurven die anwachsende Slave-Position nach jedem Zyklus durch einen wiederkehrenden Bezugspunkt zu ersetzen.
	ACHTUNG!: – Ein ON APOS Interrupt kann mit dem Befehl ON DELETE .. GOSUB .. gelöscht werden. – Das aufzurufende Unterprogramm muss innerhalb eines Programmbereichs SUBMAINPROG und ENDPROG stehen. – Während der Ausführung von Unterprogrammen, die durch einen Interrupt ausgelöst wurden, ist automatisch NOWAIT ON gesetzt.
Befehlsgruppe	INT
Querverweise	SUBPROG .. RETURN, DISABLE, ENABLE …, Prioritäten bei Interrupts, ON DELETE .. GOSUB, NOWAIT

Syntax-Beispiel	ON –APOS 800 GOSUB name
	// Unterprogramm *name* aufrufen, wenn die
	// Slave-Position 800 in negativer Richtung passiert wurde

Beispiel	CSTART ON +APOS 2000 GOSUB STOP SUBMAINPROG SUBPROG STOP CSTOP RETURN ENDPROG  Gemäß dem Programm stoppt der Antrieb sobald er die Position 2000 erreicht hat.

□ ON COMBIT .. GOSUB

Kurzinfo	Unterprogramm aufrufen, wenn Bit n des Kommunikationspuffers gesetzt wird.
Syntax	ON COMBIT n GOSUB name
Parameter	n = Bit n des Kommunikationspuffers −32 <= n<=32, n!= 0 name = Name des Unterprogramms
	ON COMBIT bezieht sich auf die ersten 32 Bit des Prozessdatenspeichers.
Beschreibung	Mit der Anweisung ON COMBIT wird ein Unterprogramm aufgerufen, wenn Bit n des Kommunikationspuffers gesetzt wird.
	ACHTUNG!: – Das aufzurufende Unterprogramm muss innerhalb des durch SUBMAINPROG und ENDPROG gekennzeichneten Programmbereichs definiert sein. – Während der Ausführung von Unterprogrammen, die durch einen Interrupt ausgelöst wurden, ist automatisch NOWAIT ON gesetzt.
Priorität	Sollten mehrere Interrupts gleichzeitig auftreten, wird zuerst das dem niedrigsten Bit zugeordnete Unterprogramm abgearbeitet. Die anderen Interrupts werden anschließend abgearbeitet. Tritt während eines Interrupt-Unterprogramms der gleiche Interrupt auf (Ausnahme: Fehler-Interrupt), wird dieser ignoriert und geht verloren.
Kompatibilität	Bei COMOPTGET und COMOPTSEND wird der Offset von 2 Worten aus Kompatibilitätsgründen beibehalten.
Befehlsgruppe	INT
Querverweise	SUBPROG .. RETURN, COMOPTGET, COMOPTSEND, Prioritäten bei Interrupts, NOWAIT
Syntax-Beispiel	ON COMBIT 5 GOSUB test // Interrupt auf Bit 5 vom Feldbus setzen

□ ON DELETE .. GOSUB

Kurzinfo	Löscht einen Positions-Interrupt: ON APOS, ON MAPOS oder ON MCPOS
Syntax	ON DELETE pos GOSUB name
Parameter	pos = Wert name = Name des Unterprogramms
Beschreibung	Der Befehl kann genutzt werden, um einen ON APOS Interrupt zu löschen, der zuvor wie folgt definiert wurde: ON sign APOS xxx GOSUB name Der Parameter 'pos' kann jeden Wert annehmen, z.B. 0. Der Wert wird nicht geprüft und hat keine Bedeutung für das Löschen des Interrupts. Die Hauptbedeutung kommt dem Parameter 'name' zu, der den Namen des Unterprogramms enthalten muss, das vorher im ON APOS Befehl definiert worden ist. Daher löscht ON DELETE pos GOSUB name jeden (!) Positions-Interrupt, der zum Unterprogramm gehört und durch den Namen erkannt wird. Siehe Beispiel 1.

ACHTUNG!:
Es werden nur Positions-Interrupts gelöscht, keine anderen Interrupt-Typen.

Umleiten eines ON .. APOS .. GOSUB Befehls

Ein Positions-Interrupt kann in ein anderes Unterprogramm umgeleitet werden. Dies definiert keinen neuen Interrupt, sondern modifiziert nur das Unterprogramm, das im Fall einer Interrupt-Erkennung ausgeführt werden muss.

Die Befehlssyntax ist die gleiche wie für den ON APOS Befehl:

ON sign APOS xxx GOSUB newname

Die Parameter 'sign' und 'xxx' müssen genau die gleichen sein, wie in der ursprünglichen Definition. Die Position, die es betrifft wird durch diese zwei Parameter identifiziert. Der Parameter 'newname' muss den aktualisierten Namen des Unterprogramms enthalten, das aufgerufen werden soll, wenn der Interrupt eintritt. Siehe Beispiel 2.

ACHTUNG!:
Es können nur Positions-Interrupts umgeleitet werden, keine anderen Interrupt-Typen.

Befehlsgruppe INT

Querverweise ON APOS .., ON MAPOS .., ON MCPOS .., ON INT ..

Programmbeispiel 1

```
ON – APOS 20000     GOSUB hitinfo     // Interrupt #1
ON – APOS 10000     GOSUB hitinfo     // Interrupt #2
ON + APOS 10000     GOSUB hitinfo     // Interrupt #3
ON + APOS 0         GOSUB hitzero     // Interrupt #4
ON – APOS 0         GOSUB hitzero     // Interrupt #5
ON INT 3            GOSUB hitinfo     // Interrupt #6
…
ON DELETE 0 GOSUB hitinfo
…
ON + APOS 99999 GOSUB hitinfo     // Neuer definierter Positions-Interrupt
```

Ergebnis:
Alle Positions-Interrupts (#1, #2, #3), die zu dem Unterprogramm 'hitinfo' gehören, werden gelöscht sobald 'ON DELETE 0 GOSUB hitinfo' ausgeführt wird. Diese Interrupts zählen nicht mehr für die maximale Anzahl der verfügbaren Interrupts und können daher auch nicht mehr freigegeben oder gesperrt werden. Alle anderen nicht-Positions-Interrupts, sogar diejenigen, die zum gleichen Unterprogramm gehören (z.B. ON INT 3) sind weiterhin gültig!

Sobald die Befehlszeile 'ON + APOS 99999 GOSUB hitinfo' ausgeführt wird, definiert dies einen neuen Positions-Interrupt, der zu dem Unterprogramm verweist, das bereits früher in Gebrauch war.

Programmbeispiel 2

```
ON – APOS 10000 GOSUB hitinfo       // Interrupt #1
ON + APOS 10000 GOSUB hitinfo       // Interrupt #2
…
ON + APOS 10000 GOSUB hitposdir     // Interrupt #2 umleiten
```

Ergebnis:
Sobald die zweite Definition des 'ON + APOS 10000 ...' Befehls ausgeführt ist, wird der Interrupt #2 zu einem neu definierten Unterprogramm 'hitposdir' umgeleitet. Es ist nach wie vor der gleiche Interrupt (d.h. kein weiterer), der nun ein anderes Unterprogramm aufruft. Die „alte” Definition des Interrupts #1 'ON - APOS 10000 GOSUB hitinfo' ist weiterhin ohne jede Veränderung gültig.

☐ ON ERROR GOSUB

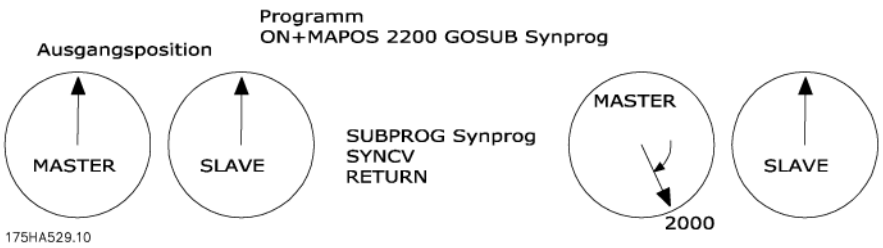
Kurzinfo	Definition eines Unterprogramms für den Fehlerfall.
Syntax	ON ERROR GOSUB name
Parameter	name = Name des Unterprogramms
Beschreibung	Mit der ON ERROR GOSUB Anweisung wird ein Unterprogramm definiert, das bei einem Fehler aufgerufen wird. Tritt nach dieser Definition zu einem beliebigen Zeitpunkt ein Fehler auf, wird das Programm nicht automatisch abgebrochen, sondern das definierte Unterprogramm aufgerufen. Innerhalb dieses Unterprogramms ist es dann möglich, gezielt auf den Fehler zu reagieren, auf Benutzereingriffe zu warten, die Fehlermeldung mit ERRCLR zu löschen oder bei nicht behebbaren Fehlern mit der EXIT Anweisung das Programm abzubrechen. Wird das Programm nicht abgebrochen, dann wird nach dem RETURN Befehl an der vor dem Aufruf des Fehlerunterprogramms bearbeiteten Programmposition fortgefahren. Mit dem CONTINUE Befehl kann man den durch den Fehler unterbrochenen Bewegungsablauf fortsetzen (Ausnahme: Synchronisationsbefehle) **ACHTUNG!:** Die ON ERROR GOSUB Anweisung sollte am Programmanfang stehen, damit sie für das ganze Programm Gültigkeit besitzt. Das aufzurufende Unterprogramm muss innerhalb des durch SUBMAINPROG und ENDPROG gekennzeichneten Programmbereichs definiert sein. Das Erkennen eines Interrupts und der Aufruf des entsprechenden Unterprogramms benötigen maximal 2 Millisekunden. **ACHTUNG!:** – Fehlerunterprogramme können nicht durch andere Interrupts unterbrochen werden. – Während der Ausführung eines Fehlerunterprogramms ist automatisch NOWAIT ON gesetzt. Wird das Fehlerunterprogramm mit einem Fehler verlassen, weil zum Beispiel kein ERRCLR durchgeführt wurde oder bereits ein neuer Fehler aufgetreten ist, erfolgt ein erneuter Aufruf. **ACHTUNG!:** Die ON ERROR GOSUB Anweisung beendet nicht HOME und INDEX Befehle. Das heißt, diese werden zu Ende ausgeführt, sobald der Fehler gelöscht ist. Um dies zu verhindern, kann man ein ON TIME 1 in die ERROR Anweisung einfügen.
Befehlsgruppe	INT
Querverweise	SUBPROG…RETURN, ERRCLR, ERRNO, CONTINUE, EXIT, Prioritäten bei Interrupts, ON TIME, NOWAIT
Syntax-Beispiel	ON ERROR GOSUB errhandle /* Definition eines Fehlerunterprogramms */ Befehlszeilen 1 … n SUBMAINPROG /* Unterprogramm 'errhandle' muss definiert sein */ SUBPROG errhandle Befehlszeilen 1 … n RETURN ENDPROG
Programmbeispiel	ERROR_01.M, IF_01.M, INDEX_01.M

☐ ON INT .. GOSUB

Kurzinfo	Definition eines Interrupt-Eingangs.
Syntax	ON INT n GOSUB name
Parameter	n = Nummer des zu überwachenden Eingangs; (Eingabebereich –8 … 8 und FC 300 Eingänge 18 … 33 und –33 … –18) Positive Eingangsnummern (1 … 8) = Reaktion auf steigende Flanke Negative Eingangsnummern (–1 … 8) = Reaktion auf fallende Flanke name = Name des Unterprogramms
Beschreibung	Mit der Anweisung ON INT GOSUB wird ein Unterprogramm definiert, das aufgerufen wird, wenn an dem überwachten Eingang eine Pegeländerung auftritt. Pro Eingang kann maximal ein Unterprogramm definiert werden. Es ist nicht möglich, für die fallende und für die steigende Flanke des gleichen Eingangs Interrupts zu definieren. Diese Definition kann jederzeit stattfinden. Tritt nach dieser Definition zu einem beliebigen Zeitpunkt der entsprechende Interrupt auf, wird das dazu gehörende Unterprogramm aufgerufen und abgearbeitet. Nach dem letzten Unterprogrammbefehl (RETURN) wird das Programm an der Position vor dem Interrupt fortgesetzt.  **ACHTUNG!:** Die Anweisung ON INT GOSUB sollte am Programmanfang stehen, damit sie für das ganze Programm Gültigkeit besitzt. Das aufzurufende Unterprogramm muss innerhalb des durch SUBMAINPROG und ENDPROG gekennzeichneten Programmbereichs definiert sein. Das Erkennen eines Interrupts und der Aufruf des entsprechenden Unterprogramms benötigen maximal 2 Millisekunden. Ein Interrupt vom Eingang FC 300 addiert im schlimmsten Fall zusätzlich 2 ms. Es ist eine minimale Signallänge von 1 ms notwendig, um eine Pegeländerung sicher zu erkennen. Informieren Sie sich bitte in den MCO 305 und FC 300 Produkthandbüchern über die Beschaltung und technischen Daten der Eingänge.  **ACHTUNG!:** – Die Anweisung ON INT GOSUB ist flanken- und nicht pegelgesteuert. – Während der Ausführung von Unterprogrammen, die durch einen Interrupt ausgelöst wurden, ist automatisch NOWAIT ON gesetzt.
Priorität	Sollten mehrere Interrupts gleichzeitig auftreten, wird zuerst das, dem kleinsten Eingang zugeordnete Unterprogramm abgearbeitet. Die andern Interrupts werden anschließend abgearbeitet. Tritt während eines Interrupt-Unterprogramms der gleiche Interrupt (Ausnahme: Fehler-Interrupt) auf, wird dieser ignoriert und geht verloren.
Befehlsgruppe	INT
Querverweise	SUBPROG..RETURN, ON ERROR .. GOSUB, WAITI, DISABLE interrupts, ENABLE interrupts, Prioritäten bei Interrupts, NOWAIT
Syntax-Beispiel	ON INT 4 GOSUB posin /* Definition von Eingang 4 (positive Flanke) */ ON INT –5 GOSUB negin /* Definition von Eingang 5 (negative Flanke) */ Befehlszeile 1 Befehlszeile n SUBMAINPROG /* Unterprogramme müssen definiert sein */ SUBPROG posin Befehlszeile 1 Befehlszeile n RETURN

	SUBPROG negin Befehlszeilen 1 … n RETURN ENDPROG
Programmbeispiel	ONINT_01.M, DELAY_01.M

□ ON MAPOS .. GOSUB

Kurzinfo	Unterprogramm aufrufen, wenn die Master-Position xxx (MU) passiert wurde.
Syntax	ON sign MAPOS xxx GOSUB name
Parameter	sign = + = wenn die Master-Position in positiver Richtung passiert wurde – = wenn die Master-Position in negativer Richtung passiert wurde xxx = Master-Position in MU name = Name des Unterprogramms 100 xxx 0 <–––––– positive Richtung ––––––> negative Richtung
Beschreibung	Mit der Anweisung ON MAPOS kann man ein Unterprogramm aufrufen, wenn eine bestimmte Master-Position [qc] in positiver bzw. negativer Richtung passiert wurde. Zum Beispiel um bei einem Linearantrieb (Slave) mit einem Verfahrbereich von 0 bis 10000 BE an einer beliebigen Position einen Ausgang zu setzen. **ACHTUNG!:** – Ein ON MAPOS Interrupt kann mit dem Befehl ON DELETE .. GOSUB .. gelöscht werden. – Das aufzurufende Unterprogramm muss innerhalb eines Programmbereichs SUBMAINPROG und ENDPROG stehen. – Während der Ausführung von Unterprogrammen, die durch einen Interrupt ausgelöst wurden, ist automatisch NOWAIT ON gesetzt.
Befehlsgruppe	INT
Querverweise	SUBPROG .. RETURN, DISABLE …, ENABLE …, Prioritäten bei Interrupts, ON DELETE .. GOSUB, NOWAIT
Syntax-Beispiel	ON +MAPOS 1200 GOSUB name // Unterprogramm name immer an Position 1200
Beispiel	 Programmgemäß startet die Geschwindigkeits-Synchronisation, nachdem der Master die Position 2200 qc in positiver Richtung erreicht hat. Dann fahren der Slave und der Master mit SYNCV.

☐ ON MCPOS .. GOSUB

Kurzinfo	Unterprogramm aufrufen, wenn die Master-Position xxx (MU) passiert wurde.
Syntax	ON sign MCPOS xxx GOSUB name
Parameter	sign = + = wenn die Master-Position in positiver Richtung passiert wurde – = wenn die Master-Position in negativer Richtung passiert wurde xxx = Master-Position in MU name = Name des Unterprogramms 100 xxx 0 <–––––– positive Richtung ––––––> negative Richtung
Beschreibung	Mit dieser für Kurvenscheibensteuerungen typischen Anweisung ON MCPOS kann man ein Unterprogramm aufrufen, wenn eine bestimmte Master-Position (MU) in positiver oder negativer Richtung passiert wurde. Auf diese Weise lassen sich nicht nur Nockenschaltwerke realisieren, sondern auch viel komplexere Aufgaben durchführen. Zum Beispiel könnte man abhängig von der Position online Parameter ändern. **ACHTUNG!:** – Vor dem Befehl ON MCPOS .. GOSUB muss immer ein DEFMCPOS oder ein SETCURVE stehen, weil sonst die Kurvenposition nicht bekannt ist – Ein ON MCPOS Interrupt kann mit dem Befehl ON DELETE… GOSUB gelöscht werden. – Das aufzurufende Unterprogramm muss innerhalb eines Programmbereichs SUBMAINPROG und ENDPROG stehen. – Während der Ausführung von Unterprogrammen, die durch einen Interrupt ausgelöst wurden, ist automatisch NOWAIT ON gesetzt.
Befehlsgruppe	INT
Querverweise	SUBPROG .. RETURN, DISABLE …, ENABLE …, Prioritäten bei Interrupts, ON DELETE .. GOSUB
Syntax-Beispiel	ON +MCPOS 1200 GOSUB parameter // Unterprogramm parameter immer an Position 1200 aufrufen
Syntax-Beispiel	Auf einem Band werden Kartons in unregelmäßigen Abständen transportiert. Durch Setzen eines Ausgangs wird der Slave immer dann gestartet, wenn die Position xxx erreicht wird. ON +MCPOS 4500 GOSUB output // Unterprogramm ausgang immer an Position 4500 aufrufen SUBMAINPROG // Unterprogramm Ausgang setzen SUBPROG output OUT 3 1 // 03 on RETURN ENDPROG

□ ON PARAM .. GOSUB

Kurzinfo	Unterprogramm aufrufen, wenn ein Parameter von außen geändert wird.
Syntax	ON PARAM n GOSUB name
Parameter	Parameternummer Name des Unterprogramms
Beschreibung	Mit der Anweisung ON PARAM kann man reagieren, wenn Parameter über das LCP-Display geändert werden und ein Unterprogramm aufrufen. Alle Parameter (32-xx, 33-xx) und alle Anwendungsparameter (19-xx) sowie allgemeine Parameter (z.B. 8-02, 5-00) können dazu benutzt werden. **ACHTUNG!:** Das aufzurufende Unterprogramm muss innerhalb eines durch SUBMAINPROG und ENDPROG gekennzeichneten Programmbereichs definiert sein. Es sind maximal 10 ON PARAM Funktionen möglich. **ACHTUNG!:** Während der Ausführung von Unterprogrammen, die durch einen Interrupt ausgelöst wurden, ist automatisch NOWAIT ON gesetzt.
Priorität	Tritt während eines Interrupt-Unterprogramms der gleiche Interrupt auf (Ausnahme: Fehler-Interrupt), wird dieser ignoriert und geht verloren.
Befehlsgruppe	INT
Querverweise	SUBPROG..RETURN, DISABLE interrupts, ENABLE interrupts, Prioritäten bei Interrupts
Syntax-Beispiel	ON PARAM 32-67 GOSUB poserr // wenn Schleppfehler geändert wird SUBMAINPROG SUBPROG poserr PRINT "Neuer Schleppfehler: ", GET POSERR RETURN

□ ON PERIOD

Kurzinfo	Ruft ein Unterprogramm in regelmäßigen Abständen auf.
Syntax	ON PERIOD n GOSUB name
Parameter	n > 20 ms = Zeit in ms, nach der das Programm immer wieder aufgerufen wird (maximal MLONG) n = 0 = Funktion abschalten name = Name des Unterprogramms
Beschreibung	Mit ON PERIOD kann man ein Unterprogramm in regelmäßigen Abständen (zeitgeführt) aufrufen. ON PERIOD wirkt wie ein Interrupt und wird alle 20 ms überprüft. **ACHTUNG!:** – Die Genauigkeit mit der die Zeit eingehalten wird, hängt vom restlichen Programm ab. Typischerweise beträgt die Genauigkeit ±1 ms. – Das aufzurufende Unterprogramm muss innerhalb des durch SUBMAINPROG und ENDPROG gekennzeichneten Programmbereichs definiert sein. – Während der Ausführung eines ON PERIOD Unterprogramms ist automatisch NOWAIT ON gesetzt.
Befehlsgruppe	INT
Querverweise	ON TIME, GOSUB, DISABLE interrupts, ENABLE interrupts, Prioritäten bei Interrupts

□ ON STATBIT .. GOSUB

Kurzinfo	Unterprogramm aufrufen, wenn Bit n des Statuswortes gesetzt wird.
Syntax	ON STATBIT n GOSUB name
Parameter	n = Bit n des Statuswortes Byte 1 Statusbyte des FC 300 Bit 1,2,4-7 ohne Bedeutung Bit 3 1 = Bewegung beendet Bit 8 1 = FC 300 abgeschaltet Byte 2 Bit 9 1 = Achse fährt Bit 10 1 = Überlauf Slave-Drehgeber Bit 11 1 = Überlauf Master-Drehgeber Bit 12 1 = FC 300 temporär abgeschaltet *) Byte 3 nwird nicht verwendet Byte 4 SYNCSTAT Bit 25 1 = SYNCREADY Bit 26 1 = SYNCFAULT Bit 27 1 = SYNCACCURACY Bit 28 1 = SYNCMMHIT Bit 29 1 = SYNCSMHIT Bit 30 1 = SYNCMMERR Bit 31 1 = SYNCSMERR name = Name des Unterprogramms *) Erläuterung: d.h. die Achse befindet sich innerhalb des Toleranzbereiches des Regelfensters Par. 32-71 REGWMAX / Par. 32-72 REGWMIN. Der FC 300 wird wieder eingeschaltet, sobald das Regelfenster verlassen wird.
Beschreibung	Mit der Anweisung ON STATBIT wird ein Unterprogramm aufgerufen, wenn Bit n des Statuswortes gesetzt ist. Diese 32 Bit des Statuswortes setzen sich aus dem Statusbyte des FC 300, dem Byte 2 des internen Status (zum Beispiel „Achse fährt") und dem Bit n von SYNCSTAT zusammen. **ACHTUNG!:** – Das aufzurufende Unterprogramm muss innerhalb des durch SUBMAINPROG und ENDPROG gekennzeichneten Programmbereichs definiert sein. – Während der Ausführung von Unterprogrammen, die durch einen Interrupt ausgelöst wurden, ist automatisch NOWAIT ON gesetzt.
Priorität	Sollten mehrere Interrupts gleichzeitig auftreten, wird zuerst das dem niedrigsten Bit zugeordnete Unterprogramm abgearbeitet. Die anderen Interrupts werden anschließend abgearbeitet. Tritt während eines Interrupt-Unterprogramms der gleiche Interrupt auf (Ausnahme: Fehler-Interrupt), wird dieser ignoriert und geht verloren.
Befehlsgruppe	INT
Querverweise	SUBPROG ..RETURN, DISABLE interrupts, ENABLE interrupts, Prioritäten der Interrupts
Syntax-Beispiel	ON STATBIT 30 GOSUB markererror /* Interrupt, wenn Fehler-Flag Master */ SUBMAINPROG SUBPROG markererror SYNCSTATCLR 32 /* Fehler-Flag SYNCMMERR löschen */ /* Wert 32 von Parameter SYNCSTATCLR, nicht Bit-Nummer! */ RETURN ENDPROG

☐ ON TIME

Kurzinfo	Einmalaufruf eines Unterprogramms.
Syntax	ON TIME n GOSUB name
Parameter	n = Zeit in ms, nach der das Unterprogramm aufgerufen wird (maximal MLONG) name = Name des Unterprogramms
Beschreibung	Nach Ablauf der gesetzten Zeit wird das entsprechende Unterprogramm einmal aufgerufen. Das Programm läuft in der Zwischenzeit normal weiter. **ACHTUNG!:** – Die Genauigkeit mit der die Zeit eingehalten wird, hängt von der eingesetzten Hardware und vom restlichen Programm ab. Typischerweise beträgt die Genauigkeit ±1 ms. – Das aufzurufende Unterprogramm muss innerhalb des durch SUBMAINPROG und ENDPROG gekennzeichneten Programmbereichs definiert sein. – Während der Ausführung eines ON TIME Unterprogramms ist automatisch NOWAIT ON gesetzt.
Befehlsgruppe	INT
Querverweise	ON PERIOD, GOSUB, DISABLE interrupts, ENABLE interrupts, Prioritäten der Interrupts
Syntax-Beispiel	OUT 1 1 /* Lampe an */ ON TIME 200 GOSUB off1 /* Lampe nach 200 ms wieder aus */ SUBMAINPROG SUBPROG off1 OUT 1 0 RETURN ENDPROG

☐ OUT

Kurzinfo	Digitale Ausgänge setzen oder zurücksetzen.
Syntax	OUT n s oder OUT X/n s
Parameter	n = Nummer des Ausgangs MCO 305: 1 – 8 (6) FC 301 outputs: 27 Relais Ausgänge: 21 (Relais 1) und 22 (Relais 2); Achtung: FC 301 < 11 kW hat nur Relais 1. MCB 105, Relais Ausgänge: X34/1 (Relais 7), X34/5 (Relais 8) und X34/10 (Relais 9). X/n = Klemmenblock / Pin Nummer s = Zustand 0 = OFF 1 = ON
Beschreibung	Der OUT Befehl kann die 8 (6) digitalen Ausgänge der MCO 305 Option, die digitalen und Relaisausgänge des FC 300 und die Relaisausgänge des MCB 105 setzen oder zurücksetzen. Der Modus für die Ausgänge 7,8 wird in Par. 33-60 IOMODE ausgewählt. Ob die Ausgänge NPN- oder PNP-schaltend benutzt werden, hängt von der Auswahl der Standardausgänge des FC 300 ab, die in Par. 5-00 gesetzt werden.  **ACHTUNG!:** Wenn eine falsche Kombination oder Pin-Nummer für X/n benutzt wird, die nicht gesetzt werden kann, wird Fehler 171 gemeldet. Aber es gibt keine Überprüfung, falls ein Eingang statt eines Ausgangs oder umgekehrt benutzt wird.  **ACHTUNG!:** Nach dem Einschalten der Anlage sind alle Ausgänge OFF. Der Schaltzustand von Ausgängen, die gemäß den I/O-Parametern vordefinierte Funktionen besitzen, wird mit dem OUT Befehl ebenfalls beeinflusst! Der aktuelle Schaltzustand bleibt auch nach Beendigung oder Abbruch eines Programms erhalten. Die Schaltlogik sowie die maximale Strombelastbarkeit sind in den Produkthandbüchern von MCO 305 und FC 300 beschrieben.
Befehlsgruppe	I/O
Querverweise	OUTB, IN, INB, Parameter: 33-60 *Klemme X59/1 und X59/2 Modus*, IOMODE, 33-63...70 *Klemme X59/n Digitaler Ausgang*, O_FUNCTION_n
Syntax-Beispiele	OUT 3 1 // MCO 305 Ausgang 3 auf 1 setzen OUT 27 1 // Ausgang 27 der FC 300 Hauptplatine auf 1 setzen OUT X59/3 1 // MCO 305 Ausgang 3 auf 1 setzen OUT X34/1 1 // erstes Relais der Relais-Option (Relais 7) auf 1 setzen
Programmbeispiel	OUT_01.M

□ OUTAN

Kurzinfo	Drehzahlsollwert setzen.
Syntax	OUTAN w
Parameter	w = Bus-Sollwert Bereich: –0X4000 – 0X4000 = –100 % – 100 %
Beschreibung	Mit dem OUTAN Befehl können Sie den Sollwert für den FC 300 Bus vorgeben. (Der Drehzahl- oder Drehmoment-Sollwert hängt davon ab, wie der FC 300 Par. 1-00 gesetzt ist.) Mit OUTAN kann man auch in „Open Loop" mit MOTOR OFF die Regelung abschalten und den FC 300 als reinen Frequenzumrichter betreiben. So können Sie APOSS benutzen um direkt gesetzte Werte auszugeben, Eingänge zu lesen usw.
	**ACHTUNG!:** Vor dem Befehl OUTAN muss MOTOR OFF ausgeführt werden. Daher ist die Schleppfehlerüberwachung nicht mehr aktiv.
Befehlsgruppe	I/O
Querverweise	MOTOR OFF, MCO 305 Produkthandbuch, FC 300 Produkthandbuch
Syntax-Beispiel	MOTOR OFF // Regelung ausschalten OUTAN 0X2000 // Drehzahlsollwert auf 50 % setzen

□ OUTB

Kurzinfo	Zustand der digitalen Ausgänge byteweise verändern.
Syntax	OUTB n w
Parameter	n = Ausgangsbyte 0 = 1 – 8 1 = 27,29 w = Wert (0 ... 255)
Beschreibung	Mit dem OUTB Befehl kann der Zustand der digitalen Ausgänge byteweise verändert werden. Der übergebene Bytewert bestimmt den Zustand der einzelnen Ausgänge. Das niederwertigste Bit des Bytewertes entspricht dabei dem Sollzustand des Ausgangs 1.
	**ACHTUNG!:** Nach dem Einschalten der Anlage sind alle Ausgänge auf OFF. Ausgänge, die gemäß den I/O-Parameter-Einstellungen vordefinierte Funktionen besitzen, werden durch den OUTB Befehl ebenfalls beeinflusst! Der aktuelle Schaltzustand bleibt auch nach Beendigung oder Abbruch eines Programms erhalten. Schaltlogik sowie maximale Strombelastbarkeit siehe MCO 305 Produkthandbuch.
Befehlsgruppe	I/O
Querverweise	OUT, IN, INB, Parameter: 33-63...70 *Klemme X59/n Digitaler Ausgang*, O_FUNCTION_n
Syntax-Beispiele	OUTB 0 10 // Ausgang 2 + 4 durchschalten, sonstige Ausgänge sperren OUTB 0 245 // Ausgang 2 und 4 sperren, alle anderen durchschalten OUTB 0 128 // nur Ausgang 8 durchschalten, andere sperren
Programmbeispiel	OUTB_01.M

□ OUTDA

Kurzinfo	Analogen FC 300 Ausgang setzen.
Syntax	OUTDA n w
Parameter	n = Ausgangsnummer (42) w = Wert (0 – 100000)
	ACHTUNG!: Parameter 6-50 muss auf „MCO gesteuert" eingestellt sein.
Beschreibung	Mit dem OUTDA Befehl kann man den analogen Ausgang der FC 300 Steuerkarte setzen. Der FC 300 hat einen analogen Ausgang, der in Parameter 6-50 konfiguriert wird. Ein Ausgang der FC 300 Steuerkarte kann nur vom Anwendungsprogramm gesetzt werden, wenn er als Optionsausgang im entsprechenden Parameter konfiguriert ist.
Befehlsgruppe	I/O
Querverweis	FC 300 Produkthandbuch, Parameter 6-50
Syntax-Beispiel	/* Voraussetzung: Parameter 6-50 ist auf "MCO gesteuert" gesetzt */ OUTDA 42 50000 /* FC 300 Ausgang auf 10 mA setzen */

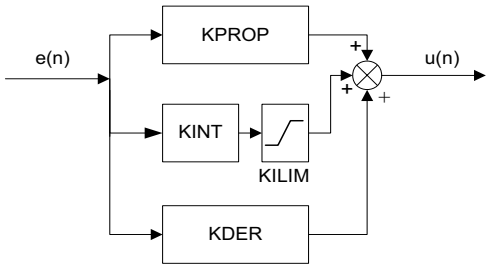
□ PCD

Kurzinfo	Pseudo-Array für den direkten Zugriff auf den Feldbus-Datenbereich.
Syntax	PCD[n]
Parameter	n = Index
Beschreibung	Ohne einen zusätzlichen Befehl COMOPTGET oder COMOPTSEND können Sie mit dem Befehl PCD direkt auf den Feldbus-Datenbereich zugreifen. Es wird wortweise (16-Bit) der Kommunikationsspeicher beschrieben oder gelesen.
	ACHTUNG!: Zusätzlich müssen die Parameter 9-15 und 9-16 mit den richtigen Werten gesetzt sein.
Befehlsgruppe	Kommunikationsoption
Querverweise	COMOPTGET, COMOPTSEND, SYSVAR

Syntax-Beispiel	Variable = PCD[1] // Wort 1
	Variable = PCD[1].2 // Bit 2 von Wort 1
	Variable = PCD[2].b1 // Byte 1 von Wort 2
	PCD[1] = Variable
	PCD[1].3 = Variable

Syntax-Beispiel	_IF (PCD[2]= = 256) THEN // Wert vergleichen _IF (PCD[3].2) THEN // ist Bit 2 von PDO 3 high?

□ PID

Kurzinfo	PID-Filter berechnen.
Syntax	u(n) = PID e(n)
Parameter	e(n) = aktuelle Abweichung (Fehler) auf die der PID-Filter angewendet werden soll
Rückgabewert	u(n) = Ergebnis der PID-Berechnung
Beschreibung	Mit dieser Funktion kann ein PID-Filter berechnet werden. Der PID-Filter arbeitet nach der Formel: u(n) = (KP * e(n) + KD *(e(n)-e(n–1)) + KI*∑e(n)) / timer wobei gilt: e(n) Fehler zum Zeitpunkt n KP *Proportionalfaktor* des PID-Reglers KD *Differentialfaktor* KI *Integralfaktor* (begrenzt durch *Integrationslimit*) timer *Abtastzeit*  Die entsprechenden Faktoren können mit folgenden Befehlen gesetzt werden: SET PID KPROP 1 /* setze KP 1 */ SET PID KDER 1 /* setze KD 1 */ SET PID KINT 0 /* setze KI 0 */ SET PID KILIM 0 /* Grenzwert für die Integralsumme = 0 */ SET PID TIMER 1 /* Abtastzeit = 1 */ wobei das folgende Syntax-Beispiel auch gleich die Default-Belegung der Faktoren zeigt.
Befehlsgruppe	I/O
Syntax-Beispiel	e = INAD 53 u = PID e PRINT "Eingang = ",e, "Ausgabe = ",u

□ POSA

Kurzinfo	Absolut zum aktuellen Nullpunkt positionieren.
Syntax	POSA p
Parameter	p = Position in Benutzereinheiten (BE) absolut zum aktuellen Nullpunkt; in der Standardeinstellung entsprechen die BE der Anzahl der Quadcounts.
Beschreibung	Mit dem POSA Befehl wird die Achse auf eine Position absolut zum aktuellen Nullpunkt bewegt. Wenn durch den Befehl POSA die *Negative* oder *Positive Software-Wegbegrenzung* (Parameter 33-41 oder 33-42) überschritten werden, wird nach dem Fehler mit dem nächsten Befehl fortgefahren. **ACHTUNG!:** Wenn ein mit SET ORIGIN gesetzter und aktiver Temporärnullpunkt existiert, wird die Positionsangabe auf diesen Nullpunkt bezogen. **ACHTUNG!:** Sollte beim Aufruf des POSA Befehls noch keine Beschleunigung und/oder Geschwindigkeit definiert sein, wird mit den Werten der Parameter 32-84 *Default-Geschwindigkeit* und 32-85 *Default-Beschleunigung* gefahren.
Befehlsgruppe	ABS

Querverweise	VEL, ACC, POSR, HOME, DEF ORIGIN, SET ORIGIN Parameter: 32-12 *Benutzerfaktor Zähler*, 32-11 *Benutzerfaktor Nenner*
Syntax-Beispiel	POSA 50000 /* Achse auf Position 50000 fahren */
Programmbeispiel	POS_01.M

□ POSA CURVEPOS

Kurzinfo	Slave auf die Kurvenposition fahren, die der Master-Position entspricht.
Syntax	POSA CURVEPOS
Beschreibung	Dieser Befehl wirkt wie ein POSA Befehl und bewegt den Slave zur entsprechenden Position der Kurve, die durch die aktuelle Master-Position vorgegeben ist.

ACHTUNG!:
Wenn ein mit SET ORIGIN gesetzter und aktiver Temporärnullpunkt existiert, wird die Positionsangabe auf diesen Nullpunkt bezogen.

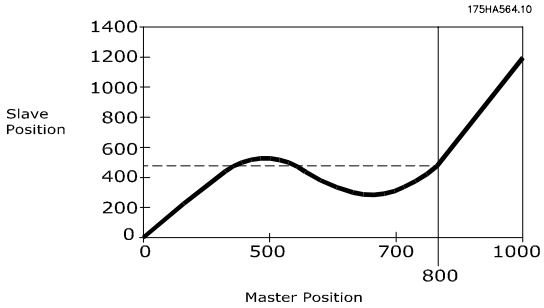
ACHTUNG!:
Sollte beim Aufruf des POSA Befehls noch keine Beschleunigung und/oder Geschwindigkeit definiert sein, wird mit den Werten der Parameter 32-84 *Default-Geschwindigkeit* und 32-85 *Default-Beschleunigung* gefahren.

Befehlsgruppe	ABS, CAM
Querverweise	CURVEPOS, SET ORIGIN
Syntax-Beispiel	POSA CURVEPOS // Slave auf die der Master-Position entsprechenden Kurvenposition fahren.

Beispiel

Fixpunkte einer Kurve:

Master	Slave
0	0
500	500
700	300
1000	1200



Angenommen, die Istposition ist 800 (vertikale Linie).

Fall 1: Istposition Master ist 800 und Istposition Slave ist 200.
POSA CURVEPOS fährt den Slave auf Position 450.

Fall 2: Istposition Master ist 800 und Istposition Slave ist 700.
POSA CURVEPOS fährt den Slave ebenfalls auf Position 450.

□ POSR

Kurzinfo	Relativ zur Istposition positionieren.
Syntax	POSR d
Parameter	d = Distanz zur Istposition in Benutzereinheiten (BE); dies entspricht in der Standardeinstellung der Anzahl Quadcounts.
Beschreibung	Der POSR Befehl bewegt die Achse auf eine Position relativ zur Istposition.

ACHTUNG!:
Wenn beim Aufruf des POSA Befehls noch keine Beschleunigung und/oder Geschwindigkeit definiert ist, wird mit Werten der Parameter 32-84 *Default-Geschwindigkeit* und 32-85 *Default-Beschleunigung* gefahren.

Befehlsgruppe	REL
Querverweise	VEL, ACC, POSA; Parameter: 32-12, 32-11 *Benutzerfaktor Zähler und Nenner*
Syntax-Beispiel	POSR 50000 /* Achse relativ um 50000 BE verfahren */
Programmbeispiel	POS_01.M

□ PRINT

Kurzinfo	Ausgabe von Informationen.
Syntax	PRINT i oder PRINT i;
Parameter	i = Information, z.B. Variable, Text, CHR (n) getrennt durch Kommata. Der Befehl CHR liefert zu einer Zahl das entsprechende ASCII-Zeichen.
Beschreibung	Mit dem PRINT Befehl können Rechenergebnisse, Variableninhalte und Textinformationen über die RS485 Kommunikationsschnittstelle am angeschlossenen PC ausgegeben werden, wenn das APOSS-Programm geöffnet und die Kommunikation hergestellt ist. Um mehrere Informationen mit einem einzigen PRINT Befehl auszugeben, müssen die einzelnen Elemente (Variablen, Texte etc.) durch ein Komma getrennt werden. Textinformationen müssen in Anführungszeichen gesetzt werden. Nach jeder PRINT Anweisung wird normalerweise ein Zeilenvorschub erzeugt. Dieser automatische Zeilenvorschub lässt sich durch einen Strichpunkt (;) nach dem letzten Ausgabeelement unterdrücken.
Befehlsgruppe	I/O
Querverweise	INKEY
Syntax-Beispiel	PRINT "Information ist wichtig !" /* Textinformation ausgeben */ PRINT "Information ist wichtig !"; /* Info ohne Zeilenumbruch ausgeben */ variable = 10 PRINT variable /* Variableninhalt ausgeben */ PRINT APOS /* Abgefragte Istposition ausgeben */ PRINT "Variable", variable,"Pos.:",APOS /* Gemischte Infos ausgeben */
Programmbeispiel	Verwendung siehe in allen Programmbeispielen.

□ PRINT DEV

Kurzinfo	Stoppt die Ausgabe von Informationen.
Syntax	PRINT DEV nn printlist
Parameter	nn = Ausgabeschnittstelle 0 = Standard Ausgabe -1 = Keine Ausgabe nach dieser Zeile printlist = normales Argument für einen PRINT Befehl
Beschreibung	PRINT DEV kann benutzt werden um alle PRINT Befehle in einem Programm zu deaktivieren, ohne jeden einzelnen Befehl kommentieren zu müssen. **ACHTUNG!:** Mit der Anweisung [-1] wird die Standard-Ausgabeschnittstelle neu definiert und gilt dann sofort für alle PRINTs, die kein DEV enthalten.
Befehlsgruppe	I/O
Querverweise	PRINT, INKEY
Syntax-Beispiel	PRINT DEV -1 "ab hier keine Ausgabe mehr" … PRINT "normaler Print " … PRINT DEV 0 "jetzt wieder Info ausgeben"

☐ PULSACC

Kurzinfo	Beschleunigung für den virtuellen Master setzen.
Syntax	PULSACC a
	 **ACHTUNG!:** Der Änderung der Beschleunigung in PULSACC wirkt erst nach dem nächsten PULSVEL Befehl.
Parameter	a = Beschleunigung in Hz/s
Beschreibung	Mit PULSACC setzen Sie die Beschleunigung für den virtuellen Master (Drehgeber-Ausgang). Das virtuelle Master-Signal bildet ein Drehgebersignal nach. Zur Berechnung der Pulsbeschleunigung PULSACC sind daher der Parameter *Drehgeberauflösung*, die Geschwindigkeit des Masters und die Rampenzeiten zu berücksichtigen. Die erzeugten Signale werden auch gleichzeitig als Master-Eingang ausgewertet, so dass MAPOS, MIPOS etc. wie in einem externen Master funktionieren. PULSACC = 0 ist Bedingung für das Abschalten des Modus „virtueller Master", vorausgesetzt es folgt ein PULSVEL Befehl.
Befehlsgruppe	SYN
Querverweis	PULSVEL
Beispiel	Das virtuelle Master-Signal soll einem Drehgebersignal von 1024 Strichen/Umdr. entsprechen. Die maximale Geschwindigkeit von 25 Drehgeber-Umdrehungen/s soll in 1 s erreicht werden. $\text{PULSACC} = \frac{\Delta \text{ Pulsgeschwindigkeit (PULSVEL) [Hz]}}{\Delta \text{ t [s]}}$ $= \frac{25 \text{ U/s} * 1024 \text{ Striche/U}}{1 \text{ s}}$ $= 25600 \text{ Striche/s}^2 = 25600 \text{ Hz/s}$

☐ PULSVEL

Kurzinfo	Geschwindigkeit für den virtuellen Master setzen.
Syntax	PULSVEL v
Parameter	v = Geschwindigkeit in Pulsen pro Sekunde [Hz]
Beschreibung	Mit PULSVEL setzen Sie die Geschwindigkeit für den virtuellen Master (Drehgeber-ausgang). Das virtuelle Master-Signal bildet ein Drehgebersignal nach. Zur Berechnung der Pulsgeschwindigkeit sind daher der Parameter *Drehgeberauflösung* und die Master-Geschwindigkeit zu berücksichtigen.
Befehlsgruppe	SYN
Querverweis	PULSACC
Beispiel	Das virtuelle Master-Signal soll einem Drehgebersignal von 2048 Strichen/Umdr. bei einer Drehgeberdrehzahl von 50 Umdr./s entsprechen. $\text{PULSVEL} = \text{Drehgeberstriche/Umdr.} * \frac{\text{Umdrehungen}}{\text{s}}$ = 2048 * 50 Hz = 102400 Hz

□ REPEAT .. UNTIL ..

Kurzinfo	Bedingte Schleife mit Endkriterium (Wiederhole … bis Bedingung erfüllt)
Syntax	REPEAT UNTIL Bedingung
Parameter	Bedingung = Abbruchkriterium
Beschreibung	Mit einer REPEAT **..** UNTIL Konstruktion kann man den dazwischen liegenden Programmbereich in Abhängigkeit von einem beliebigen Abbruchkriterium ein- oder mehrfach wiederholen. Das Abbruchkriterium setzt sich aus einer oder mehreren Vergleichsoperationen zusammen und wird stets am Schleifenende überprüft. Solange das Abbruchkriterium nicht erfüllt ist, wird der Schleifeninhalt wiederholt abgearbeitet. **ACHTUNG!:** Da das Abbruchkriterium erst am Schleifenende überprüft wird, werden die Befehle innerhalb der Schleifenkonstruktion mindestens einmal ausgeführt Um eine Endlosschleife zu vermeiden, müssen die innerhalb der Schleife abgearbeiteten Befehle direkt oder indirekt Einfluss auf das Ergebnis der Abbruchüberprüfung haben.
Befehlsgruppe	CON
Querverweise	LOOP, WHILE **..** DO **..** ENDWHILE
Syntax-Beispiel	REPEAT /* Schleife starten */ Befehlszeile 1 Befehlszeile n UNTIL (A !=1) /* Abbruchbedingung */
Programmbeispiel	REPEA_01.M, DIM_01.M, ONINT_01.M, OUT_01.M, INKEY_01.M

□ RST ORIGIN

Kurzinfo	Temporären Nullpunkt löschen.
Syntax	RST ORIGIN
Beschreibung	Mit dem RST ORIGIN Befehl wird ein zuvor mit SET ORIGIN gesetzter temporärer Nullpunkt wieder gelöscht, und alle folgenden absoluten Positionierbefehle (POSA) beziehen sich wieder auf den Realnullpunkt.
Befehlsgruppe	INI
Querverweise	SET ORIGIN, DEF ORIGIN, POSA
Syntax-Beispiel	RST ORIGIN /* temporären Nullpunkt zurücksetzen */
Programmbeispiel	TORIG_01.M, OUT_01.M, VEL_01.M

□ SAVE part

Kurzinfo	Arrays oder Parameter im EEPROM sichern.
Syntax	SAVE part part = ARRAYS, AXPARS, GLBPARS oder USRPARS
Beschreibung	Werden Array-Elemente oder Parameter geändert, während das Programm läuft, können die geänderten Werte mit diesen Befehlen im EEPROM gespeichert werden. SAVE GLBPARS Sichert die globalen Parameter der Gruppen 30-5* und 33-8*) und die Anwendungsparameter (Gruppe 19-**) im EEPROM. SAVE AXPARS Sichert alle anderen Achsenparameter. SAVE USRPARS Sichert nur Anwendungsparameter (Gruppe 19-**). **ACHTUNG!:** Das EEPROM kann diesen Befehl nur bis zu 10000-mal ausführen.
Befehlsgruppe	INI
Querverweise	DELETE ARRAYS, SAVEPROM

☐ SAVEPROM

Kurzinfo	Speicher in EEPROM sichern.
Syntax	SAVEPROM
Beschreibung	Wenn Arrayelemente oder Anwendungsparameter (Gruppe 19-**) geändert werden, während das Programm läuft, bietet SAVEPROM die Möglichkeit, die geänderten Werte zu speichern. Dies muss mit SAVEPROM explizit ausgelöst werden. Der Befehl löst denselben Vorgang aus, wie er auch aus dem Menü *Steuerung* heraus gestartet werden kann. Wenn Sie nur Array-Elemente oder nur globale Parameter und Benutzerparameter sichern wollen, benutzen Sie die entsprechenden Befehle SAVE .. ARRAY, GLBPAR oder USRPARS.  **ACHTUNG!:** Die Ausführungszeit von SAVEPROM hängt von der Menge der zu sichernden Daten ab. Es können bis zu 4 Sekunden sein.  **ACHTUNG!:** Bitte beachten Sie, dass Achsparameter nicht mit SAVEPROM gespeichert werden. Dazu müssen Sie den Befehl SAVE AXPARS benutzen.  **ACHTUNG!:** Das EEPROM kann diesen Befehl nur bis zu 10000-mal ausführen.
Befehlsgruppe	INI
Syntax-Beispiel	PRINT "Einen Moment Geduld" SAVEPROM PRINT "Danke"

☐ SET

Kurzinfo	Parameter setzen.
Syntax	SET par v
Parameter	par = Parameterkennung v = Parameterwert
Beschreibung	Mit dem SET Befehl können bestimmte Achsparameter und globale Parameter temporär während der Programmlaufzeit verändert werden. Die zulässigen Parameterkennungen finden Sie in der Parameter-Referenz.  **ACHTUNG!:** Die Parameteränderungen gelten nur solange das Programm läuft. Nach dem Programmende oder -abbruch sind wieder die ursprünglichen Parameter gültig. Die Parameteränderungen können mit dem Befehl SAVEPROM permanent gespeichert werden.
Befehlsgruppe	PAR
Portabilität	SET I_xxx Befehle funktionieren weiterhin und werden automatisch mit den neuen I_FUNKTION Parameter umgesetzt.
Querverweise	GET, Parameter-Referenz
Syntax-Beispiele	SET POSLIMIT 100000 /* Positive Wegbegrenzung setzen */ SET KPROP 150 /* Proportionalfaktor ändern */ SET PRGPAR 2 /* Aktivierte Programmnummer ändern */ SET I_FUNCTION_9_n /* ersetzt bisherigen Befehl SET I_BREAK */

☐ SETCURVE

Kurzinfo	CAM-Kurve setzen
Syntax	SETCURVE array
Parameter	array = Name des Arrays bzw. der Kurve
Beschreibung	Mit SETCURVE wird die CAM-Kurve, die in dem 'array' beschrieben ist, ausgewählt. SETCURVE array muss immer vor den Befehlen CURVEPOS, SYNCCxx, SYNCCSTART oder SYNCCSTOP. benutzt werden. Sobald der Befehl ausgeführt wird, sind die notwendigen Vorberechnungen bereits durchgeführt.  **ACHTUNG!:** Vor dem Befehl SETCURVE bzw. am Anfang des Programms muss die DIM-Anweisung mit dem Namen der Kurve bzw. des Arrays und der Anzahl der Array-Elemente stehen. Sind mehrere Arrays bzw. Kurven in der cnf-Datei, dann muss die Reihenfolge in der DIM-Anweisung mit der Reihenfolge der Arrays in der cnf-Datei übereinstimmen.  **ACHTUNG!:** <u>Wenn SYNCC nicht aktiv ist:</u> Wird SETCURVE benutzt, wenn SYNCC nicht aktiv ist, dann wird durch den Befehl SETCURVE die Kurven-Master-Position zurückgesetzt, und zwar abhängig von der aktuellen Master-Position. Das bedeutet, dass CMASTERCPOS (SYSVAR 4230) aus MAPOS. berechnet wird. Diese Position wird also nicht mehr durch SYNCC zurückgesetzt, sondern kann nur durch ein DEFMCPOS oder durch eine neue SETCURVE außerhalb des SYNCC-Modus zurückgesetzt werden. <u>Wenn SYNCC aktiv ist:</u> Wird SETCURVE aber benutzt, während SYNCC aktiv ist, wird CMASTERCPOS nicht verändert. Alle anderen Parameter wie 32-11 *Benutzerfaktor Zähler*, 32-12 *Benutzerfaktor Nenner*, 33-23 *Startverhalten für Sync.*, 33-15 und 33-16 *Markeranzahl Master* und *Slave*, 33-17 und 33-18 *Markerabstand Master* und *Slave*, 33-21 und 33-22 *Master und Slave-Marker Toleranzfenster* und alle Kurven-Array-Informationen werden nach dem nächsten Re-Start der Kurve aktualisiert. Während SYNCC aktiv ist, kann die Position CMASTERCPOS nur durch einen Befehl DEFMCPOS, der mit dem nächsten Re-Start der Kurve ausgeführt wird, oder MOVE SYNCORIGN, der sofort ausgeführt wird, beeinflusst werden. CMASTERCPOS (SYSVAR) und CURVEPOS werden nun auch aktualisiert, sogar wenn SYNCC nicht mehr aktiv ist. Diese Werte werden aktualisiert nach einem Befehl SETCURVE (falls SYNCMSTART < 2000 ist) oder nach SYNCC und dem ersten Master-Marker (falls SYNCMSTART = 2000).  **ACHTUNG!:** Das Übertragen des Arrays zum DSP kann einige ms dauern. Ein Kurvenarray mit 900 Werten dauert etwa 40 ms. Deshalb ist die maximale Größe eines Arrays auf 2000 begrenzt. (Die meisten Kurven haben ohnehin nicht mehr als einige Hundert Werte.)
Befehlsgruppe	PAR
Querverweise	DIM, CMASTERCPOS (SYSVAR), CURVEPOS,
Syntax-Beispiel	DIM curve [280] // siehe Anzahl der Elemente in der Titelleiste des CAM-Editors SETCURVE curve

□ SETMORIGIN

Kurzinfo	Beliebige Position als Nullpunkt für den Master setzen.
Syntax	SETMORIGIN wert
Parameter	wert = absolute Position
Beschreibung	Mit dem SETMORIGIN Befehl können Sie eine beliebige Position als neuen Nullpunkt für den Master setzen.
	ACHTUNG!: Der Befehl SETMORIGIN hebt den Befehl DEFMORIGIN auf.
	ACHTUNG!: Um den Nullpunkt für den Master wieder zu ändern, müssen Sie ihn daher mit SETMORIGIN oder DEFMORIGIN neu setzen. RST ORIGIN hat für den Nullpunkt des Masters keine Auswirkung.
Befehlsgruppe	INI
Querverweise	DEFMORIGIN, MAPOS
Syntax-Beispiel	SETMORIGIN 10000 /* Nullpunkt für den Master auf 10000 setzen */

□ SET ORIGIN

Kurzinfo	Absolute Position als Temporärnullpunkt setzen.
Syntax	SET ORIGIN p
Parameter	p = Absolute Position in Bezug zum Realnullpunkt
Beschreibung	Mit dem SET ORIGIN Befehl kann eine beliebige absolute Position vorübergehend als neuer Bezugspunkt für den absoluten Positionierbefehl (POSA) gesetzt werden. Diese Position wird Temporärnullpunkt genannt. In Verbindung mit dem Befehl CURVEPOS kann man festlegen, dass die aktuelle Slave-Position mit dem entsprechenden Wert der Kurve übereinstimmt.
	ACHTUNG!: Es ist möglich, mehrere SET ORIGIN ohne ein vorheriges RST ORIGIN auszuführen. Die absolute Positionsangabe bezieht sich immer auf den Realnullpunkt. Der letzte ausgeführte SET ORIGIN bestimmt somit die Lage des temporären Nullpunkts in Bezug zum Realnullpunkt.
Befehlsgruppe	INI
Querverweise	RST ORIGIN, DEF ORIGIN, POSA, CURVEPOS
Syntax-Beispiel	SET ORIGIN 50000 /* Temporärnullpunkt auf 50000 setzen */
Syntax-Beispiel	SET ORIGIN (–CURVEPOS) // Temporärnullpunkt auf den Anfang der Kurve setzen
Programmbeispiel	TORIG_01.M, OUT_01.M, VEL_01.M

□ SETVLT

Kurzinfo	Setzt einen FC 300 Parameter.
Syntax	SETVLT par v
Parameter	par = Parameternummer v = Parameterwert
Beschreibung	Mit dem SETVLT Befehl können bestimmte Parameter im FC 300 temporär verändert und somit auch die Konfiguration des FC 300 temporär werden. Da ausschließlich Ganzzahlenwerte übertragen werden, muss der zu übertragende Parameterwert mit dem zugehörigen Umwandlungsindex angepasst werden. Die Liste der FC 300-Parameter mit den zugehörigen Umwandlungsindizes finden Sie im FC 300 Produkthandbuch. **ACHTUNG!:** Die Parameteränderungen werden nur im RAM gespeichert. Nach dem Aus- und Wiedereinschalten werden wieder die ursprünglichen Parameter eingelesen.
Befehlsgruppe	PAR
Querverweise	GETVLT
Syntax-Beispiel	/* Par. 3-03 "Maximaler Sollwert" high auf 60 Hz ändern */ /* -Umwandlungsindex = -3 (multipliziert mit 10^3 beim Senden) */ SETVLT 3-03 60000

□ SETVLTSUB

Kurzinfo	Setzt einen FC 300-Parameter mit Indexnummer.
Syntax	SETVLTSUB par indxno v
Parameter	par = Parameternummer indxno = Indexnummer v = Parameterwert
Beschreibung	Mit SETVLT Befehlen können FC 300 Parameter und somit auch die Konfiguration des FC 300 vorübergehend geändert werden, in diesem Fall auch alle Parameter mit Indexnummer. Da ausschließlich Ganzzahlenwerte übertragen werden, muss der zu übertragende Parameterwert mit dem zugehörigen Umwandlungsindex angepasst werden. Die Liste der FC 300-Parameter mit den zugehörigen Umwandlungsindizes finden Sie im FC 300 Produkthandbuch. **ACHTUNG!:** Die Parameteränderungen werden nur im RAM gespeichert. Nach dem Aus- und Wiedereinschalten werden die ursprünglichen Parameter wieder eingelesen.
Befehlsgruppe	PAR
Querverweise	GETVLTSUB
Syntax-Beispiel	SETVLT 0-25 1 100 // Index 1 des Parameters 25 "Quick Menü" auf 100 "Konfiguration" setzen

☐ STAT

Kurzinfo	Status der Achse und der Steuerung abfragen.
Syntax	erg = STAT
Rückgabewert	erg = Achs- und Steuerungsstatus (4-Byte-Wert): Byte 3 MSB Bit 0 1 = Achse fährt Bit 1 1 = Überlauf Slave-Drehgeber Bit 2 1 = Überlauf Master-Drehgeber Bit 3 1 = Lageregelung ist temporär abgeschaltet *) Byte 2 Statusbyte der Lagereglereinheit Bit 7 1 = Lageregelung abgeschaltet Bit 2 1 = Position erreicht Bit 0,1,3-6 ohne Bedeutung Byte 1 wird nicht verwendet Byte 0 LSB Bit 7 1 = Endschalter aktiv Bit 6 1 = Referenzschalter aktiv Bit 5 1 = Startschalter aktiv Bit 2 1 = Lageregelung abgeschaltet Bit 0,1,3,4 nicht verwendet *) d.h. die Achse befindet sich innerhalb des Toleranzbereiches des Regelfensters REGWMAX / REGWMIN. Sobald das Regelfenster verlassen wird, wird die Lageregelung wieder eingeschaltet.
Beschreibung	Der STAT Befehl liefert den aktuellen Status der Lagereglereinheit sowie der abgefragten Achse zurück. Zum Beispiel ob die Lageregelung abgeschaltet, die Bewegung beendet oder der Endschalter aktiv ist. Der Zustand der Programmausführung kann nicht mit STAT, sondern nur mit AXEND abgefragt werden. Der Status wird aus vier Byte zusammengesetzt; Bedeutung siehe oben.
Befehlsgruppe	I/O
Querverweise	AXEND
Syntax-Beispiel	PRINT STAT /* Statuswort ausgeben */
Programmbeispiel	STAT_01.M

□ SUBMAINPROG .. ENDPROG

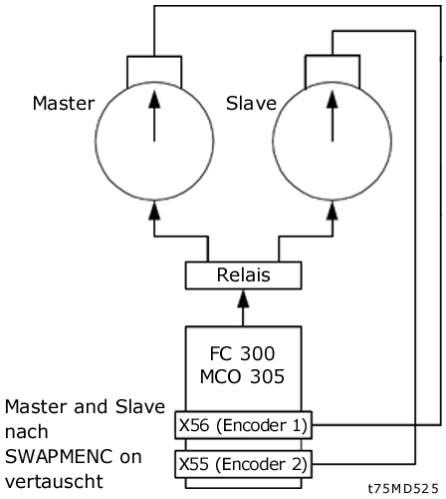
Kurzinfo	Definition des Unterprogrammbereichs.
Syntax	SUBMAINPROG ENDPROG
Beschreibung	Das Kennwort SUBMAINPROG leitet den Unterprogrammbereich ein, ENDPROG beendet diesen speziellen Programmbereich. Unter einem Unterprogramm versteht man Befehlsfolgen, die mit der GOSUB Anweisung von verschiedenen Programmpositionen aus aufgerufen und ausgeführt werden können. Innerhalb des Unterprogrammbereichs müssen alle verwendeten Unterprogramme enthalten sein. Es ist zwar möglich, den Unterprogrammbereich an beliebiger Stelle innerhalb eines Programms einzufügen, aus Gründen der Übersichtlichkeit wird jedoch empfohlen, ihn an den Anfang oder Ende des Programms zu stellen. **ACHTUNG!:** Es darf nur einen Unterprogrammbereich innerhalb eines Programms geben.
Befehlsgruppe	CON
Querverweise	SUBPROG .. RETURN, GOSUB, ON ERROR GOSUB, ON INT n GOSUB
Syntax-Beispiel	SUBMAINPROG /* Beginn des Unterprogrammbereichs */ Unterprogramm 1 Unterprogramm n ENDPROG /* Ende des Unterprogrammbereichs */
Programmbeispiel	GOSUB_01.M, AXEND_01.M, ERROR_01.M, INCL_01.M, STAT_01.M

□ SUBPROG name .. RETURN

Kurzinfo	Definition eines Unterprogramms.
Syntax	SUBPROG name RETURN
Parameter	name = Name der Unterprogramms
Beschreibung	Die Anweisung SUBPROG kennzeichnet den Beginn eines Unterprogramms. Direkt nach SUBPROG muss der Name des Unterprogramms stehen. Der Name kann aus einem oder mehreren Zeichen bestehen und muss eindeutig sein, das heißt es darf nicht mehrere Unterprogramme mit dem gleichen Namen geben. Mit der GOSUB Anweisung und dem Namen kann ein Unterprogramm zu jedem beliebigen Zeitpunkt aufgerufen und ausgeführt werden. Ein Unterprogramm kann beliebig viele Befehlszeilen enthalten und auf alle Programmvariablen zugreifen. Der letzte Befehl innerhalb eines jeden Unterprogramms muss die RETURN Anweisung sein, durch die das Unterprogramm verlassen und die Programmausführung mit dem auf die GOSUB Anweisung folgenden Befehl fortgesetzt wird.
	**ACHTUNG!:** Alle Unterprogramme müssen sich innerhalb des durch SUBMAINPROG und ENDPROG definierten Programmbereichs befinden. Es ist nicht zulässig innerhalb eines Unterprogramms ein zweites Unterprogramm zu deklarieren.
Befehlsgruppe	CON
Querverweise	SUBMAINPROG .. ENDPROG, GOSUB, ON ERROR GOSUB, ON INT .. n GOSUB
Syntax-Beispiel	SUBMAINPROG /* Beginn Unterprogrammbereich */ SUBPROG up1 /* Beginn von up1 */ Befehlszeile 1 Befehlszeile n RETURN /* Ende von up1 */ ENDPROG /* Ende Unterprogrammbereich */
Programmbeispiel	GOSUB_01.M, AXEND_01.M, ERROR_01.M, IF_01.M, STAT_01.M

☐ SWAPMENC

Kurzinfo	Master- und Slave-Drehgeber intern tauschen.
Syntax	SWAPMENC s
Parameter	s = Bedingung on = Eingang des Master-Drehgebers ist der Istwert-Eingang für die Regelung. off = Eingang des Slave-Drehgebers ist der Istwert-Eingang für die Regelung.
Beschreibung	Dieser Befehl erlaubt es, intern die Master- und Slave-Drehgeber zu vertauschen. Dies ist insbesondere dann sinnvoll, wenn man mit einer Steuerung zwei Motoren abwechselnd verwenden will. Vor dem Befehl SWAPMENC ON/OFF muss zur Vermeidung eines Schleppfehlers immer ein MOTOR OFF ausgeführt werden. Auch müssen die Regler- oder Achsparameter geändert werden, falls die beiden Motoren unterschiedlich sind. Die Motorleitungen werden über Relais umgeschaltet.



ACHTUNG!:
Bei diesem Wechsel gehen keine Positionen verloren, auch wenn die Motoren von Hand bewegt werden, während der andere Motor geregelt ist. Man hat auf den jeweils nicht geregelten Antrieb immer auch Zugriff über MAPOS.

Befehlsgruppe	INI
Querverweise	MAPOS
Syntax-Beispiel	SWAPMENC ON // Slave-Drehgeber intern mit Master-Drehgeber tauschen.

Beispiel

```
MOTOR OFF
OUT 1 1          // Motorleitungen umschalten
SET KPROP ...    // Achsparameter ändern
SWAPMENC ON      // intern Drehgeber umschalten
MOTOR ON         // Regelung wieder einschalten
POSA 10000
   // den am Master-Drehgeber angeschlossenen Motor verfahren
MOTOR OFF
OUT 1 0                 // Motorleitungen umschalten
SET KPROP ...           // Achsparameter ändern
SWAPMENC OFF            // intern Drehgeber umschalten
MOTOR ON                // Regelung wieder einschalten
POSA 0      // jetzt wieder den Motor, der am Slave-Drehgeber
            // angeschlossen ist verfahren
```

Normalerweise liest der untere Sockel X56 das Signal vom Master-Antrieb und X55 liest die Signale vom Slave-Antrieb. Nach einem SWAPMENC ON führt der Master eine Positionierung auf 1000 aus.

Nach einem SWAPMENC OFF führt dann der Slave diese Positionierung aus.
Die beiden Antriebe müssen nicht unbedingt Master und Slave sein. Es können auch zwei unterschiedliche Motoren mit einem FC 300 / MCO 305 sein.

☐ SYNCC

Kurzinfo	Synchronisation im CAM-Modus
Syntax	SYNCC num
Parameter	num = Anzahl der Kurven, die abgearbeitet werden sollen; 0 = der Antrieb bleibt im CAM-Modus bis mit Befehlen wie MOTOR STOP, CSTART, POSA etc. ein anderer Mode gestartet wird.
Beschreibung	Der Befehl SYNCC startet den CAM-Modus (Kurvenscheibensteuerung). Von diesem Augenblick werden die Kurvenpositionen des Masters hoch gezählt, und zwar abhängig von den tatsächlichen Master-Positionen und dem in Par. 33-23 definierten *Startverhalten für Sync*: Wo und wann angefangen wird, zu zählen. Mit dem Parameter SYNCMSTART = 2000 werden die Kurvenpositionen des Masters erst nach dem nächsten Master-Marker gezählt.

ACHTUNG!:
Der Befehl SYNCC startet weder den Slave, noch unterbricht er eine Fahrbewegung (z. B. CVEL); nur SYNCCSTART startet tatsächlich.

ACHTUNG!:
Der Antrieb bleibt solange im CAM-Modus bis *num* Kurven erfolgreich abgearbeitet wurden.

Wenn (nach *num* Kurven) die Synchronisation normal verlassen wird, wird – falls kein SYNCCSTOP mit einem entsprechenden Punktepaar definiert ist, – das Start-Stop-Punktepaar 2 benutzt, um den Antrieb anzuhalten. Dieser wird dann an der Position *slavepos* (siehe Parameter) stehen bleiben.

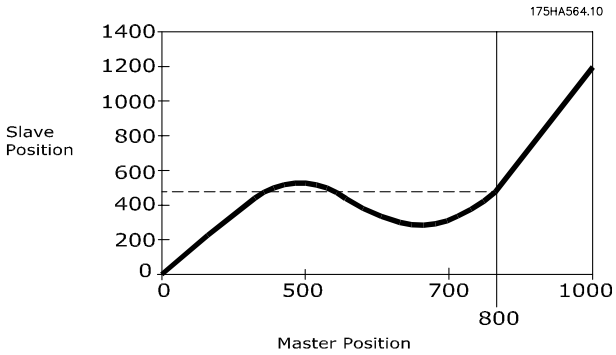
Befehlsgruppe	CAM
Querverweise	SYNCCSTART

Syntax-Beispiel

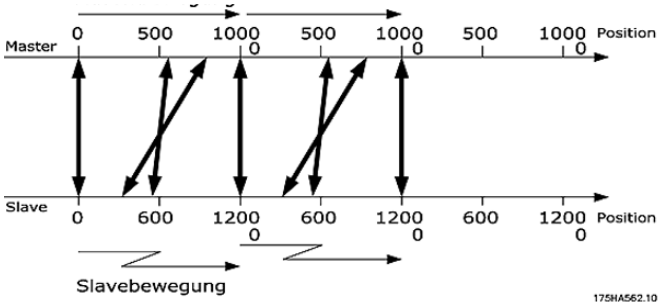
```
DIM curve [280]   // siehe Anzahl der Elemente in der Titelleiste des CAM-Editors
SETCURVE curve    // Kurve setzen
SYNCC             // Synchronisation im CAM-Modus
```

Beispiel Fixpunkte einer Kurve:

Master	Slave
0	0
500	500
700	300
1000	1200

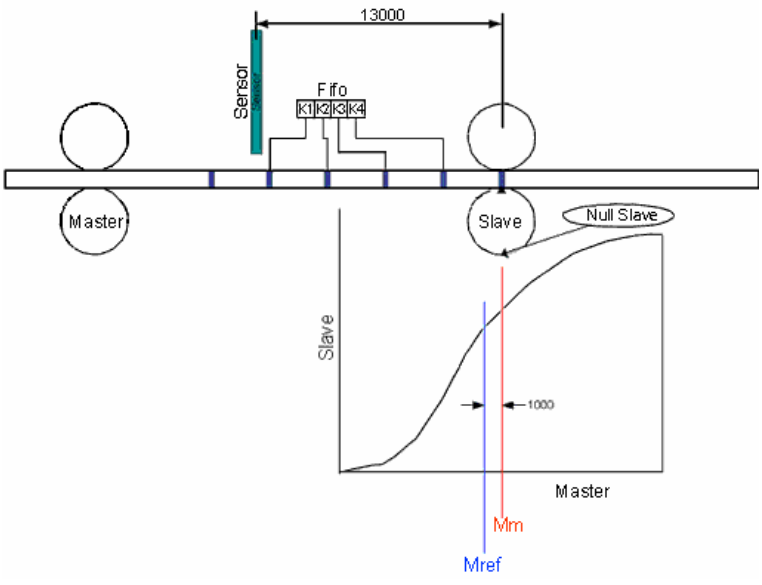


Demzufolge sperrt SYNCC 1 die Slave- und Master-Position gemäß der Array-Definition.



☐ SYNCCMM

Kurzinfo	Synchronisation im CAM-Modus mit Markerkorrektur.
Syntax	SYNCCMM num
Parameter	num = Anzahl der Kurven, die abgearbeitet werden sollen; 0 = der Antrieb bleibt im CAM-Modus bis mit Befehlen wie MOTOR STOP, CSTART, POSA etc. ein anderer Mode gestartet wird
Beschreibung	Der Befehl SYNCCMM bewirkt wie SYNCC eine Synchronisation im CAM-Modus, führt aber zusätzlich eine Markerkorrektur (nur, wenn der Master vorwärts fährt) durch. Um den Abstand zwischen Sensor und Arbeitspunkt zu speichern, wird der Par. 33-17 *Markerabstand Master* benutzt. Damit kann die Markerposition ohne Änderung der Kurve korrigiert werden. Und es sind auch größere Sensorabstände als die eigentliche Kurvenlänge möglich. In diesem Fall wird für die Markerkorrektur ein FIFO benutzt (siehe Beispiel). Der Marker kann der Nullimpuls des Drehgebers oder ein externes 24-Volt-Signal sein. **ACHTUNG!:** Der Befehl SYNCCMM startet weder den Slave, noch unterbricht er eine Fahrbewegung (z. B. CVEL); nur SYNCCSTART startet tatsächlich. **ACHTUNG!:** Der Antrieb bleibt solange im CAM-Modus bis 'num' Kurven erfolgreich abgearbeitet wurden. Wenn (nach 'num' Kurven) die Synchronisation normal verlassen wird, wird – falls kein SYNCCSTOP mit einem entsprechenden Punktepaar definiert ist, – das Start-Stop-Punktepaar 2 benutzt, um den Antrieb anzuhalten. Dieser wird dann an der Position 'slavepos' (siehe Parameter) stehen bleiben.
Befehlsgruppe	CAM
Querverweis	Par. 33-17 *Markerabstand Master*
Syntax-Beispiel	SETCURVE curve SYNCCMM 1 // 1 x Synchronisieren im CAM-Modus mit Markerkorrektur
Beispiel	Wenn zum Beispiel die Kurvenlänge 3000 und der Abstand des Sensors zum Arbeitspunkt 13000 ist, gibt es ein FIFO mit 4 Registern und einen Offset von 1000, der betrachtet werden muss.



☐ SYNCCMS

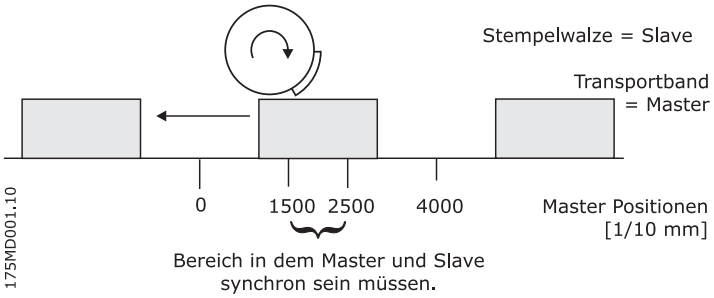
Kurzinfo	Synchronisation im CAM-Modus mit Markerkorrektur des Slaves.
Syntax	SYNCCMS num
Parameter	num = Anzahl der Kurven, die abgearbeitet werden sollen; 0 = der Antrieb bleibt im CAM-Modus bis mit Befehlen wie MOTOR STOP, CSTART, POSA etc. ein anderer Mode gestartet wird.
Beschreibung	Der Befehl SYNCCMS bewirkt wie SYNCC, eine Synchronisation im CAM-Modus, führt aber zusätzlich eine Markerkorrektur des Slaves durch. Hier wird nicht die Kurvenposition korrigiert, sondern die Slave-Position. Es wird – im Gegensatz zu SYNCCMM,– kein FIFO gebildet. Der Marker kann der Nullimpuls des Drehgebers oder ein externes 24-Volt-Signal sein.  **ACHTUNG!:** Der Befehl SYNCCMS startet weder den Slave, noch unterbricht er eine Fahrbewegung (z.B. CVEL); nur SYNCCSTART startet tatsächlich.  **ACHTUNG!:** Der Antrieb bleibt solange im CAM-Modus bis 'num' Kurven erfolgreich abgearbeitet wurden. Wenn (nach 'num' Kurven) die Synchronisation normal verlassen wird, wird – falls kein SYNCCSTOP mit einem entsprechenden Punktepaar definiert ist, – das Start-Stop-Punktepaar 2 benutzt, um den Antrieb anzuhalten. Dieser wird dann an der Position 'slavepos' (siehe Parameter) stehen bleiben.
Befehlsgruppe	CAM
Querverweise	Par. 33-18 *Markerabstand Slave*
Syntax-Beispiel	SETCURVE curve SYNCCMS 0 // Synchronisieren im CAM-Modus mit Markerkorrektur des Slaves.

☐ SYNCCSTART

Kurzinfo	Slave zur Synchronisation im CAM-Modus starten.
Syntax	SYNCCSTART pnum
Parameter	pnum = Start-(Stop-)Punktepaar-Nummer pnum > 0 Bei Erreichen des entsprechenden A-Punktes wird mit dem Einkuppeln begonnen; vorausgesetzt, der Master fährt in positiver Richtung; die Einkuppelkurve wird im B-Punkt beendet. Wenn A- und B-Punkt identisch sind, wird der Slave mit der eingestellten Maximalgeschwindigkeit – also ohne Kurve – eingekuppelt, sobald der Master diesen Punkt erreicht hat. pnum = 0 Der Slave wird sofort mit der eingestellten Maximalgeschwindigkeit eingekuppelt (aufsynchronisiert). Dabei ist es egal, in welcher Richtung der Master fährt und ob er überhaupt fährt. pnum < 0 Es wird ebenfalls das entsprechende Punktepaar verwendet, allerdings beginnt das Einkuppeln beim B-Punkt und wird beim A-Punkt – also in negativer Richtung beendet.
Beschreibung	Der Befehl startet die Bewegung des Slaves. Mit 'pnum' wählt man das Punktepaar aus, das festlegt, an welcher Master-Position die Synchronisation startet und wo sie beendet sein soll. Beim Vorwärtsfahren startet die Synchronisation am A-Punkt und wird bis zum B-Punkt beendet. Beim Rückwärtsfahren wird sie am B-Punkt gestartet und bis zum A-Punkt beendet.
Befehlsgruppe	CAM
Querverweise	SETCURVE, Start-Stop-Punkte
Syntax-Beispiel	SETCURVE curve SYNCC 0 // Synchronisieren im CAM-Modus SYNCCSTART 1 // Slave am A-Punkt vom Start-Stop-Punktepaar 1 einkuppeln

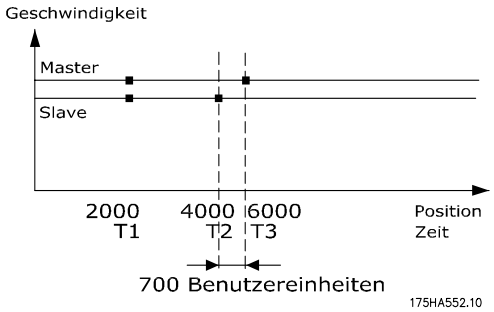
□ SYNCCSTOP

Kurzinfo	Slave nach der CAM-Synchronisation anhalten.
Syntax	SYNCCSTOP pnum slavepos
Parameter	pnum = (Start-)Stop-Punktepaar
	pnum > 0 Bei Erreichen des entsprechenden A-Punktes wird mit dem Auskuppeln begonnen; vorausgesetzt, der Master fährt in positiver Richtung; die Auskuppelkurve wird im B-Punkt beendet. Wenn A- und B-Punkt identisch sind, wird der Slave mit der eingestellten Maximalgeschwindigkeit – also ohne Kurve – ausgekuppelt, sobald der Master diesen Punkt erreicht hat.
	pnum = 0 Der Slave wird sofort mit der eingestellten Maximalgeschwindigkeit ausgekuppelt. Dabei ist es egal, in welcher Richtung der Master fährt und ob er überhaupt fährt.
	pnum < 0 Es wird ebenfalls das entsprechende Punktepaar verwendet, allerdings beginnt das Auskuppeln beim B-Punkt und wird beim A-Punkt – also in negativer Richtung beendet.
	slavepos = Position, an der der Slave nach dem Auskuppeln stehen soll.
	ACHTUNG!: Beim Vorwärtsfahren beginnt das Auskuppeln am A-Punkt und endet am B-Punkt; beim Rückwärtsfahren umgekehrt.
	ACHTUNG!: Wird das Programm ohne SYNCCSTOP Befehl verlassen, wird standardgemäß mit dem zweiten Punktepaar zum ausgekuppelt und an der in den → *Kurvendaten* definierten Slave-Stop-Position angehalten.
Beschreibung	Der Befehl stoppt die Synchronisation ohne den SYNCC Modus zu verlassen. Der Slave wird entsprechend dem Punktepaar, das in 'pnum' definiert ist, ausgekuppelt. Dann erst wird der Slave tatsächlich angehalten. Wenn der Stopp-Punkt erreicht ist, muss der Slave auf 'slavepos' sein.
Befehlsgruppe	CAM
Querverweise	Slave-Stop-Position
Syntax-Beispiel	SETCURVE curve SYNCC 0 // Synchronisieren im CAM-Modus SYNCCSTART 1 // Slave mit Start-Punktepaar 1 starten SYNCCSTOP 2 0 // Slave mit Stopp-Punktepaar 2 an der // Slave-Position 0 bzw. 3600 anhalten
Beispiel	Slave Positionen[Grad] Stempel Anfang 120°, Ende 240°



□ SYNCERR

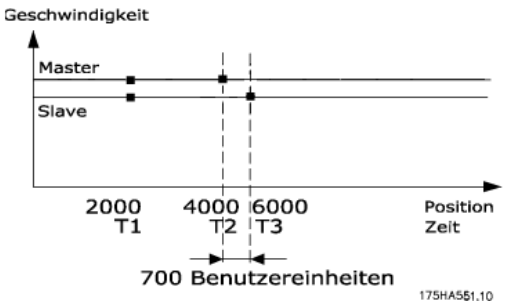
Kurzinfo	Aktuellen Synchronisationsfehler einer Achse bzw. des Slaves abfragen.
Syntax	erg = SYNCERR
Rückgabewert	erg = aktueller Synchronisationsfehler des Slaves in BE [qc] bzw. im CAM-Modus in BE und a) als absoluter Wert, wenn in Par. 33-13 SYNCACCURACY die Größe des Genauigkeitsfensters mit positivem Vorzeichen definiert ist; b) mit Vorzeichen, wenn in SYNCACCURACY die Größe des Fensters mit negativem Vorzeichen definiert ist.
Beschreibung	SYNCERR liefert den aktuellen Synchronisationsfehler in qc bzw. im CAM-Modus in Benutzereinheiten BE zurück. Das ist der Abstand zwischen der aktuellen Master-Position (umgerechnet mit Getriebefaktor und Offset) und der Istposition der entsprechenden Achse bzw. des Slaves. Wenn der Par. 33-13 *Genauigkeitsfenster für Positionssynchronisation* mit negativem Vorzeichen definiert wird, können Sie auch feststellen, ob die Synchronisation vorausläuft (negatives Ergebnis) oder hinterherläuft (positives Ergebnis).
	ACHTUNG!: Bis Optionskarte Version < 5.00: SYNCERR funktioniert nur im Synchronisationsbetrieb. Sobald man SYNCM oder SYNCP verlässt, werden die Pulse nicht mehr gezählt. SYNCERR wird nur innerhalb eines Synchronisationsbefehls aktualisiert. Mit Optionskarte ab Software 5.00: SYNCERR wird (ebenso wie die interne Master-Sollposition G_Mpcmd) aktualisiert, wenn SYNCP oder SYNCM nicht mehr aktiv sind, z.B. nach einem MOTOR STOP.
Befehlsgruppe	I/O
Querverweise	TRACKERR, MAPOS, APOS, Parameter: 33-12 *Positionsoffset für Synchronisation* (SYNCPOSOFFS), 33-10 und 33-11 *Synchronisationsfaktor Master* und *Slave*, 33-13 *Genauigkeitsfenster für Positionssynchronisation* (SYNCACCURACY)
Syntax-Beispiel	PRINT SYNCERR /* aktuellen Synchronisationsfehler abfragen */
Beispiele	SYNCACCURACY = 1000 Demzufolge gibt SYNCERR den absoluten Wert 700 zurück.



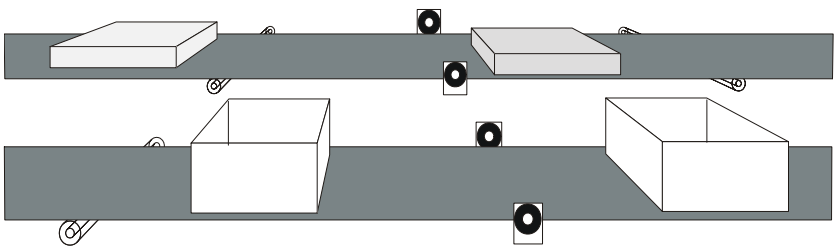
SYNCACCURACY = 1000

SYNCERR wird den absoluten Wert 700 anzeigen, auch wenn der Slave dem Master voraus ist.

Daher gibt SYNCERR den Wert –700 zurück und zeigt damit, dass der Slave vor dem Master ist.

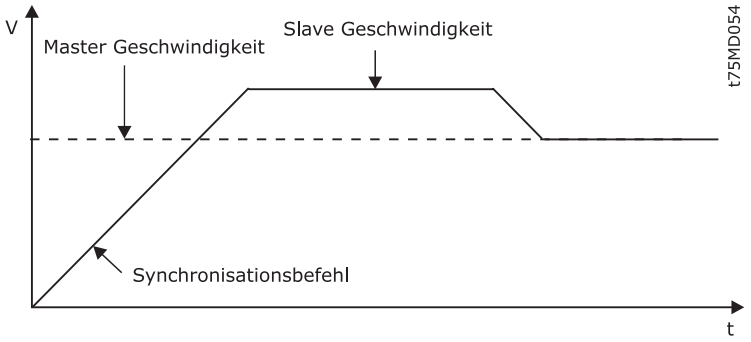


☐ SYNCM

Kurzinfo	Winkel-/Positionssynchronisation mit dem Master mit Markerkorrektur.
Syntax	SYNCM
Beschreibung	Der Befehl SYNCM bewirkt genau wie der folgende SYNCP Befehl eine Winkel-/Positionssynchronisation mit dem Master, führt aber zusätzlich eine Markerkorrektur durch. Dabei wird bereits während des Anlaufs der Synchronisation auf den nächsten errechneten Marker aufsynchronisiert. So kann man zum Beispiel unterschiedliches Laufverhalten wie Schlupf mit ausgleichen. Nachdem die Synchronisation hergestellt ist, wird bei jedem Marker überprüft, welche Abweichung vorliegt (oder bei jedem n-ten Marker, falls die Markeranzahl für Master und Slave nicht identisch ist). Diese wird als neuer Offset in die Regelung eingebracht und es wird sofort versucht, die Abweichung auszugleichen. Dabei werden allerdings die eingestellten Werte für Geschwindigkeit VEL sowie Beschleunigung ACC oder DEC nicht überschritten **ACHTUNG!:** Zu den Parametern, die bei der SYNCP schon verwendet werden, sind hier auch noch Par. 33-25 SYNCREADY und Par. 33-24 SYNCFAULT von Bedeutung. **ACHTUNG!:** Da folgende Parameter zu einer Überbestimmung führen können, sollten Sie darauf achten, dass die Werte sinnvoll sind, zueinander passen und mit denen der Getriebefaktoren konsistent sind. Par. 33-15, 33-16 *Markeranzahl Master* und *Slave* Par. 33-17, 33-18 *Markerabstand Master* und *Slave* Par. 33-19, 33-20 *Markertyp Master* und *Slave* **ACHTUNG!:** SYNCM sollten Sie nur einmal aufrufen, denn die Synchronisierung läuft bis zum nächsten Fahr- oder Stoppbefehl. Jeder weitere SYNCM Befehl führt dazu, dass die Synchronisation von vorne beginnt, was normalerweise nicht beabsichtigt ist, außer Sie setzen den aktuellen SYNCERR zurück. Wenn in Par. 33-23 *Startverhalten bei Sync.* definiert, wird beim Start von SYNCM auf die erste Auswertung der Markerpulse gewartet und erst dann der in Par. 33-12 definierte *Positionsoffset für Synchronisation* angewandt.
Marker Signal	Der Marker kann der Nullimpuls des Drehgebers oder ein externes 24-Volt-Signal sein. (I5 = Master; I6 = Slave)
Befehlsgruppe	SYN
Syntax-Beispiel	SYNCM /* Synchronisieren der Position mit Markerkorrektur */
Beispiel	 Selbst wenn beide Bänder synchron laufen, würden die Deckel nie auf gleicher Höhe mit den Schachteln sein. Mit SYNCM wird durch die Auswertung der externen Marker die Positionsabweichung zwischen Master und Slave ermittelt und ausgeglichen.

☐ SYNCP

Kurzinfo	Winkel-/Positionssynchronisation mit dem Master.
Syntax	SYNCP
Beschreibung	Der Befehl SYNCP führt eine Winkel-/Positionssynchronisation mit dem Master durch. Dabei wird die Position entsprechend der Getriebefaktoren zum Master synchron gehalten, das heißt bei einer Störung von außen wird anschließend versucht, die entsprechende Strecke wieder aufzuholen.



Dabei werden allerdings die eingestellten Werte für Geschwindigkeit VEL sowie Beschleunigung ACC oder DEC nicht überschritten.

Die folgenden Parameter beeinflussen das Verhalten:
Par. 33-10 *Synchronisationsfaktor Master* und
Par. 33-11 *Synchronisationsfaktor Slave* (Getriebefaktoren)
Par. 33-12 *Positionsoffset für Synchronisation*
Par. 33-13 *Genauigkeitsfenster für Positionssync.* (Genauigkeit für Flag)
Par. 32-68 *Reversierungsverhalten Slave*

Beim Aufsynchronisieren wird wie folgt vorgegangen:

Wenn der Befehl SYNCP startet, wird die aktuelle Master-Position ermittelt und als Bezugsposition festgehalten. Aus der Master-Geschwindigkeit wird – unter Berücksichtigung der erlaubten Beschleunigung – die erforderliche Slave-Geschwindigkeit errechnet, um die Master-Position zu erreichen. Der Slave wird solange beschleunigt, bis die errechnete Position erreicht ist oder bis er dicht genug an der Bezugsposition ist, um diese zu erreichen.

ACHTUNG!:
Sobald die Abweichung zwischen Slave- und Master-Position kleiner als Par. 33-13 SYNCACCURACY ist, wird das ACCURACY-Flag gesetzt.

Wenn Par. 32-68 REVERS so eingestellt ist, dass Zurückfahren nicht erlaubt ist, aber aus irgendeinem Grund der Slave weiter als der Master ist (z.B. weil nur der Master rückwärts gefahren ist), wartet der Slave mit Geschwindigkeit 0 solange, bis die entsprechende Bezugsposition kommt. Dabei berücksichtigt der Slave seine Beschleunigungszeit und fährt ggf. schon los, bevor der Master die Slave-Position erreicht hat.

Statt diesem Aufholverfahren kann man auch zuerst den Slave mit CVEL auf die ungefähre Master-Geschwindigkeit bringen und dann SYNCP auslösen.

Eine Veränderung des Par. 33-12 *Positionsoffset für Synchronisation* während des Aufholens führt zu erneutem Aufsynchronisieren mit Rampen (siehe oben).

ACHTUNG!:
SYNCP sollten Sie nur einmal aufrufen, denn die Synchronisierung läuft bis zum nächsten Fahr- oder Stoppbefehl. Jeder weitere SYNCP Befehl führt dazu, dass die Synchronisation von vorne beginnt, was normalerweise nicht beabsichtigt ist, außer Sie setzen den aktuellen SYNCERR zurück.

Befehlsgruppe	SYN
Syntax-Beispiel	SYNCP /* Normales Synchronisieren der Position */ CVEL 50 /* Vor dem Synchronisieren Geschwindigkeit erreichen */ CSTART WAITT 500 SYNCP

□ SYNCSTAT

Kurzinfo	Flag für Synchronisationsstatus abfragen.
Syntax	erg = SYNCSTAT
Rückgabewert	erg = Synchronisationsstatus mit folgender Bedeutung:

	Wert	Bit
Par. 33-25 SYNCREADY	1	0
Par. 33-24 SYNCFAULT	2	1
Par. 33-13 SYNCACCURACY	4	2
SYNCMMHIT	8	3
SYNCSMHIT	16	4
SYNCMMERR	32	5
SYNCSMERR	64	6

Beschreibung

Folgende Flags sind definiert und können mit SYNCSTAT abgefragt werden: READY, FAULT, ACCURACY und jeweils für Master und Slave die MHIT und MERR.

SYNCACCURACY

Jede ms wird geprüft ob SYNCERR < SYNCACCURACY gilt und falls dies erfüllt ist, wird SYNCACCURACY (56) gesetzt, andernfalls wird das Flag zurückgesetzt. Diese Überprüfung findet sowohl bei SYNCP als auch bei SYNCM statt.

Dieses Flag wird nicht bei SYNCV verwendet.

Beim Ausführen eines SYNCP oder SYNCM Befehls wird dieses Flag zurückgesetzt.

SYNCFAULT / SYNCREADY

Bei jedem SYNCP bzw. SYNCM Befehl werden diese Flags zurückgesetzt. Danach wird bei jedem Markerpuls des Slaves (SYNCP) bzw. bei Vorhandensein eines Markerpulses des Masters und eines Markerpulses des Slaves (SYNCM) geprüft, ob SYNCACCURACY gesetzt ist oder nicht.

Wenn es gesetzt ist, wird der Ready-Zähler erhöht und der Fault-Zähler auf 0 gesetzt, andernfalls wird der Fault-Zähler erhöht und der Ready-Zähler auf 0 gesetzt.

Ist der Ready-Zähler größer als der in Par. 33-25 SYNCREADY vorgegebene Wert, wird das Flag SYNCREADY gesetzt, im anderen Fall wird es zurückgesetzt.

Ist der Fault-Zähler größer als der in Parameter SYNCREADY bzw. SYNCFAULT, wird das Flag SYNCFAULT gesetzt, andernfalls wird es zurückgesetzt.

SYNCMMHIT / SYNCSMHIT

SYNCMMHIT und SYNCSMHIT werden gesetzt, wenn der Master-Marker bzw. Slave-Marker erkannt wird. Bei jedem SYNCM Befehl werden diese Flags zurückgesetzt. Danach wird nach dem ersten Auftauchen eines Markerpulses bzw. beim n-ten Markerpuls (Par. 33-15 *Markeranzahl Master*) das Flag SYNCMMHIT gesetzt.

Analoges gilt für SYNCSMHIT beim Slave.

ACHTUNG!:
Dieses Flag wird nicht mehr zurückgesetzt, es sei denn SYNCM wird neu gestartet oder mit dem Befehl SYNCSTATCLR explizit gelöscht.

SYNCMMERR / SYNCSMERR

Wenn in den Parametern 33-21 oder 33-22 ein *Master-* oder *Slave-Marker Toleranzfenster* definiert ist, werden SYNCMMERR bzw. SYNCSMERR gesetzt, sobald die maximal erlaubte Distanz erreicht ist und kein Marker erkannt wurde.

	Beispiel: Abstand zwischen 2 Master-Markern Par. 33-17 *Markerabstand* = 30000 Par. 33-21 *Master-Marker Toleranzfenster* = 1000 Das Flag wird also bei 31000 gesetzt, wenn kein Marker erkannt wurde. Bei jedem SYNCM Befehl werden diese Flags gelöscht. Wenn *Master-Marker Toleranzfenster* = 0 und damit kein Toleranzfenster definiert ist, wird bei jedem Markerpuls (bzw. bei jedem n-ten) geprüft, ob der Abstand zwischen den zwei zuletzt registrierten Markern kleiner ist als das 1,8-fache des in Par. 33-17 vorgegebenen *Markerabstands*. Wenn nicht, wird das entsprechende Flag gesetzt.  **ACHTUNG!:** Diese Flags werden automatisch zurückgesetzt: Bei der nächsten erfolgreichen Markerkorrektur und bei erneutem Start von SYNCM oder durch den Befehl SYNCSTATCLR.
Befehlsgruppe	SYN
Querverweise	SYNCSTATCLR

Syntax-Beispiel	IF (SYNCSTAT & 4) THEN OUT 1 1
	/* Wenn ACCURACY dann Ausgang setzen */
	ENDIF

□ SYNCSTATCLR

Kurzinfo	Zurücksetzen der Flags MERR und MHIT.
Syntax	SYNCSTATCLR wert Der SYNCSTATCLR Befehl sollte nur in einem Unterprogramm zur Fehlerbehandlung eingesetzt werden (siehe ON ERROR GOSUB).
Parameter	wert = 8 = SYNCMMHIT 16 = SYNCSMHIT 32 = SYNCMMERR 64 = SYNCSMERR
Beschreibung	Mit SYNCSTATCLR kann man die dem 'wert' entsprechenden Bits im SYNCSTAT zurücksetzen und damit die Fehler-Flags MERR und die HIT-Flags MHIT löschen. Alle anderen Flags können nicht verändert werden.
Befehlsgruppe	SYN
Querverweise	ON STATBIT, ON ERROR GOSUB, ERRNO, CONTINUE, MOTOR ON
Syntax-Beispiel	SYNCSTATCLR 32 /* aktuelle Fehlermeldung löschen */

☐ SYNCV

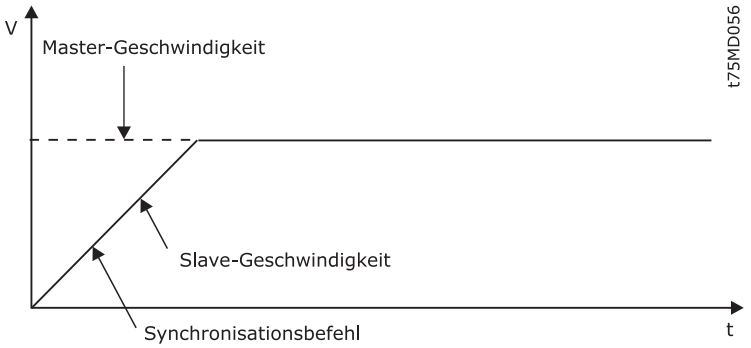
Kurzinfo	Geschwindigkeitssynchronisation mit dem Master.
Syntax	SYNCV

ACHTUNG!:
SYNCM sollten Sie nur einmal aufrufen, denn die Synchronisierung läuft bis zum nächsten Fahr- oder Stoppbefehl. Jeder weitere SYNCM Befehl führt dazu, dass die Synchronisation von vorne beginnt, was normalerweise nicht beabsichtigt ist, außer Sie setzen den aktuellen Synchronisationsfehler SYNCERR zurück.

ACHTUNG!:
Schlepp- und Synchronisationsfehler werden im SYNCV Modus nicht überwacht; daher wird empfohlen die Hardware-Drehgeberüberwachung zu benutzen.

Beschreibung

Mit SYNCV wird eine Geschwindigkeitssynchronisation mit dem Master durchgeführt, zum Beispiel nach einer Störung von außen. Dabei wird ausschließlich die Geschwindigkeit betrachtet und nicht versucht, die Position aufzuholen.



Beim Aufsynchronisieren sowie während der Synchronisation wird darauf geachtet, dass weder die voreingestellte Geschwindigkeit VEL noch die voreingestellte Beschleunigung ACC oder DEC überschritten wird.

Für die Synchronisation werden die Parameter der Getriebefaktoren verwendet: Par. 33-10 *Synchronisationsfaktor Master*, Par. 33-11 *Synchronisationsfaktor Slave*.

Außerdem werden die Geschwindigkeiten nicht einfach durch Differenz der aktuellen minus der letzten Geschwindigkeit ermittelt (Master/Slave), sondern die Werte werden entsprechend der Einstellungen Par. 33-26 *Geschwindigkeitsfilter* (SYNCVFTIME) gefiltert. Dabei wird der Filter für den Slave aus der Maximalgeschwindigkeit bestimmt.

Das heißt, VELMAX * 5 entspricht der Drehgeberauflösung für die Filtertabelle, wobei VELMAX die Geschwindigkeit in qc/ms ist. (Die Formel ergibt sich unter der Annahme, dass die Filtertabelle für die Drehgeberauflösung mit einer Maximalgeschwindigkeit von 3000 U/Min gemacht wurde.)

Beim Übergang von der Geschwindigkeitsregelung zur Positionsregelung wird versucht, dies möglichst ruckfrei auszuführen. Dazu wird die neue Sollposition so definiert, dass gilt:

command_pos = actual_pos + error

old_error, CVEL und AVEL werden beibehalten.

Befehlsgruppe	SYN
Querverweise	Parameter der Gruppe AXS.

☐ SYSVAR

Kurzinfo	Systemvariable (Pseudo-Array) liest Systemwerte.
Syntax	SYSVAR[n] n = index
	 **ACHTUNG!:** Die Werte der Systemvariablen sind interne, hardwareabhängige Werte, die sich verändern können.
Beschreibung	Mit der Systemvariablen SYSVAR – einem vorbereitetem Pseudo-Array – können Sie detaillierte Systeminformationen lesen. Diesen Index benötigen Sie auch, wenn Sie mit LINKSYSVAR die Systemvariable mit dem LCP-Display verknüpfen oder Aufzeichnungsdaten einer Testfahrt mit TESTSETP festlegen.
Befehlsgruppe	CON
Querverweise	LINKSYSVAR, TESTSETP

System- und Achsprozessdaten

Index	Beschreibung Systemprozessdaten
1	Eingangs-Byte 0 (Eingänge 1..8 von MCO 305)
2	Eingangs-Byte 1 (Eingänge 18..33 von CC)
3	Eingangs-Byte 2 (Eingänge 9..10 / 12 von MCO 305)
9	Ausgangs-Byte 0
17	Oberen 2 Byte die vom APOSS Befehl STAT geliefert werden
22	Interner Millisekunden Zähler: Wert, den auch der APOSS Befehl TIME liefert.
28	Aktueller Motorstrom [1/100 A]; (Parameter 16-14)
30	Motorspannung [1/10 V]; (Parameter 16-12)
31	FC 300 Status (Parameter 16-03)
32	Hauptistwert (HIW) (Parameter 16-05)
33	Aktuelle Zeilennummer des APOSS Programms, falls mit #DEBUG NOSTOP gearbeitet wurde.
34	Motorfrequenz (Parameter 16-13)
35	Motordrehmoment (Parameter 16-16)

Index	Beschreibung Achsprozessdaten
4096	Istposition Slave [qc] (vgl. APOS [BE])
4097	Sollposition Slave [qc] (vgl. CPOS [BE])
4098	Letzte Slave Indexposition [qc] (vgl. IPOS [BE])
4099	Aktuelle Geschwindigkeit in qc/st, wobei [st] die in _GETVEL gesetzte Abtastzeit ist.
4100	Aktuelle Master-Geschwindigkeit (wie oben).
4101	Aktueller Positionsfehler in [qc].
4102	Enthält nach dem ersten Überlauf des Absolutgebers die Anzahl Umdrehungen des Drehgebers, vorausgesetzt die Strichzahl pro Umdrehung ist in Par. 32-01 *Inkrementalgeber Auflösung* (ENCODER) richtig eingetragen.
4103	Wie oben für den Master.
4105	Istposition Master [qc] (vgl. MAPOS [BE])
4106	Letzte Master Indexposition [qc] (vgl. MIPOS [BE])
4107	Interne aktuelle Geschwindigkeit (ACTPOS – letzte ACTPOS) (qc/1 ms)
4108	Interne Master-Geschwindigkeit (siehe oben).
4109	Aktuelle Frequenz der Master-Simulation (1/1000 Hz) (siehe PULSVEL)
4110	Gibt an, ob Master-Simulation aktiv ist oder nicht (1 bzw. 0).
4111	Aktueller Sollwert der von der Steuerung durch den Lageregler ausgegeben wird (zwischen –FFFFF und FFFFF bzw. –1048575 und 1048575 dezimal).
4113	Aktuell verwendeter Timer für die PID-Schleife (TIMER).
4114	Aktuell verwendeter Timer für den Profilgenerator (PROFTIME).
4115	Gibt an, ob negative Sollwerte ausgegeben werden (!=0) oder nicht (=0).
4116	Gibt an ob Sollwert mit 0-10V und Richtungsausgang ausgegeben wird (>0). Falls ja enthält dieser Parameter die Ausgangsnummer (1..8) die verwendet wird.
4117	Aktuelle Beschleunigung des virtuellen Masters.
4118	Zielfrequenz für virtuellen Master (Einheit siehe oben).
4119	Vlamode (absolut / relative ?)
4120	Anzahl der qc zwischen Index-Impulsen.
4121	Typ des Z-Impulses *Markertyp Marker* (SYNCMTYPM).
4122	Benutzer-Sollwert, gegeben durch OUTAN, Skalierung siehe 4111.
4123	Interner Parameter: Drehgebertyp Slave (ENCDODERTYPE = 0..2)
4124	Interner Parameter: Drehgebertyp Master (MENCODERTYPE = 0..6)
4125	Drehgeberauflösung Slave
4126	Drehgeberauflösung Master
4127	Gibt an, ob der Verstärker bei Motor OFF auf Stopp-Modus (!=0) oder Warten (==0) gesetzt ist.
4128	Interner Parameter: Liefert die gleiche Information wie MAVEL, jedoch immer die Information des echten Drehgebers, auch wenn der Master simuliert wird (MENCODERTYPE == 6).

Index	Beschreibung Achsprozessdaten, Profilgenerator-Werte		
4218	Gibt alle 32 Flags des Profilgenerators aus. Diese sind:		
	PG_FLAG_BUSY	1L	// Flag für Busy Information
	PG_FLAG_COMMANDERR	2L	// Flag für Command Error aufgetreten (nicht verw.)
	PG_FLAG_POSREACHED	4L	// Flag für Position erreicht
	PG_FLAG_INDEX_HIT	8L	// Flag für Index erkannt
	PG_FLAG_WRAP_OCC	16L	// Flag für Wrap-around aufgetreten (nicht verw.)
	PG_FLAG_POS_ERR	32L	// Flag für Positionsfehler aufgetreten
	PG_FLAG_BRKPT_RCHD	64L	// Flag für Breakpoint erreicht (nicht verw.)
	PG_FLAG_FLOATING	128L	// Flag für MOTOR OFF
	PG_FLAG_MOVING	1L << 8	// Flag für Achse fährt
	PG_FLAG_OVERFLOWS	2L << 8	// Flag für Überlauf der Slave-Position
	PG_FLAG_OVERFLOWM	4L << 8	// Flag für Überlauf der Master-Position
	PG_FLAG_POSFLOAT	8L << 8	// Flag für Lageregelung temporär abgeschaltet
	PG_FLAG_INTERNTST	64L << 8	// Flag zum internen Gebrauch
	PG_FLAG_SYNCREADY	1L << 24	// Flag für Synchronisation fertig
	PG_FLAG_SYNCFAULT	2L << 24	// Flag für Synchronisation nicht erfolgt
	PG_FLAG_SYNCACCUR	4L << 24	// Flag für geforderte Genauigkeit erreicht (Sync)
	PG_FLAG_SYNCMMHIT	8L << 24	// Flag für Master-Marker erkannt
	PG_FLAG_SYNCSMHIT	16L << 24	// Flag für Slave-Marker erkannt
	PG_FLAG_SYNCMMERR	32L << 24	// Flag für Markerabstand Master überschritten
	PG_FLAG_SYNCSMERR	64L << 24	// Flag für Markerabstand Slave überschritten
	PG_FLAG_TESTFLAG	128L << 24	// Flag zum internen Gebrauch

Index		Beschreibung Achsprozessdaten / CAM-Profil
4220	CINDEX	Aktueller Index in der Kurven-Interpolationsumgebung (Nummer des aktuellen Interpolationspunktes 0..Intno-1)
4221	CVINDEX	Aktuell benutzter Index. Der Wert ist gleich CINDEX, wenn weder Start noch Stopp aktiv sind. In diesem Fall zeigt CVINDEX auf die Start- oder Stopp-Kurvendaten.
4222	CMAXINDEX	Maximal erlaubter Index (Intno – 1)
4223	CIMPS	Kurvenposition innerhalb des aktuellen Interpolations-Intervalls (ganzzahliger Teil des 64-Bit-Wertes).
4224	CMILEN	Länge des Interpolations-Intervalls in MU Einheiten (ganzzahliger Teil des 64-Bit-Wertes).
4225	CWRAP	Aktueller Kurvenzähler (0 .. CCOUNTER), wird nach jedem Re-Start von SYNCC zurückgesetzt.
4226	CSSTART	Offset für den Start des Slaves der aktuellen Kurve in qc (für geschlossene Kurven immer 0).
4227	CCOUNTER	Anzahl der Kurven die abgearbeitet werden sollen (letzter SYNCC Befehl).
4228	CCURVEPOS	Kurvenposition des Slaves in BE Einheiten (aktualisiert in SETCURVE und wenn SYNCCxx aktiv ist).
4229	CSLAVECPOSQ	Aktuelle Kurvenposition des Slaves in qc (relativ zu CSSTART).
4230	CMASTERCPOS	Aktuelle Kurvenposition des Masters in MU Einheiten. (Wie in SETCURVE initialisiert und nach SYNCCxx aktualisiert.) Siehe auch CURVEPOS.
4231	GETCMDVEL	Liest den Befehl VEL (intgr part * 128) mit Vorzeichen (siehe auch 4186).
4240	PFG_G_STARTKORR	Enthält den ersten Korrekturwert nach Start von SYNCM. Dieser gibt an, wie viele Markerfehler vom Startvorgang kompensiert werden müssen. (Wird gefiltert falls SYNCOFFTTIME und Par. 33-29 *Filterzeit für Markerkorrektur* (SYNCMFTIME) gesetzt sind).

Index		Beschreibung Achsprozessdaten / CAM-Profil
4241	PFG_G_START KORRREST	Enthält den Rest des Startkorrekturwerts, der noch abgearbeitet werden muss. Skaliert mit PFG_G_SCALESHIFT.
4242	PFG_G_KORRFILT	Enthält den gefilterten Korrekturwert skaliert mit PFG_G_SCALESHIFT.
4243	PFG_G_LASTMMDIST	Enthält den zuletzt gemessenen Abstand zwischen 2 Master-Markern (qc Master).
4244	PFG_G_MMARK CORR	Enthält den Getriebe-Korrekturfaktor der berechnet wurde, skaliert mit PFG_G_SCALESHIFT.
4245	PFG_G_KORRUNFILT	Enthält den letzten ungefilterten Korrekturwert (qc - slave).
4246	PFG_G_MDISTMARK	Enthält den Abstand der aktuellen Master-Position vom letzten Master-Marker in % des nominellen Markerabstands.
4247	PFG_G_SDISTMARK	Enthält den Abstand der aktuellen Slave-Position vom letzten Slave-Marker in % des nominellen Markerabstands.
4248	PFG_G_START KORRVAL	STARTKORRVAL ist der Wert, um den bei jeder Markerkorrektur der Start-Korrekturwert abgebaut wird.
4249	PFG_G_LASTSM DIST	Zuletzt gemessener Abstand zwischen zwei Slave-Marker in qc - Slave.
4250	PFG_G_MARKER FILTER	Tau für den PT-Filter um den mittleren Markerabstand zu berechnen.
4251	PFG_G_KORRTAU	Tau für den PT-Filter für die Berechnung von PFG_G_KORREKTUR.
4252	PFG_G_INTMM ERROR	Summe aller Markerabstandsfehler (aktueller – mittlerer Abstand).
4253	PFG_G_MMARKERR	Gefilterte Summe aller Fehler (skalierter Wert).
4275	PFG_G_JSTATE	Enthält den Status des Bewegungsprofils der Ruckbegrenzung; das kann sein: PGS_JRK_ STOPPED 0 // gestoppt PGS_JRK_VEL_CONST 1 // konstante Geschwindigkeit (d.h. Zielgeschwindigkeit) PGS_JRK_ACC_UP 2 // Beschleunigung (Ramp-up zur Zielgeschwindigkeit) PGS_JRK_ACC_CONST 3 // konstante Beschleunigung (gegen Zielgeschwindigkeit) PGS_JRK_ACC_DOWN 4 // Beschleunigung (Ramp-down zur Zielgeschwindigkeit) PGS_JRK_ACC_DOWN_HIGH 5 // Beschleunigung (Ramp-down zur max. Beschleunigung PGS_JRK_DEC_UP 6 // Verzögerung (Ramp-up zur Zielgeschwindigkeit) PGS_JRK_DEC_CONST 7 // konstante Verzögerung (gegen Zielgeschwindigkeit) PGS_JRK_DEC_DOWN 8 // Verzögerung (Ramp-down zur Zielgeschwindigkeit) PGS_JRK_DEC_DOWN_HIGH 9 // Verzögerung (Ramp-down zur max. Verzögerung) PGS_JRK_DEC_UP_CROSS 10 // Verzögerung (Ramp-up wird Geschwindigkeit 0 kreuzen) PGS_JRK_DEC_CONST_CROSS 11 // konstante Verzögerung (wird Geschwindigkeit 0 kreuzen) PGS_JRK_DEC_DOWN_CROSS 12 // Verzögerung (Ramp-down wird Geschwindigkeit 0 kreuzen)
4276	PFG_G_VCMDSIGNED	Vorzeichenbehaftete Sollgeschwindigkeit [qc/st], identisch wie SYSVAR[4186], aber mit Vorzeichen.

Index		Beschreibung Achsprozessdaten / CAM-Profil
4277	PFG_G_JERKSTOPPATH	Liefert die Länge des Bremswegs unter der Annahme, dass ein MOTOR STOP mit aktueller Beschleunigung und Geschwindigkeit durchgeführt würde. Für diese Berechnung werden die gerade aktiven Werte für VEL, ACC, DEC und JERKMIN benutzt. Dieser Bremsweg variiert natürlich, wenn zum Beispiel die Geschwindigkeit oder JERKMIN oder irgendein anderer Parameter verändert wird. Diese Berechnung kann auch benutzt werden, wenn gerade ein anderer Rampentyp als '2' benutzt wird. In diesem Fall wird der Wert so berechnet als wäre RAMPTYPE 2 aktiv. Bitte beachten Sie, dass der Wert sich ändern kann, wenn das System in eine Überschwingsituation gerät. Das bedeutet, wenn ein POSA Befehl mit einer Position ausgeführt wurde, die nicht erreicht werden kann, wird der Bremsweg so berechnet, als müsste auf die Geschwindigkeit 0 abgebremst werden. Aber sobald die endgültige Position passiert ist, wird das Profil erneut berechnet, ohne dass man anhalten muss.

□ TESTSETP

Kurzinfo	Aufzeichnungsdaten für Testfahrt festlegen.
Syntax	TESTSETP ms wi1 wi2 wi3 arrayname
Parameter	ms = Abstand in Millisekunden zwischen zwei Messungen wi 1…3 = Indizes der drei Werte, die aufgezeichnet werden sollen. Es gelten die Vereinbarungen für das Systemarray. Es werden immer drei Werte aufgezeichnet. arrayname = Name des Arrays, das für die Aufzeichnung verwendet wird.
Array-Format	Innerhalb des Arrays werden die Werte wie folgt gespeichert (alle Werte 4 Byte):

Bezeichnung	Inhalt	Bedeutung
Version	000	Version der Datenstruktur
Achsnummer	0..	Gibt an für welche Achse die Daten gemessen wurden
ms	1	Abstand zwischen zwei Messungen in ms
wi1	i	Wert, der an Stelle 1 aufgezeichnet wurde (Index)
wi2	i	Wert, der an Stelle 2 aufgezeichnet wurde (Index)
wi3	i	Wert, der an Stelle 3 aufgezeichnet wurde (Index)
Anzahl	nn	Gibt an wie viele Messungen folgen
Daten	…	Messdaten
…	…	(insgesamt nn*3)
Anzahl	0-mm	Anzahl Messungen (falls weitere vorliegen)
Daten	…	(siehe oben)

Bitte achten Sie darauf, dass die Größe des Arrays für die Aufzeichnung ausreicht. Für den Header benötigen Sie 6 Elemente, für die Anzahl 1 Element und für jede Messung 3 Elemente. Bei 100 Messungen benötigen Sie also 307 Elemente.

Beschreibung

Standardgemäß können Sie aus dem APOSS Menü eine → *Testfahrt* auslösen, die Soll- und Istposition, Geschwindigkeit, Beschleunigung und Strom aufzeichnet und deren Ergebnis Sie in der Testfahrt-Grafik sehen können.

Mit den beiden Befehlen TESTSETP und TESTSTART können Sie darüber hinaus weitere oder andere Parameter, zum Beispiel die Master-Position aufzeichnen. Und im Gegensatz zu Testfahrt können Sie diese Daten während der Ausführung eines Programms aufzeichnen.

Mit TESTSETP legen Sie die Parameter der Aufzeichnung fest (welche Parameter wie oft aufgezeichnet werden sollen und in welches Array) und mit TESTSTART starten Sie dann die Aufzeichnung.

Befehlsgruppe I/O

Querverweis TESTSTART, DIM, SYSVAR, *Testfahrt → Aufzeichnung anzeigen*

Syntax-Beispiel

```
DIM tstfahrtarray[307]          // Array mit 307 Elementen
TESTSETP 3 0X1001 0X1009 0X1005 tstfahrtarray
   // aktuelle Slave-Position, Istposition Master und
   // aktuellen Schleppabstand aufzeichnen
... Positionierfahrt starten ...
TESTSTART 100                   // Aufzeichnung starten
```

□ TESTSTART

Kurzinfo	Aufzeichnung der Testfahrt starten.
Syntax	TESTSTART anz
Parameter	anz = Anzahl der durchzuführenden Messungen
	Sollte in einem Array nicht genügend Platz für 'anz' Messungen sein, wird der Fehler 171 „Array zu klein” ausgelöst.
Beschreibung	Mit diesem Befehl starten Sie die Aufzeichnung einer Testfahrt mit den in TESTSETP definierten Inhalten. Die aufgezeichneten Daten können Sie sich dann auch – soweit sinnvoll – mit *Testfahrt → Aufzeichnung anzeigen* grafisch darstellen lassen. Dafür stehen die vier Grafiken Position, Geschwindigkeit, Beschleunigung und Strom zur Verfügung.
Befehlsgruppe	I/O
Querverweis	TESTSETP, Testfahrt
Syntax-Beispiel	SYNCP // Synchronisieren der Position WAITI 1 ON // Wenn Taste gedrückt wird TESTSTART 200 // Aufzeichnung starten (200 Messungen)

Syntax-Beispiel	NOWAIT ON // nicht warten bis Position erreicht ist
	VEL 50
	POSA 100000 // Positionierung mit Geschwindigkeit 50% starten
	WHILE (APOS<50000) DO // Warten bis Position 50000 erreicht ist
	ENDWHILE
	VEL 100 // Geschwindigkeit auf 100% erhöhen
	TESTSTART 200 // Aufzeichnung starten (200 Messungen)
	DELAY 20 // 20 ms warten
	POSA 100000 // Positionierung mit neuer Geschwindigkeit starten
	NOWAIT OFF // Warten bis Positionierung zu Ende

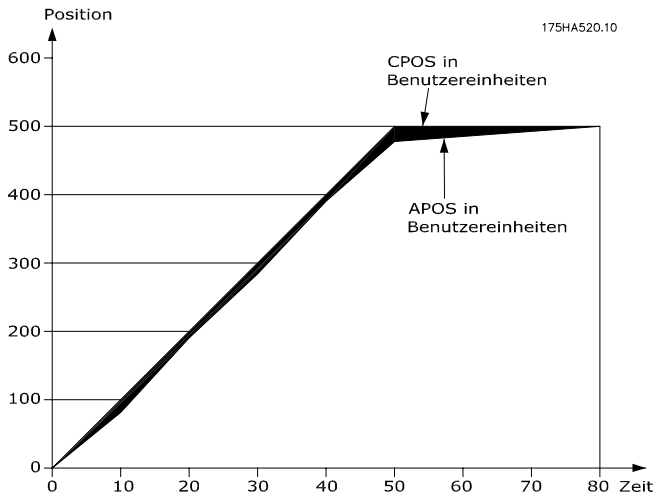
□ TIME

Kurzinfo	Systemzeit auslesen.
Syntax	erg = TIME
Rückgabewert	erg = Systemzeit in Millisekunden seit Einschalten
	ACHTUNG!: Bitte beachten Sie, dass der Wert, wenn er MLONG erreicht hat, auf –MLONG wechselt.
Beschreibung	Mit dem TIME Befehl kann die interne Systemzeit ausgelesen werden. Der TIME Befehl eignet sich vor allem, um die Ausführungszeit einer Befehlssequenz oder Maschinenzykluszeiten zu berechnen.
Befehlsgruppe	I/O
Syntax-Beispiel	PRINT TIME /* aktuelle Systemzeit ausgeben */ timestop1 = TIME /* aktuelle Systemzeit zwischenspeichern */
Programmbeispiel	ACC_01.M, DELAY_01.M, EXIT_01.M, GOSUB_01.M

☐ TRACKERR

Kurzinfo	Aktuellen Schleppabstand einer Achse abfragen.
Syntax	erg = TRACKERR
Rückgabewert	erg = aktueller Schleppabstand der Achse n in BE
Beschreibung	Fragt die Differenz zwischen Sollwert (CPOS und Istposition (APOS, ab, d.h., es werden die auftretenden Fehler betrachtet. Es ist zu beachten, dass die Istposition nicht notwendigerweise die gleiche sein muss wie der Sollwert, und dass diese nicht automatisch kompensiert wird.
Befehlsgruppe	I/O
Querverweise	APOS, CPOS, Par. 32-67 *max. tolerierter Positionsfehler*
Syntax-Beispiel	PRINT TRACKERR /* aktuellen Schleppabstand von Achse 1 abfragen */

Beispiel	POSA 500
	WHILE(1) DO
	{
	PRINT "Sollposition", CPOS
	PRINT "Istposition", APOS
	PRINT "Fehler", TRACKERR
	WAITT 10
	}
	ENDWHILE
	Ausgabe:
	Sollposition 100
	Istposition 98
	Fehler 2
	Sollposition 200
	Istposition 199
	Fehler 1
	Sollposition 300
	Istposition 297
	Fehler 3 …
	Sollposition 500
	Istposition 500
	Fehler 0
	… und so weiter
	Der dunkel markierte Bereich zwischen CPOS und APOS im Zeitintervall kann so mit TRACKERR ausgelesen werden. Um alle Fehler während der gesamten Positionierung auszulesen, sollte TRACKERR in einer Schleife ständig die Fehler verfolgen.



□ VEL

Kurzinfo	Geschwindigkeit für relative und absolute Bewegungen setzen.
Syntax	VEL v $\text{Sollgeschwindigkeit [U/Min]} = V * \frac{\text{Par. 32 - 80 Maximalgeschwindigkeit}}{\text{Par. 32 – 83 Geschwindigkeitsteiler}}$
Parameter	v = Normierter Geschwindigkeitswert
Beschreibung	Mit dem VEL Befehl wird die Geschwindigkeit für die nächsten absoluten und relativen Positioniervorgänge und die maximal zulässige Geschwindigkeit für Synchronisationsvorgänge bestimmt. Der Wert bleibt solange gültig, bis mit einem weiteren VEL Befehl eine neue Geschwindigkeit gesetzt wird. Der zu übergebende Geschwindigkeitswert wird zu den Parametern 32-80 *Maximalgeschwindigkeit* und 32-83 *Geschwindigkeitsteiler* in Bezug gesetzt. Wenn der übergebene Geschwindigkeitswert gleich dem *Geschwindigkeitsteiler* ist, wird mit der in Parameter *Maximalgeschwindigkeit* festgelegten Drehzahl gefahren. ANMERKUNG: Die Geschwindigkeit des Slaves wird im Synchronisationsmodus auch durch den Befehl VEL begrenzt.  **ACHTUNG!:** Wurde vor einem Positionier- oder Synchronisierbefehl noch keine Geschwindigkeit definiert, wird mit dem in Par. 32-84 *Default-Geschwindigkeit* festgelegten Wert gefahren. Soll während des Positioniervorgangs die Geschwindigkeit geändert werden, ist dies bei NOWAIT ON möglich, wenn dem VEL Befehl noch einmal ein POSA auf die gewünschte Zielposition folgt. Die maximal zulässige Geschwindigkeit kann jederzeit mit dem Befehl VEL geändert werden, wenn dem Befehl VEL nochmals ein SYNCV, SYNCP oder SYNCM folgt.
Befehlsgruppe	REL, ABS
Querverweise	ACC, POSA, POSR, NOWAIT Parameter: 32-80 *Maximalgeschwindigkeit*
Syntax-Beispiel	VEL 100 /* Geschwindigkeit 100 */
Programmbeispiel	VEL_01.M

□ VLTALARMSTAT

Kurzinfo	Gibt an, ob ein Alarm vorliegt oder nicht.
Syntax	VLTALARMSTAT
Beschreibung	Der Befehl VLTALARMSTAT gibt an, ob ein oder mehrere Alarmmeldungen vorhanden sind. Dabei gibt es zwei Möglichkeiten: 1<<3 oder 1<<6 abhängig davon, ob der Motor mit einem Reset wieder gestartet werden muss (Trip) oder nicht. Bit 3 = Alarm (oder mehrere) vorhanden Bit 6 = Trip Lock Alarm (oder mehrere) vorhanden
Befehlsgruppe	CON
Querverweise	VLTERRCLR

Syntax-Beispiel	IF (VLTALARMSTAT) THEN
	PRINT " Alarm aktiv ", VLTALARMSTAT
	VLTERRCLR
	ENDIF

☐ VLTCONTROL

Kurzinfo	Setzt das VLT Steuerwort im Status MOTOR OFF
Syntax	VLTCONTROL Wert Steuerwort
Parameter	wert
Beschreibung	VLTCONTROL kann das VLT Steuerwort (STW) im MOTOR OFF Status setzen. Dieser Befehl kann benutzt werden, um das Steuerwort auf einen beliebigen Wert zu setzen. Der Anwender ist verantwortlich für den richtigen Wert (besonders beim Bit DATA VALID). Die Befehle verhalten sich wie folgt: Wenn VLTCONTROL zum ersten Mal benutzt wird, wird in den vom Anwender gesteuerten Modus gewechselt. In diesem Modus wird das Steuerwort überhaupt nicht beeinflusst. Der Anwender ist verantwortlich für das Steuerwort. So lange das System in diesem Modus ist, beeinflusst OUTAN nur den Sollwert und nicht das Steuerwort. Es kann dann nur mit einem MOTOR ON Befehl oder durch Starten eines neuen APOSS Programms in den normalen Modus zurückgekehrt werden.
Befehlsgruppe	CON
Querverweise	MOTOR OFF, MOTOR ON, OUTAN
Syntax-Beispiel	MOTOR OFF … VLTCONTROL 0x047C // Antrieb einschalten OUTAN 0x1000 … MOTOR ON // Anwendersteuerung des Steuerworts abschalten

☐ VLTERRCLR

Kurzinfo	Löscht einen VLT-Alarm.
Syntax	VLTERRCLR
Beschreibung	Der Befehl VLTERRCLR löscht einen VLT-Alarm ohne Auswirkung auf einen vorhandenen Fehler der Optionskarte. Dieser Befehl kann an jeder Stelle im APOSS Programm benutzt werden.
Befehlsgruppe	CON
Querverweise	VLTALARMSTAT

Syntax-Beispiel	IF (VLTALARMSTAT) THEN
	PRINT " Alarm aktiv ", VLTALARMSTAT
	VLTERRCLR
	ENDIF

☐ WAITAX

Kurzinfo	Warten bis Zielposition erreicht ist.
Syntax	WAITAX
Beschreibung	Der WAITAX Befehl ist für die Verwendung bei aktivem NOWAIT Modus vorgesehen. Damit wird im NOWAIT ON der Zustand erreicht, dass nach einem Positionierbefehl mit der weiteren Abarbeitung des Programms gewartet wird, bis die abgefragte Achse ihre Sollposition erreicht hat.
Befehlsgruppe	CON
Querverweise	NOWAIT ON/OFF, POSA, POSR, AXEND, STAT, WAITI
Syntax-Beispiel	WAITAX /* Warten bis die Achse die Bewegung beendet hat */ WAIT AX /* Alternative Schreibweise */
Programmbeispiel	WAIT_01.M, VEL_01.M

☐ WAITI

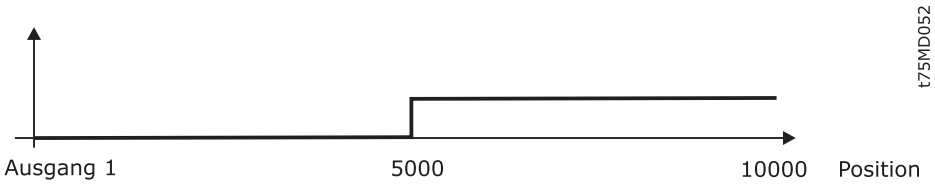
Kurzinfo	Warten auf bestimmten Eingangszustand.
Syntax	WAITI n s
Parameter	n = Nummer des Eingangs 1 – 8 oder 16 – 33 s = erwarteter Zustand: ON = High-Signal anliegend OFF = Low-Signal anliegend
Beschreibung	Der WAITI Befehl wartet mit der weiteren Programmausführung, bis der entsprechende Eingang den gewünschten Signalzustand aufweist.  **ACHTUNG!:** Wenn der erwartete Eingangszustand nie auftritt, bleibt das Programm an diesem Befehl „hängen”. Für das sichere Erkennen eines Signalzustandes ist eine minimale Signallänge notwendig! Informieren Sie sich im FC 300 Produkthandbuch und FC 300 Projektierungshandbuch über die Beschaltung und technischen Daten der Eingänge.
Befehlsgruppe	CON
Querverweise	ON INT .. GOSUB, DELAY, WAITT, WAITAX
Syntax-Beispiel	WAITI 4 ON /* Warten bis an Eingang 4 High-Pegel anliegt */ WAITI 4 1 /* 3 alternative Schreibweisen */ WAIT I 4 ON WAIT I 4 1
Syntax-Beispiel	WAITI 6 OFF /* Warten bis an Eingang 6 Low-Pegel anliegt */ WAITI 6 0 /* 3 alternative Schreibweisen */ WAIT I 6 OFF WAIT I 6 0
Programmbeispiel	WAIT_01.M

☐ WAITNDX

Kurzinfo	Warten bis die nächste Indexposition erreicht ist.
Syntax	WAITNDX t
Parameter	t = Timeout (maximale Wartezeit) in ms
Beschreibung	Warten auf Index mit Überprüfung des Timeouts. Es wird solange gewartet, bis entweder der Index der Achse n gefunden oder die Zeit (Timeout) überschritten wurde.
	ACHTUNG!: Wenn die Zeit überschritten wurde, wird ein Fehler ausgelöst, der zum Beispiel mit einer ON ERROR Funktion ausgewertet werden kann.
	ACHTUNG!: Der Befehl WAITNDX kann bei Einsatz von Absolutgebern (siehe Par. 32-00 *Inkrementalgeber Signaltyp*) nicht verwendet werden.
Befehlsgruppe	CON
Querverweise	WAITI, WAITP, INDEX
Syntax-Beispiel	CVEL 1 CSTART WAITNDX 10000 /* Wartet max. 10 s, dass die Achse die Indexposition erreicht */ OUT 1 1

☐ WAITP

Kurzinfo	Warten bis eine bestimmte Position erreicht ist.
Syntax	WAITP p
Parameter	p = absolute Position auf die gewartet wird
Beschreibung	Der WAITP Befehl bewirkt, dass mit der weiteren Programmausführung gewartet wird, bis die Position p erreicht ist. Wenn aus der Geschwindigkeit und der Istposition hervorgeht, dass der Punkt p bereits überschritten wurde, wird der Befehl ebenfalls beendet.
	ACHTUNG!: Aktive ON INT oder ON PERIOD Befehle können die Genauigkeit und Reproduzierbarkeit beeinflussen.
Befehlsgruppe	CON
Querverweise	DELAY, WAITI, WAITAX
Syntax-Beispiel	NOWAIT ON POSA 10000 WAITP 5000 /* bis Position 5000 warten */ OUT 1 1 /* Ausgang 1 setzen */ NOWAIT OFF



☐ WAITT

Kurzinfo	Zeitverzögerung
Syntax	WAITT t
Parameter	t = Verzögerung in Millisekunden (maximal MLONG)
Beschreibung	Mit dem WAITT Befehl können Sie eine definierte Programmverzögerung erzielen. Der Übergabeparameter gibt dabei die Verzögerungszeit in Millisekunden an.
	ACHTUNG!: Wenn während der Zeitverzögerung ein Interrupt auftritt, wird nach der Abarbeitung der Interrupt-Prozedur der komplette Wartevorgang von neuem begonnen. Daher sollte im Allgemeinen statt WAITT der Befehl DELAY wegen seines konstanten Zeitverhaltens verwendet werden.
Befehlsgruppe	CON
Querverweise	DELAY, WAITI, WAITAX
Syntax-Beispiel	WAITT 5000 /* 5 Sekunden warten */ WAIT T 5000 /* Alternative Schreibweise */
Programmbeispiel	WAIT_01.M

☐ WHILE .. DO .. ENDWHILE

Kurzinfo	Bedingte Schleife mit Überprüfung des Abbruchkriteriums am Schleifenanfang (Während Bedingung erfüllt, wiederhole …)
Syntax	WHILE (Bedingung) DO ENDWHILE
Parameter	Bedingung = Abbruchkriterium
Beschreibung	Mit der WHILE **..** DO **..** ENDWHILE Konstruktion kann man den eingeschlossenen Programmbereich in Abhängigkeit von einem beliebigen Kriterium ein- oder mehrfach wiederholen. Das Schleifenkriterium setzt sich aus einer oder mehreren Vergleichsoperationen zusammen und wird stets am Schleifenanfang überprüft. Das kann bei einem negativen Ergebnis bereits bei der ersten Überprüfung dazu führen, dass die Befehle innerhalb der Schleife nicht ausgeführt werden und das Programm sofort nach der ENDWHILE Anweisung fortgesetzt wird.  **ACHTUNG!:** In Abhängigkeit von dem Schleifenkriterium kann es vorkommen, dass der Schleifeninhalt nicht abgearbeitet wird. Um eine Endlosschleife zu vermeiden, müssen die innerhalb der Schleife abgearbeiteten Befehle direkt oder indirekt Einfluss auf das Ergebnis der Abbruchüberprüfung haben.
Befehlsgruppe	CON
Querverweise	LOOP, REPEAT .. UNTIL

Syntax-Beispiel	WHILE (A != 1 AND B == 0) DO
	Befehlszeile 1
	Befehlszeile n
	ENDWHILE

Programmbeispiel	WHILE_01.M, INKEY_01.M

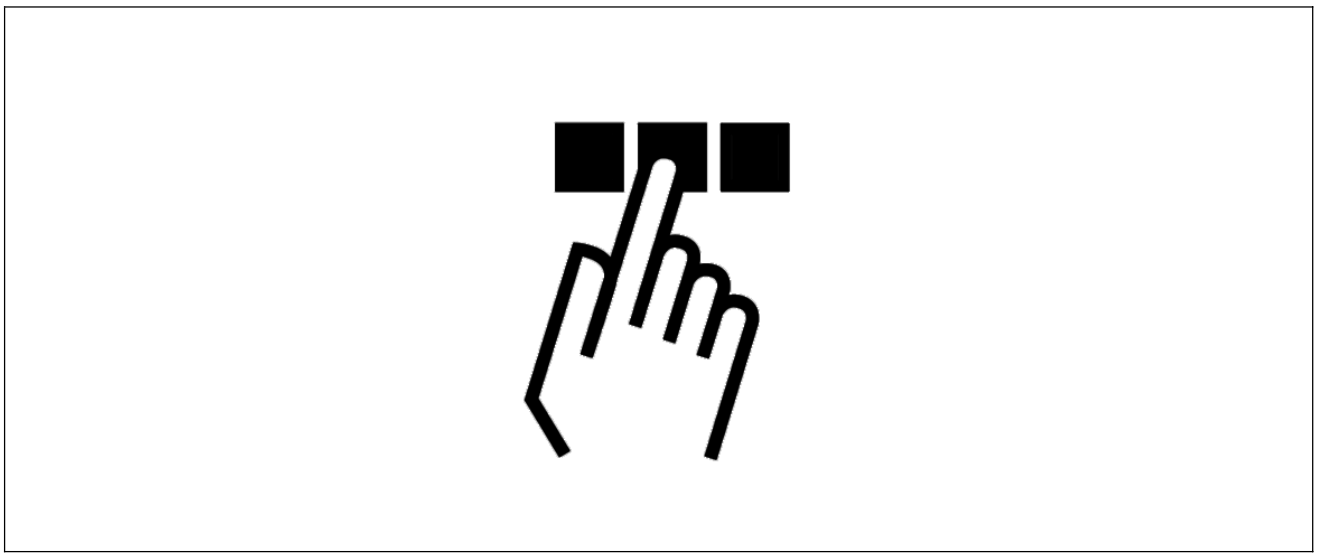
☐ _GETVEL

Kurzinfo	Abtastzeit für AVEL und MAVEL verändern.
Syntax	var = _GETVEL t Anzeige der Werte in BE/s bei AVEL bzw. qc/s bei MAVEL.
Parameter	t = Abtastrate in Millisekunden
Beschreibung	Mit dem _GETVEL Befehl können Sie die Abtastzeit für AVEL und MAVEL verändern. AVEL und MAVEL arbeiten standardgemäß mit einer Abtastzeit von 20 ms, dadurch ist die Auflösung besser. Allerdings liegt nur alle 20 ms ein neuer Wert vor. Der Befehl _GETVEL dauert genauso lange wie der zugewiesene Wert, z.B. dauert _GETVEL 200 ca. 200 ms.
Befehlsgruppe	I/0
Querverweise	AVEL, MAVEL
Syntax-Beispiel	var = _GETVEL 200 Damit wird die Messauflösung wesentlich besser, allerdings erhält man Veränderungen immer erst mit einer Verzögerung von 200 ms.

☐ #INCLUDE

Kurzinfo	Einfügen des Inhalts einer Datei an der aktuellen Progammposition.
Syntax	#INCLUDE datei
Parameter	datei = Vollständiger Name der einzufügenden Datei (Pfadangaben sind unzulässig)
Beschreibung	Die #INCLUDE Anweisung teilt dem Compiler mit, während der Übersetzung eines Programms in steuerungsspezifische Befehle an der entsprechenden Programmposition den Inhalt der angegebenen Datei einzufügen. Die #INCLUDE Anweisung ist also kein echter Befehl, der eine entsprechende Reaktion innerhalb der Steuerung auslöst, sondern eine Anweisung für das Übersetzungsprogramm, eine so genannte Compiler-Direktive. #INCLUDE kann an jeder beliebigen Programmposition und auch mehrfach innerhalb eines Programms eingesetzt werden. Es muss jedoch darauf geachtet werden, dass die in der einzubindenden Datei enthaltenen Befehle auch an der aktuellen Programmposition verwendet werden dürfen und dass der Befehlsaufbau korrekt ist. Die #INCLUDE Anweisung eignet sich vor allem, um häufig benötigte Unterprogramme in separaten Dateien abzuspeichern und innerhalb des Bereichs SUBMAINPROG **..** ENDPROG in die Anwendung zu integrieren.
	ACHTUNG!: Die einzubindende Datei muss sich im aktuellen Verzeichnis befinden. Der angegebene Dateiname muss die Endung „.m" haben. Die innerhalb der einzubindenden Datei enthaltenen Befehle müssen eine korrekte Syntax aufweisen.
Befehlsgruppe	CON
Syntax-Beispiel	#INCLUDE INC_UP01.M /* Inhalt von Datei INC_UP01.M einfügen */
Programmbeispiel	INCL_01.M + INCSTA01.M + INCPOS01.M + INCIN01.M

Parameter-Referenz



□ FC 300, MCO 305 und Anwendungsparameter

Grundsätzlich gibt es diese drei Hauptparametertypen: FC 300 Parameter, MCO 305 Parameter und Anwendungsparameter (Gruppe 19-**):

– **FC 300 und MCO 305 Parameter**

Die Parameter, die den Frequenzumrichter betreffen, sind im FC 300 Produkthandbuch beschrieben. Der folgende Abschnitt beschreibt alle Parameter, die notwendig oder hilfreich sind, wenn die MCO 305 Option eingesetzt wird.

Werkseinstellungen und Reset

Alle Parameter haben ab Werk eine Default-Einstellung, die durch eine manuelle Initialisierung des FC 300 oder mit Hilfe der Parametersatz-Kopie (Par. 0-51) zurückgesetzt werden kann. (Weitere Details finden Sie im FC 300 Produkthandbuch.)

Die Parameter können auch im Menü *Steuerung* → *Reset* → *Parameter* oder → *Vollständig* auf die Werkseinstellungen zurückgesetzt werden. Das Löschen des gesamten Speichers im Menü *Steuerung* → *Speicher* → *EEPROM löschen* setzt ebenfalls die Parameter auf die Werkseinstellungen zurück.

– **Anwendungsparameter**

Die Anwendungsparameter 19-00 to 19-99 werden im APOSS-Programm mit dem Befehl LINKGPAR definiert und im LCP-Display angezeigt.

Die Parameter 19-90 to 19-99 sind Nur-Lesen-Parameter, die zum Auslesen der Daten in Zeile 1 oder 2 des LCP-Displays genutzt werden können. Mit Parameter 0-2* LCP Display können Sie auswählen, welche Parameter an welcher Stelle dargestellt werden sollen.

□ Parameterzugriff

Es gibt drei Methoden, auf die Parameter zuzugreifen:

- LCP
- PC Software MCT 10
- Feldbus

∗ Standardeinstellung | **[ ] bei Kommunikation über serielle Schnittstelle benutzter Wert**

□ Initialisierung auf die Werkseinstellungen

Es gibt zwei Methoden zum Initialisieren des Frequenzumrichters auf die Werkseinstellungen:

Empfohlene Initialisierung (mit Par. 14-22):

1. Parameter 14-22 wählen.
2. [OK] drücken.
3. „Initialisierung" wählen.
4. [OK] drücken.
5. Netzversorgung trennen und warten bis das Display abschaltet.
6. Netzversorgung wieder einschalten – der Frequenzumrichter ist nun zurückgesetzt.

ACHTUNG!:
MCO 305 Programme und Array sind davon nicht betroffen.

Par. 14-22 initialisiert alles außer:	
14-50	*EMV 1*
8-30	*FC-Protokoll*
8-31	*Adresse*
8-32	*FC-Baudrate*
8-35	*FC-Antwortzeit Min.-Delay*
8-36	*FC-Antwortzeit Max.-Delay*
15-00 to 15-05	Betriebsdaten
15-20 to 15-22	Protokollierung
15-30 to 15-32	Fehlerspeicher

Manuelle Initialisierung:

1. Netzversorgung trennen und warten bis das Display abschaltet.
2. Gleichzeitig [Status] + [Main Menu] + [OK]-Tasten drücken.
3. Netzversorgung wieder einschalten und dabei die Tasten weiterhin gedrückt halten.
4. Nach ca. 5 s die Tasten loslassen.
5. Der Frequenzumrichter ist nun gemäß den Werkseinstellungen programmiert.

Diese Methode initialisiert alles außer:	
15-00	*Betriebsstunden*
15-03	*Anzahl Netz-Ein*
15-04	*Anzahl Übertemperaturen*
15-05	*Anzahl Überspannungen*

ACHTUNG!:
Bei einer manuellen Initialisierung werden auch die Einstellungen der seriellen Kommunikation und der Fehlerspeicher zurückgesetzt.
Und alle MCO 305 Programme und Arrays werden gelöscht!

□ Parameter lesen und schreiben

Im Anwendungsprogramm gibt es eine Lesezugriff auf alle FC 300 Parameter inklusive MCO 305 Parameter und Anwendungsparameter (Gruppe 19-**). Es gibt zwei Befehle, um Parameter zu lesen:

- GET wird benutzt, um alle MCO 305 betreffenden Parameter zu lesen, das sind die Gruppen 19-**, 32-**, 33-** und 34-**.
- GETVLT wird benutzt, um alle anderen FC 300 Parameter zu lesen.

Es gibt auch einen Schreibzugriff auf FC 300 Parameter, aber mit einigen Einschränkungen: Die Parametergruppen 16-** und 34-** sind Nur-Lesen-Parameter und können daher nicht geändert werden. Einige der FC 300 Parameter können nur geändert werden, wenn der Antrieb angehalten wird und können daher nicht während des Betriebs geändert werden. Die vollständige Beschreibung aller Parameter finden Sie im FC 300 Produkthandbuch.

Es gibt zwei Befehle, um Parameter zu schreiben:

- SET wird benutzt, um alle MCO 305 betreffenden Parameter zu schreiben, das sind die Gruppen 19-**, 32-** und 33-**.
- SETVLT wird benutzt, um alle anderen FC 300 Parameter zu schreiben.

Überblick

	Parameter	Befehl	Beispiel
Lesen	32-**, 33-** und 34**	GET name	var = GET ENCODER
	19-**	GET nummer	var = GET 1900
	Alle anderen FC 300 Parameter	GETVLT nummer	var = GETVLT 1610
Schreiben	32-** und 33-**	SET name	SET ENCODER 1024
	19-**	SET nummer	SET 1900 555
	Alle anderen FC 300 Parameter	SETVLT nummer	SETVLT 303 1500

ACHTUNG!:
Parameter, die mit SET oder SETVLT geändert wurden, werden nicht im RAM gespeichert und gehen daher beim Ausschalten verloren. Ausnahme: Die Anwendungsparameter (Gruppe 19-**) werden automatisch beim Ausschalten gespeichert. Die anderen MCO 305 Parameter (Gruppe 32-** und 33-**) können mit dem Befehl SAVE AXPARS gespeichert werden.

□ Parameter ändern und speichern

Parameter, die über das LCP oder mittels *Steuerung* → *Parameter* → *Achsen* geändert werden, werden in das EEPROM gespeichert und bleiben auch nach dem Stromabschalten erhalten.

Parameter, die durch das APOSS Anwendungsprogramm mit dem Befehl SETVLT geändert werden, werden nur im RAM gespeichert und sind daher nach dem Stromabschalten verloren.

Parameter, die durch das APOSS Anwendungsprogramm mit dem Befehl SET geändert werden, sind nur aktiv während das Anwendungsprogramm läuft. Diese Parameter können mit dem Befehl SAVEPROM oder durch Drücken der [OK]-Taste am FC 300 Display in das EEPROM gespeichert werden und bleiben dann auch nach dem Stromabschalten erhalten.

ACHTUNG!:
Bitte beachten Sie, dass ein EEPROM eine begrenzte Lebenszeit hat; es kann aber ungefähr 10000-mal programmiert werden.

□ Übersicht FC 300 Parameter

Wenn eine MCO 305 Option installiert ist, werden neue Parameter (Gruppe 19-**, 32-**, 33-** und 34-**) ergänzt und zusätzlich einige vorhandene FC 300 Parameter modifiziert; einige Parameter erhalten weitere Auswahlmöglichkeiten und einige bekommen andere Standardwerte. Im Folgenden finden Sie eine Übersicht der Parameter, die dies betrifft:

Par. Nummer	Neue Auswahlmöglichkeiten	Neue Standardwerte
0-20 0-21 0-22 0-23 0-24	[1990] - [1999] [3400] - [3410] [3421] - [3430] [3440] - [3441] [3450] - [3462]	-
1-02	[4] MCO Drehgeber 1 [5] MCO Drehgeber 2	-
1-62	-	0%
3-15 3-16	-	[0] Keine Funktion

∗ Standardeinstellung	[ ] bei Kommunikation über serielle Schnittstelle benutzter Wert

Par. Nummer	Neue Auswahlmöglichkeiten	Neue Standardwerte
3-17		
3-41 3-42 3-51 3-52 3-61 3-62 3-71 3-72	-	0,01 s
4-10	-	„Beide Richtungen”
5-10 5-11 5-12 5-13	-	[0] Ohne Funktion
5-30 5-31 5-32 5-33	[51] MCO gesteuert	[51] MCO gesteuert
5-40	[51] MCO gesteuert	[51] MCO gesteuert
5-60 5-63	[51] MCO gesteuert	[51] MCO gesteuert
6-50	[52] MCO 0-20 mA	[52] MCO 0-20 mA
6-60	[53] MCO 4-20 mA	
7-00	[4] MCO Drehgeber 1 [5] MCO Drehgeber 2	-
8-02	[5] Option C0	[5] Option C0
9-15	[3401] - [3410]	Index [0] 3401 Index [1] 3402 Index [2] 3403 Index [3] 3404 Index [4] 3405 Index [5] 3406 Index [6] 3407 Index [7] 3408 Index [8] 3409 Index [9] 3410
9-16	[3421] - [3430]	Index [0] 3401 Index [1] 3402 Index [2] 3403 Index [3] 3404 Index [4] 3405 Index [5] 3406 Index [6] 3407 Index [7] 3408 Index [8] 3409 Index [9] 3410

ACHTUNG!:
Grundsätzlich ist es sehr wichtig, die FC 300 Parameter passend zum Motor zu optimieren – möglichst mit AMA – um ein gutes Steuerungsverhalten zu erreichen.

Der *Sollwertbereich* in Par. 3-00 muss in Übereinstimmung mit Par. 32-80 *Maximalgeschwindigkeit* gesetzt werden, bevor die Regelungsparameter optimiert werden.

∗ Standardeinstellung	[ ] bei Kommunikation über serielle Schnittstelle benutzter Wert

□ Einstellungen für die Anwendung

□ 19-** Anwendungsparameter

19-00 ...19-89 Anwendungsparameter

Bereich

-2147483648 – 2147483647

(Der tatsächliche Bereich, der im LCP-Display zu sehen ist, wird mit LINKGPAR festgelegt.)

Funktion

Die Anwendungsparameter werden benutzt, um anwendungsspezifische Daten für das Anwendungsprogramm einzugeben. Anwendungsparameter werden mit dem Befehl LINKGPAR erzeugt. Dabei ist es möglich, sowohl einen Parameternamen als auch die minimale und maximale Begrenzung der Einstellung zu definieren. Siehe auch LINKGPAR Beschreibung.

ANMERKUNG: Anwendungsparameter sind im LCP-Display nur sichtbar und erreichbar, wenn sie im Anwendungsprogramm erzeugt und definiert wurden.

Syntax-Beispiel

LINKGPAR 1901 "name" 0 100000 0
/* Verknüpfe Par. 19-01 mit LCP */

19-90 .. 19-99 Nur-Lesen Anwendungsparameter

Bereich

-2147483648 – 2147483647

(Der Bereich hängt von den Daten ab, die mit dem auszulesenden Parameter verknüpft sind.)

Funktion

Nur-Lesen Anwendungsparameter werden benutzt, um zusätzliche interne Prozessdaten und anwendungsspezifische Daten des Anwendungsprogramms auszulesen. Nur-Lesen Anwendungsparameter werden erzeugt mit:

- LINKSYSVAR Befehl für interne Prozessdaten,
- LINKGPAR Befehl für anwendungsspezifische Daten;

wobei auch der Parametername festgelegt werden kann. Siehe auch LINKSYSVAR und LINKGPAR Beschreibung.

Wie die Parameter 19-90 bis 19-99 im LCP angezeigt werden, wird in den Parametern 0-20 bis 0-24 *Display-Modus* bestimmt.

ANMERKUNG: Nur-Lesen Anwendungsparameter sind im LCP-Display nur sichtbar, wenn sie im Anwendungsprogramm erzeugt und definiert wurden.

□ MCO Parameter

Die MCO Parameter für den FC 300 sind zur einfachen Auffindung und Auswahl in verschiedenen Gruppen organisiert.

32-**	**MCO Grundeinstellungen**	
32-0*	Drehgeber 2 - Slave	Seite 187
32-3*	Drehgeber 1 - Master	Seite 188
32-5*	Rückführungsquelle	Seite 193
32-6*	PID-Regelung	Seite 194
32-8*	Geschwindigkeit & Beschleunigung	Seite 196

33-**	**MCO weitere Einstellungen**	
33-0*	Homefahrt	Seite 199
33-1*	Synchronisation	Seite 200
33-4*	Grenzwertbehandlung	Seite 209
33-5*	I/O Konfiguration	Seite 211

34-**	**MCO Datenanzeigen**	Seite 217
34-0*	PCD Schreib-Parameter	Seite 217
34-2*	PCD Lese-Parameter	Seite 217
34-4*	Eingänge & Ausgänge	Seite 218
34-5*	Prozessdaten	Seite 218

□ Allgemeine Information zu den Parameterwerten

Einige Grenzwerte sind auf Grund der besseren Lesbarkeit mit 1 Mrd. angegeben. Der exakte Wert beträgt jedoch 1.073.741.823 (= MLONG).

□ Eingabebereich

Die Überschreitung der angegebenen Eingabebereiche wird vom Programm nicht geprüft, da es wegen der großen Wertebereiche keine sinnvollen Kontrollmöglichkeiten gibt.

ACHTUNG!:
Schon innerhalb der angegebenen Bereiche kann es durch die großen Leistungsunterschiede der Motoren und den vielseitigen Anwendungsmöglichkeiten zu unsinnigen Eingaben kommen. Es liegt daher in der Verantwortung der Programmierer und Anwender, auf die zulässigen Leistungsbereiche der Antriebe und des Systems zu achten.

ACHTUNG!:
Wenn der Parameterwert außerhalb des definierten Wertebereichs ist, wird der Befehl nicht korrekt dargestellt und ausgeführt.

□ MCO Grundeinstellungen

□ 32-0* Drehgeber 2 - Slave

Folgende Parameter konfigurieren die Schnittstelle für den Drehgeber 2:

32-00 Inkrementalgeber Signaltyp

ENCODERTYPE

Option

	Keiner	[0]
*	RS422 (TTL/Leitungstreiber)	[1]
	SinCos 1Vss	[2]

Funktion

Legt den Typ des Inkrementalgebers fest, der mit der Drehgeber 2 Schnittstelle (X55) verbunden ist.

Wählen Sie *Keiner* [0] wenn kein Inkrementalgeber verbunden ist.

Wählen Sie *RS422 (TTL/Leitungstreiber)* [1] wenn ein digitaler Inkrementalgeber mit einer Schnittstelle gemäß RS422 angeschlossen ist.

Wählen Sie *SinCos 1Vss* [2] wenn ein analoger Inkrementalgeber mit einer Ausgangsspannung von 1 Vss angeschlossen ist.

32-01 Inkrementalgeber Auflösung

ENCODER

Bereich [Unit]

1 – MLONG [Pulse/U]	*	1024

Funktion

Die Drehgeberauflösung wird benutzt, um sowohl die Geschwindigkeit in U/Min (Umdrehungen pro Minute) zu berechnen als auch den Timeout für die Erkennung des Nullimpulses in Verbindung mit HOME und INDEX festzulegen.

Setzen Sie die Auflösung des Inkrementalgebers, der mit der Drehgeber 2 Schnittstelle (X55) verbunden ist:

- Digitaler Inkrementalgeber (32-00 = [1]): Die Auflösung muss in Pulsen pro Umdrehungen gesetzt werden.
- Analoger Inkrementalgeber (32-00 = [2]): Die Anzahl der sinusförmigen Perioden pro Umdrehung ergibt die Auflösung.

Die Drehgeberauflösung finden Sie auf dem Typenschild oder im Datenblatt des Drehgebers.

ANMERKUNG: Die maximale Frequenz des Drehgebersignals darf 410 kHz nicht überschreiten.

ANMERKUNG: Der Parameter wird nur angezeigt, wenn Par. 32-00 ≠ 0.

32-02 Absolutgeber Protokoll

ENCODERABSTYPE

Option

*	Keiner	[0]
	SSI	[4]
	SSI mit Filter	[5]

Funktion

Bestimmt den Typ des Absolutgebers, der an der Drehgeber 2 Schnittstelle (X55) angeschlossen ist.

Wählen Sie *Keiner* [0] wenn kein Absolutgeber angeschlossen ist.

Wählen Sie *SSI* [4] wenn ein Absolutgeber mit SSI Schnittstelle angeschlossen ist.

Wählen Sie *SSI mit Filter* [5] wenn ein Absolutgeber mit SSI Schnittstelle angeschlossen ist und die Kommunikation/das Signal instabil ist.

Ein Sprung in den Positionsdaten wird erkannt, wenn er größer als die Drehgeberauflösung/2 ist. Dieser wird mit Hilfe eines künstlichen Positionswertes korrigiert, der auf Basis der letzten Geschwindigkeit berechnet wird. Wenn der Fehler länger als 100 Datenausgaben (> 100 ms) anhält, wird nicht weiter korrigiert, was dann tatsächlich zu einem „Positionsfehler" (Fehler 108) führt.

Die gesamte Anzahl der Fehler wird in einer internen Variablen gespeichert, die mit SYSVAR[16] ausgelesen werden kann.

ACHTUNG!:
Folgende Befehle können mit Absolutgebern nicht benutzt werden: DEF ORIGIN, HOME, INDEX, IPOS und WAITNDX.

32-03 Absolutgeber Auflösung

ENCODERABSRES

Bereich

1 … MLONG	∗ 8192

Funktion

Die Drehgeberauflösung wird benutzt, um die Geschwindigkeit in U/Min zu berechnen.

Setzen Sie die Auflösung des Absolutgebers, der mit der Drehgeber 2 Schnittstelle (X55) verbunden ist, in Positionen pro Umdrehung. Sie finden die Drehgeberauflösung auf dem Typenschild oder im Datenblatt.

ANMERKUNG: Der Parameter wird nur angezeigt, wenn Par. 32-02 ≠ 0.

32-05 Absolutgeber Datenlänge

ENCODERDATLEN

Bereich [Unit]

8 – 37 [Bit]	∗ 25

Funktion

Bestimmen Sie die Anzahl der Datenbits für den angeschlossenen Absolutgeber; siehe Datenblatt des Drehgebers. Dies ist notwendig, um für MCO 305 die richtige Anzahl der Taktbits zu berechnen.

ANMERKUNG: Der Parameter wird nur angezeigt, wenn Par. 32-02 ≠ 0.

32-06 Absolutgeber Taktfrequenz

ENCODERFREQ

Bereich [Unit]

78,125 – 2.000,000 [kHz]	∗ 262,000

Funktion

Bestimmt die Frequenz des Taktsignals des Absolutgebers durch MCO 305. Setzen Sie eine geeignete Frequenz für den angeschlossenen Drehgeber.

ANMERKUNG: Der Parameter wird nur angezeigt, wenn Par. 32-02 ≠ 0.

32-07 Absolutgeber Takterzeugung

ENCODERCLOCK

Option

	Aus	[0]
∗	Ein	[1]

Funktion

Auswahl ob Drehgeber 0 ein Absolutgeber-Taktsignal erzeugen soll oder nicht.

Wählen Sie *Aus* [0] wenn mehrere MCO 305 mit dem gleichen Absolutgeber verbunden sind. Denn nur einer (MCO 305) darf das Taktsignal und nur einer (Drehgeber oder MCO 305) das Datensignal erzeugen, wenn mehrere MCO 305 miteinander verbunden sind.

Wählen Sie *Ein* [1] wenn MCO 305 nur mit einem Drehgeber verbunden ist.

ANMERKUNG: Der Parameter wird nur angezeigt, wenn Par. 32-02 ≠ 0.

32-08 Absolutgeber Kabellänge

ENCODERDELAY

Bereich [Einheit]

0 – 300 m	∗ 0

Funktion

Wenn das Kabel zu lang ist, würden die Takt- und Datensignale des Absolutgebers (SSI) außerhalb der Synchronisation sein. MCO 305 kompensiert automatisch die Verzögerung durch das Kabel, wenn die Kabellänge bekannt ist. Diese Kompensation basiert auf einer Verzögerung von ungefähr 6 ns ($6 * 10^{-9}$ Sekunden) pro Meter.

Bestimmen Sie die gesamte Kabellänge (in Meter) zwischen MCO 305 und dem Absolutgeber.

ANMERKUNG: Der Parameter wird nur angezeigt, wenn Par. 32-02 ≠ 0.

32-09 Drehgeber-Überwachung

ENCODERMONITORING

Option

*	Aus	[0]
	3 Channels	[1]
	2 Channels	[2]

Funktion

Die Überwachung von offenem Stromkreis und Kurzschluss der Drehgebereingänge kann an- und abgeschaltet werden.

Wählen Sie *Aus* [0] wenn keine Hardware-Überwachung benötigt wird.

Wählen Sie [1] wenn alle 3 Kanäle, also A, B und Index überwacht werden sollen.

Wählen Sie [2] wenn nur 2 Kanäle, A, B überwacht werden sollen.

Wenn die Hardware-Überwachung eingeschaltet ist, wird bei einem Fehler am Drehgeber ein Fehlercode (Fehler 192) ausgegeben.

32-10 Drehrichtung

POSDRCT

Option

*	Keine Aktion	[1]
	Sollwert umgedreht	[2]
	Benutzereinheiten umgedreht (–1)	[3]
	BE und Sollwert umgedreht (–2)	[4]

Funktion

Normalerweise bewirkt ein positiver Sollwert auch eine positive Änderung der Position. Falls dies nicht der Fall ist, kann der Sollwert intern umgedreht werden. Es gibt folgende Möglichkeiten:

1 = Keine Veränderung, d.h. positive Sollwerte ergeben positive Drehgeberwerte.

2 = Das Vorzeichen des Sollwertes wird intern getauscht (Plus wird Minus und umgekehrt). Dies kommt einem Umdrehen der Motorleitungen gleich, bzw. dem Vertauschen der A- und B-Spur beim Drehgeber.

3 = Das Vorzeichen der Benutzereinheit wird gedreht. Positive Sollwerte ergeben demnach positive Drehgeberwerte, die aber negativ angezeigt werden. Dies gilt für alle Ausgaben (APOS, CPOS, …), alle Benutzereingaben (POSA, POSR, …) und alle Synchronisationsfaktoren sowie Geschwindigkeiten (CVEL, Par. 33-03 *Homefahrt-Geschwindigkeit*).

4 = Wie [2], d.h. Vorzeichen des Sollwertes wird intern getauscht und zusätzlich wird das Vorzeichen der Benutzereinheit negiert.

Die Richtung der Synchronisation (Verhältnis zum Master) kann durch negativen Par. 33-10 *Synchronisationsfaktor Master* umgedreht werden.

32-11 Benutzerfaktor Nenner

POSFACT_N

Bereich

1 – MLONG	* 1

Funktion

Alle Wegangaben in Fahrbefehlen erfolgen in Benutzereinheiten [BE] und werden intern in Quadcounts umgerechnet. So ist es durch eine entsprechende Wahl dieser Normierungsgröße möglich, mit beliebigen technischen Maßangaben (zum Beispiel mm) zu arbeiten.

Der Faktor ist ein Bruch, der sich aus Zähler und Nenner zusammensetzt.

$$1\ \text{BE} = \frac{\text{Par. 32 - 12 Benutzerfaktor Zähler}}{\text{Par. 32 - 11 Benutzerfaktor Nenner}}$$

Die Normierung bestimmt, wie viele Quadcounts eine Benutzereinheit ergeben: Wenn der Faktor zum Beispiel 50375/1000 beträgt, entspricht 1 BE genau 50,375 qc.

Im CAM-Modus wird der Parameter benutzt, um die Einheit für den Slave-Antrieb festzulegen, damit man auch im CAM-Editor mit sinnvollen Einheiten arbeiten kann. Siehe Voraussetzung der Formel und Beispiel bei Par. 32-12 *Benutzerfaktor Zähler*.

$$\frac{\text{Getriebefaktor} * \text{Drehgeberauflösung} * 4}{\text{Skalierfaktor}}\ \text{qc} = 1\ \text{BE}$$

Außerdem kann man die Kurven mit diesem Faktor stauchen oder strecken, ohne jeweils neue Kurven definieren zu müssen. Die Verwendung von Zähler und Nenner für den Getriebefaktor führt zu einem sehr präzisen Ergebnis, da in fast allen Fällen Übersetzungen als Bruch darstellbar sind.

32-12 Benutzerfaktor Zähler

POSFACT_Z

Bereich

1 – MLONG/max. Position (BE) ∗ 1

Funktion

Wegangaben in Fahrbefehlen erfolgen in Benutzereinheiten und werden intern in Quadcounts umgerechnet. So ist es durch eine entsprechende Wahl dieser Normierungsgröße möglich, mit beliebigen technischen Maßangaben (zum Beispiel mm) zu arbeiten.

Der Faktor ist ein Bruch, der sich aus Zähler und Nenner zusammensetzt.

$$1\ \text{BE} = \frac{\text{Par. 32 - 12 Benutzerfaktor Zähler}}{\text{Par. 32 - 11 Benutzerfaktor Nenner}}$$

Die Normierung bestimmt, wie viele Quadcounts eine Benutzereinheit ergeben.

Im CAM-Modus wird der Parameter benutzt, um die Einheit für den Slave-Antrieb festzulegen, damit man im CAM-Editor mit sinnvollen Einheiten arbeiten kann. Siehe Beispiel 2.

$$\frac{\text{Getriebefaktor} * \text{Drehgeberauflösung} * 4}{\text{Skalierfaktor}}\ \text{qc} = 1\ \text{BE}$$

vorausgesetzt dass:

$$\text{Getriebefaktor} = \frac{\text{Motorumdrehungen}}{\text{Umdrehungen am Abtrieb}}$$

Drehgeber = Inkrementalgeber (bei Absolutgebern entfällt der Multiplikator 4)

Skalierfaktor = Anzahl der Benutzereinheiten BE [qc], die einer Umdrehung am Antrieb entsprechen.

Außerdem kann man die Kurven mit diesem Faktor stauchen oder strecken, ohne jeweils neue Kurven definieren zu müssen. Die Verwendung von Zähler und Nenner für den Getriebefaktor führt zu einem sehr präzisen Ergebnis, da in fast allen Fällen Übersetzungen als Bruch darstellbar sind.

Beispiel 1

Welle oder Spindel:

25 Motorumdrehungen ergeben 1 Spindelumdrehung; Getriebefaktor = 25/1

Drehgeber-Auflösung (Inkrementalgeber) = 500

Spindelsteigung = 1 Umdrehung der Spindel = 5 mm

Skalierfaktor, wenn mit 1/10 mm Auflösung gearbeitet werden soll = 5 * 10 = 50

$$\frac{25/1 * 500 * 4}{50}\ \text{qc} = \frac{25 * 10 * 4}{1}\ \text{qc} = \frac{1000}{1}\ \text{qc} = 1\ \text{BE}$$

Par. 32-12 *Benutzerfaktor Zähler* = 1000
Par. 32-11 *Benutzerfaktor Nenner* = 1

Beispiel 2

Walze:

Getriebefaktor = 5/1

Drehgeberauflösung (Inkrementalgeber) = 500

Eine Walzenumdrehung beträgt 360 Grad. In diesem Beispiel soll mit einer Auflösung von 1/10 Grad gearbeitet werden. Das bedeutet, dass eine Walzenumdrehung in 3600 Arbeitseinheiten eingeteilt wird:
Skalierfaktor = 3600

$$\frac{5/1 * 500 * 4}{3600}\ \text{qc} = \frac{5 * 500 * 4}{3600}\ \text{qc} = 1\ \text{BE}$$

$$= \frac{25}{9}\ \text{qc} = 1\ \text{BE} = \frac{\text{Par. 32 - 12 Benutzerfakter Zähler}}{\text{Par. 32 - 11 B}\textit{enutzerfaktor}\ \text{Nenner}}$$

Par. 32-12 *Benutzerfaktor Zähler* = 25
Par. 32-11 *Benutzerfaktor Nenner* = 9

□ 32-3* Drehgeber 1 - Master

Folgende Parameter konfigurieren die Schnittstelle für den Drehgeber 1:

32-30 Inkrementalgeber Signaltyp

MENCODERTYPE

Option

	Keine	[0]
*	RS422 (TTL/Leitungstreiber)	[1]

Funktion

Legt den Typ des Inkrementalgebers fest, der mit der Drehgeber 1 Schnittstelle (X56) verbunden ist.

Wählen Sie *Keine* [0] wenn kein Inkrementalgeber angeschlossen ist.

Wählen Sie *RS422 (TTL/Leitungstreiber)* [1] wenn ein digitaler Inkrementalgeber mit einer Schnittstelle gemäß RS422 angeschlossen ist.

32-31 Inkrementalgeber Auflösung

MENCODER

Bereich [Unit]

	1 – MLONG [Pulse/U]	* 1024

Funktion

Setzt die Auflösung des Inkrementalgebers der mit der Schnittstelle von Drehgeber 1 (X56) verbunden ist:

– Digitaler Inkrementalgeber (32-30 = [1]): Die Auflösung muss in Pulsen pro Umdrehung gesetzt werden.

Die Drehgeberauflösung finden Sie auf dem Typenschild oder im Datenblatt des Drehgebers.

ANMERKUNG: Die maximale Frequenz des Drehgebersignals darf 410 kHz nicht überschreiten.

ANMERKUNG: Der Parameter wird nur angezeigt, wenn Par. 32-30 ≠ 0.

32-32 Absolutgeber Protokoll

MENCODERABSTYPE

Option

*	Keiner	[0]
	SSI	[4]
	SSI mit Filter	[5]

Funktion

Bestimmt den Typ des Absolutgebers, der an der Schnittstelle von Drehgeber 1 (X56) angeschlossen ist.

Wählen Sie *Keiner* [0] wenn kein Absolutgeber angeschlossen ist.

Wählen Sie *SSI* [4] wenn ein Absolutgeber mit SSI Schnittstelle angeschlossen ist.

Wählen Sie *SSI mit Filter* [5] wenn ein Absolutgeber mit SSI Schnittstelle angeschlossen ist und die Kommunikation/das Signal instabil ist.

Virtueller Master: Mit dem Drehgebertyp [5] ist es möglich mit einem APOSS-Befehl einen Master zu simulieren, zum Beispiel wenn die Master-Position über den Bus gelesen wird. Die simulierten Master-Positionen werden mit der Systemvariablen SYSVAR[4105] gesetzt und gelesen.

ACHTUNG!:
Der MIPOS Befehl kann mit einem Absolutgeber nicht benutzt werden.

32-33 Absolutgeber Auflösung

MENCODERABSRES

Bereich [Unit]

	1 – MLONG [PPR]	* 8192

Funktion

ANMERKUNG: Der Parameter wird nur angezeigt wenn Par. 32-32 ≠ 0.

32-35 Absolutgeber Datenlänge

MENCODERDATLEN

Bereich [Einheit]

	8 – 37 [Bit]	* 25

Funktion

Bestimmt die Anzahl der Datenbits für den angeschlossenen Absolutgeber, siehe Datenblatt des Drehgebers. Diese Angabe braucht MCO 305, um die richtige Anzahl der Taktbits zu erzeugen.

ANMERKUNG: Der Parameter wird nur angezeigt wenn Par. 32-32 ≠ 0.

32-36 Absolutgeber Taktfrequenz

MENCODERFREQ

Bereich [Unit]

78.125 – 2000.000 [kHz] ∗ 262.000

Funktion

Bestimmt die Frequenz des Taktsignals des Absolutgebers durch MCO 305. Setzen Sie eine geeignete Frequenz für den angeschlossenen Drehgeber.

ANMERKUNG: Der Parameter wird nur angezeigt, wenn Par. 32-32 ≠ 0.

32-37 Absolutgeber Takterzeugung

MENCODERCLOCK

Option

	Aus	[0]
∗	Ein	[1]

Funktion

Auswahl ob Drehgeber 0 ein Absolutgeber-Taktsignal erzeugen soll oder nicht.

Wählen Sie *Aus* [0] wenn mehrere MCO 305 mit dem gleichen Absolutgeber verbunden sind oder wenn eine MCO 305 mit einer anderen MCO 305 verbunden ist, an der ein absoluter virtueller Master aktiv ist. Denn nur einer (MCO 305) darf das Taktsignal und nur einer (Drehgeber oder MCO 305) das Datensignal erzeugen, wenn mehrere MCO 305 miteinander verbunden sind.

Wählen Sie *Ein* [1] wenn eine MCO 305 nur mit einem Absolutgeber verbunden ist.

ANMERKUNG: Der Parameter wird nur angezeigt, wenn Par. 32-32 ≠ 0.

32-38 Absolutgeber Kabellänge

MENCODERDELAY

Option

0 – 300 m ∗ 0

Funktion

Wenn das Kabel zu lang ist, würden die Takt- und Datensignale des Absolutgebers (SSI) außerhalb der Synchronisation liegen. MCO 305 kompensiert automatisch die Verzögerung durch das Kabel, wenn die Kabellänge bekannt ist. Diese Kompensation basiert auf eine Verzögerung von ungefähr 6 ns ($6 * 10^{-9}$ Sekunden) pro Meter.

Bestimmen Sie die gesamte Kabellänge (in Meter) zwischen MCO 305 und dem Absolutgeber.

ANMERKUNG: Der Parameter wird nur angezeigt, wenn Par. 32-32 ≠ 0.

32-39 Drehgeber-Überwachung

MENCODERMONITORING

Option

∗	Aus	[0]
	3 Channels	[1]
	2 Channels	[2]

Funktion

Die Überwachung von offenem Stromkreis und Kurzschluss der Drehgebereingänge kann an- oder abgeschaltet werden.

Wählen Sie *Aus* [0] wenn Sie keine Hardware-Überwachung benötigen.

Wählen Sie [1] wenn alle 3 Kanäle, also A, B und Index überwacht werden sollen.

Wählen Sie [2] wenn nur 2 Kanäle, A, B überwacht werden sollen.

Wenn die Hardware-Überwachung eingeschaltet ist, wird bei einem Fehler am Drehgeber eine Fehlermeldung (Fehler 192) ausgegeben.

32-40 Drehgeber Abschlusswiderstand

MENCODERTERM

Option

	Aus	[0]
*	Ein	[1]

Funktion

Die Abschlusswiderstände können für den Drehgeber 1 an- und abgeschaltet werden.

Wählen Sie *Aus* [0] wenn eine hohe Eingangs-Impedanz erforderlich ist:

- Ein Drehgeber ist mit mehreren MCO 305 verbunden.
- Der virtuelle Master-Ausgang einer MCO 305 ist mit mehreren MCO 305 verbunden.

Wählen Sie *Ein* [1] wenn der Drehgeber nur mit dieser einen MCO 305 verbunden ist.

Das Verdrahtungsdiagramm finden Sie im MCO 305 Produkthandbuch.

□ 32-5* Rückführungsquelle

32-50 Rückführung Slave

Option

*	Enc 2	[2]
	Motor Steuerung	[3]

Funktion

Wählen Sie die Rückführungsquelle für MCO:

Wählen Sie [2] für Enc2. Wenn das Motorsteuerprinzip "*Flux mit Motorrückführung*" (Par. 1-01) benutzt wird, ist es möglich die Flux Rückführungsquelle (Par. 1-02) für MCO 305 zu benutzen.

Wählen Sie [3] für eine MCO 305 Rückführung von der Rückführungsquelle wie in Par. 102 festgelegt. Dies kann ein interner 24 V Encoder, eine Encoder Option oder eine Drehgeber Option sein.

□ 32-6* PID-Regelung

Optimieren Sie die Steuerung mit folgenden Regelungs-Parametern:

32-60 Proportionalfaktor

KPROP

Bereich

0 – 100000 ∗ 30

Funktion

Der *Proportionalfaktor* KPROP gibt den linearen Korrekturfaktor an, mit dem die Abweichung zwischen der aktuellen Soll- und Istposition bewertet und eine entsprechende Korrektur der Motordrehzahl vorgenommen wird.

Faustregel: KPROP größer = Antrieb wird „steifer”
KPROP zu hoch = Neigung zum starken Überschwingen.

32-61 Differentialwert für PID-Regelung

KDER

Bereich

0 – 100000 ∗ 0

Funktion

Der *Differentialfaktor* KDER ist der Korrekturfaktor, mit dem die Geschwindigkeit der Änderung eines Motorpositionsfehlers bewertet wird.

Der *Differentialfaktor* wirkt der durch einen hohen P-Anteil verursachten Überschwingungsneigung entgegen und „dämpft” das System. Ein zu groß gewählter Differentialfaktor führt jedoch zu einem „nervösen” Antrieb.

32-62 Integralfaktor

KINT

Bereich

0 – 100000 ∗ 0

Funktion

Der *Integralfaktor* KINT ist der Gewichtungsfaktor, mit dem im Zeitpunkt n die Summe aller Motorpositionsfehler bewertet wird.

Der *Integralfaktor* des PID-Filters bewirkt ein entsprechend zeitlich anwachsendes, korrigierendes Motordrehmoment. Durch den *Integralfaktor* wird ein statischer Positionsfehler zu Null ausgeregelt, auch wenn eine konstante Last am Motor anliegt. Ein zu großer *Integralfaktor* führt jedoch zu einem „nervösen” Antrieb.

32-63 Grenzwert für die Integralsumme

KILIM

Bereich

0 – 1000 ∗ 1000

0 = Integral aus

Funktion

Dieser Parameter begrenzt die Integralsumme, um bei Rückführungsfehlern Instabilität und Schwingen zu vermeiden.

32-64 PID Bandbreite

BANDWIDTH

Bereich [Einheit]

0 – 1000 [1/10 %] ∗ 1000

0 = PID aus

Funktion

Der Wert 1000 bedeutet, dass der PID-Filter den vollen Sollwert ausgeben kann. Bei einer *Bandbreite* von 500 werden nur 50 % des Sollwerts ausgeben. Kleinere Werte als 1000 begrenzen also den P-Anteil entsprechend.

Die Bandbreite, in der der PID-Regel-Algorithmus wirken soll, kann begrenzt werden, um zum Beispiel bei schwingungsgefährdeten Systemen das Aufschaukeln der Schwingungen zu vermeiden.

Dann ist es jedoch notwendig, wesentlich höhere Werte für die Parameter 32-65 *Geschwindigkeits-* und 32-66 *Beschleunigungs-Feed-forward* einzugeben, um die entsprechende Regelung zu erreichen. Ein so eingestelltes System ist zwar nicht mehr so dynamisch, dafür aber wesentlich stabiler und neigt weniger zu unkontrollierten Schwingungen.

32-65 Geschwindigkeits-Feed-forward

FFVEL

Bereich

0 – 100000 ∗ 0

Funktion

Wenn eine Regelung in der *Bandbreite* begrenzt ist, muss eine Grundgeschwindigkeit vorgegeben werden, damit ausgeschlossen wird, dass die Regelung durch die eingestellte Begrenzung das Fahren des Antriebs gänzlich verhindert.

Geschwindigkeits-Feed-forward gibt an, mit welcher Geschwindigkeit der Vorwärtsschub ausgeführt wird.

Beim Arbeiten mit einem normalen PID-Algorithmus muss FFVEL immer dieselbe Größe wie der D-Anteil haben, um eine typische D-Dämpfung zu erreichen.

32-66 Beschleunigungs-Feed-forward

FFACC

Bereich

0 – 100000	∗ 0

Funktion

Geben Sie eine Grundbeschleunigung vor, wenn Sie die Regelung in der Bandbreite begrenzt haben. Damit verhindern Sie, dass die Regelung durch die eingestellte Begrenzung überhaupt nicht beschleunigt. FFACC gibt an mit welcher Beschleunigung der Vorwärtsschub ausgeführt wird.

Bei einem normalen PID-Algorithmus ist dieser Wert 0.

32-67 Max. Tolerierter Positionsfehler

POSERR

Bereich [Einheit]

1 – MLONG [qc]	∗ 20000

Funktion

Der *maximal tolerierte Positionsfehler* definiert die erlaubte Toleranz zwischen der aktuellen Istposition und der errechneten Sollposition. Wird der mit POSERR definierte Wert überschritten, wird die Lageregelung aller Achsen abgeschaltet und ein Schleppfehler ausgelöst.

Der Schleppabstand hat keinen Einfluss auf die Positioniergenauigkeit, sondern bestimmt lediglich, wie exakt der theoretisch errechnete Verfahrweg eingehalten werden muss, ohne dass ein Fehler ausgelöst wird.

ACHTUNG!:
Der Schleppabstand darf aus Sicherheitsgründen nicht zu groß gewählt werden, um Mensch und Maschine nicht zu gefährden.

ACHTUNG!:
Andererseits können zu kleine Werte für den *max. tolerierten Positionsfehler* häufige Fehlermeldungen zur Folge haben. Als Richtwert kann die vierfache Strichzahl des Drehgebers angesetzt werden, was wiederum einer Drehgeberumdrehung entspricht.

32-68 Reversierungsverhalten Slave

REVERS

Option

∗	Reversieren erlaubt	[0]
	Reversieren nur erlaubt, wenn der Master rückwärts fährt	[1]
	Reversieren gesperrt	[2]

Funktion

REVERS legt das Verhalten beim Rückwärtsfahren (Fahren in negativer Richtung) fest: Ob Reversieren erlaubt ist, nur erlaubt ist, wenn der Master rückwärts fährt oder grundsätzlich gesperrt ist**.**

Die entsprechende Einschränkung ist immer gültig, d.h. bei der Antriebssynchronisation (SYNCP, SYNCV, SYNCM, SYNCC, etc.), den Positionierbefehlen (POSA, POSR), dem Drehzahlbefehl CVEL und auch bei der *Testfahrt*.

Wenn Sie das automatische Reversieren bei der *Testfahrt* verhindern wollen, stellen Sie den Parameter auf [1] oder [2] ein.

32-69 Abtastzeit für PID-Regelung

TIMER

Bereich [unit]

1 – 1000 [ms]	∗ 1

Funktion

Der Parameter TIMER bestimmt die Abtastzeit des Regelalgorithmus. Erhöhen Sie den Wert der Werkseinstellung zum Beispiel

- bei sehr kleinen Pulsfrequenzen wie 1 bis 2 qc per Abtastzeit. Sie brauchen mindestens 10 bis 20 qc per Abtastzeit.
- Oder bei sehr trägen Systemen mit einer großen Totzeit. Würde man hier mit 1 ms regeln, würden große Motoren schwingen.

Sinnvollerweise setzt man den Wert nicht höher als 1000 (= 1 s). Das wäre bereits eine sehr träge Regelung.

ACHTUNG!:
Beachten Sie, dass der Parameter einen direkten Einfluss auf die PID-Schleife hat, wenn Sie zum Beispiel den TIMER verdoppeln, wirkt Par. 32-60 *Proportionalfaktor* doppelt so stark.

32-70 Abtastzeit für Profilgenerator

PROFTIME

Option

*	1 ms	[1]
	2 ms	[2]
	3 ms	[3]
	4 ms	[4]
	5 ms	[5]

Funktion

Der Parameter ermöglicht es die Abtastzeit für den Profilgenerator unabhängig von der Abtastzeit der PID-Regelung zu setzen.

Bei anspruchsvollen Regelaufgaben im Hintergrund (SYNCP, SYNCM, SYNCC) kann die Ausführungszeit des APOSS-Programms drastisch ansteigen. In solchen Fällen kann die Abtastzeit des Profilgenerators auf [2] erhöht werden, um mehr Zeit für das APOSS-Programm zur Verfügung zu haben. Höhere Werte als 2 ms sind aber kaum von Vorteil.

ACHTUNG!:
Nach einem SET PROFTIME Befehl müssen die Befehle VEL, ACC und DEC gesetzt werden.

32-71 Größe des Regelfensters (Aktivierung)

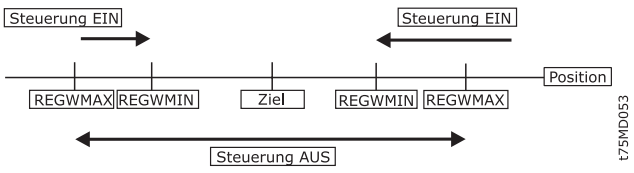
REGWMAX

Bereich

0 – MLONG [qc] * 0

Funktion

Die Parameter REGWMAX und REGWMIN werden benutzt, um die Lageregelung innerhalb von definierten Bereichen (*Regelfenster*) an- und abschalten zu können: REGWMAX gibt dabei die Größe des Fensters an, außerhalb dessen die Regelung wieder beginnen soll.



32-72 Größe des Regelfensters (Deaktiv.)

REGWMIN

Bereich

0 – MLONG [qc] * 0

Funktion

REGWMIN gibt die Größe des Fensters an, innerhalb dessen die Regelung deaktiviert werden soll, bis wieder das Regelfenster Par. 32-71 *Größe des Regelfensters (Aktivierung)* erreicht wird.

□ 32-8* Geschwindigkeit & Beschleunigung

Benutzen Sie folgende Parameter zum Bestimmen der Geschwindigkeit, Beschleunigung und Rampen.

32-80 Maximalgeschwindigkeit (Encoder)

VELMAX

Bereich [Unit]

1 – 100000 [RPM] * 1500

Funktion

VELMAX definiert die Nenngeschwindigkeit des Antriebs in U/Min. Der Wert wird zur Berechnung von Rampen und Ist-Geschwindigkeiten benötig.

ACHTUNG!:
Die Nenngeschwindigkeit bezieht sich auf die Drehzahl des Drehgebers.

32-81 Kürzeste Rampe

RAMPMIN

Bereich [Einheit]

0,001 – 3600,000 [s] * 1,000

Funktion

Der Parameter legt die *kürzeste Rampe* (maximale Beschleunigung) fest. Er gibt an wie lange die Beschleunigungsphase mindestens dauert, um die Nenngeschwindigkeit zu erreichen.

Wenn Sie mit MCO 305 arbeiten, dann sollten Sie die Rampen immer über die Optionskarte setzen und nicht im FC 300. Die FC 300 Rampen müssen immer auf Minimum gesetzt sein.

32-82 Rampenform

RAMPTYPE

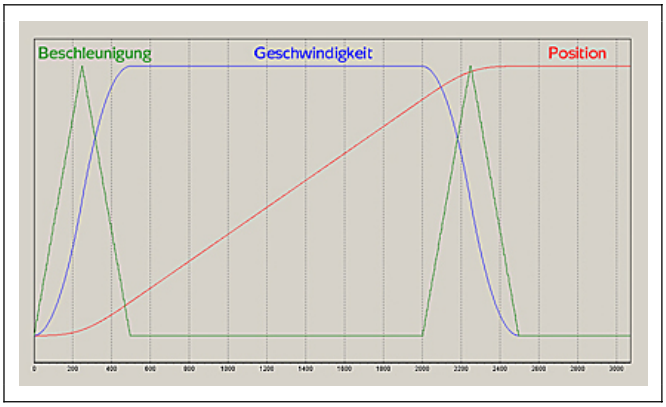
Option

*	Linear	[0]
	S-Rampe	[1]
	Bewegungsprofil mit Ruckbegrenzung	[2]

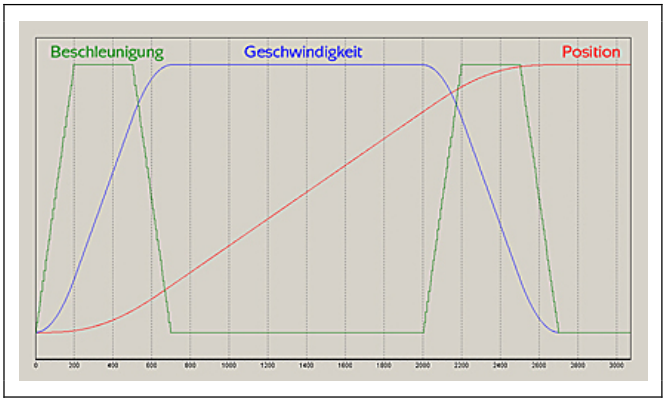
Funktion

RAMPTYPE bestimmt die Rampenform: Trapez, sinusförmig oder mit Ruckbegrenzung. Diese Rampentypen sind für alle Fahrbewegungen (POSA, POSR, CVEL und MOTOR STOP) relevant, nicht aber bei SYNCx-Befehlen.

Rampentyp 0: Trapez



Rampentyp 1: S-Rampe



Rampentyp 2: Mit Ruckbegrenzung

Die Bewegung beginnt mit der Beschleunigung 0. Diese wird durch einen maximalen Ruck solange erhöht, bis die maximale Beschleunigung, die in RAMPMIN festgelegt ist, erreicht ist. Dann wird die Bewegung mit der maximalen Beschleunigung fortgeführt. Am Ende wird die Beschleunigung durch den maximalen Ruck verringert bis die Beschleunigung erneut 0 beträgt.

Der maximale Ruck wird durch den internen Parameter *Ruckdauer* JERKMIN berechnet.

Funktion JERKMIN

JERKMIN ist die Zeitspanne in [ms] in der die definierte Maximalbeschleunigung erreicht werden soll.

Es stehen vier verschiedene JERKMIN Varianten zur Verfügung, siehe „Ruckbegrenzung" im Kapitel „Funktionen und Beispiele": JERKMIN, JERKMIN2, JERKMIN3, JERKMIN4.

Der bei Par. 32-82 *Rampentyp* = 2 benutzte maximale Ruck wird durch JERKMIN mit folgender Formel berechnet:

$$\text{Max. Beschleunigung} = \frac{\text{Maximale Geschwindigkeit}}{\text{Par. 32 - 81 Kürzeste Rampe}}$$

$$\text{Maximaler Ruck} = \frac{\text{Maximale Beschleunigung}}{\text{Par. Ruckdauer JERKMIN}}$$

Beachten Sie, dass Par. 32-81 *Kürzeste Rampe* und *Ruckdauer* Zeitangaben in Millisekunden sind.

Kalkulationsbeispiel:

Par. 32-80 VELMAX = 3000 (U/min)
Par. 32-01 ENCODER = 500 Pulse/Umdrehung
Par. 32-81 RAMPMIN = 500 ms
Parameter JERKMIN = 200 ms

Das resultiert in:

$$\text{VELMAX} = 3000 * 500 * \frac{4}{60} = 100.000 \text{ qc/s} = 100 \text{ qc/ms}$$

MaxAcc = 200.000 qc/s^2 = 0,2 qc/ms^2
MaxJerk = 1.000.000 qc/s^3 = 0,001 qc/ms^3

ACHTUNG!:
Eine geänderte JERKMIN Einstellung wird nach dem nächsten Fahrbefehl (POSA, POSR, CVEL oder MOTOR STOP) ausgeführt und in der Beschleunigungsberechnung berücksichtigt. Im NOWAIT ON Modus ist es somit möglich, während einer Bewegung durch nochmaligen Aufruf des POSA Befehls den Bewegungsvorgang online (das heißt ohne Stopp) an neue Definitionen anzupassen.

ACHTUNG!:
Wenn bei JERKMIN2, JERKMIN3 und/oder JERKMIN4 „0" gesetzt ist, wird der gleiche Wert wie bei JERKMIN verwendet.

ACHTUNG!:
Der Parameter *Ruckdauer* JERKMIN ist nicht im Dialogfeld Achsenparameter Geschwindigkeit implementiert und kann auch nicht über das LCP-Panel eingegeben werden. Der Parameter muss daher im Programm gesetzt werden, zum Beispiel:

```
SET JERKMIN 100    // Zeit in ms bis zum Erreichen
                   // der Maximalbeschleunigung
SET RAMPTYPE 2
      // Bewegungsprofil mit Ruckbegrenzung
```

32-83 Geschwindigkeitsteiler

VELRES

Bereich

1 – 10000	∗ 100

Funktion

Der *Geschwindigkeitsteiler* definiert eine Bezugsgröße für die Geschwindigkeitswerte der Fahrbefehle und Parameter. Die Angabe der Geschwindigkeit und Beschleunigung kann dann in ganzen Zahlen, bezogen auf diese Normierung, erfolgen. Der Wert 100 bedeutet, dass sich die Angaben in den Befehlen auf 100 beziehen, also in Prozent.

32-84 Default-Geschwindigkeit

DFLTVEL

Bereich

1 – p. 32-83 VELRES	∗ 50

Funktion

Default-Geschwindigkeit gibt die Default-Geschwindigkeit an, die immer dann verwendet wird, wenn keine Geschwindigkeit im Verfahrsatz definiert wurde. Der Wert bezieht sich auf den *Geschwindigkeitsteiler* VELRES.

32-85 Default-Beschleunigung

DFLTACC

Bereich

1 – p. 32-83 VELRES	∗ 50

Funktion

Default-Beschleunigung gibt die Beschleunigung an, die verwendet wird, wenn keine explizite Angabe vorliegt. Die Angabe erfolgt im Verhältnis zu Par. 32-81 *Kürzeste Rampe* und bezieht sich auf den Par. 32-83 *Geschwindigkeitsteiler*.

□ MCO weitere Einstellungen

□ 33-0* Homefahrt

Benutzen Sie folgende Parameter um das Verhalten während der Homefahrt festzulegen:

33-00 Homefahrt erzwingen?

HOME_FORCE

Option

*	Homefahrt nicht erzwingen	[0]
	Homefahrt erzwingen	[1]

Funktion

0 = Nach dem Einschalten gilt die Istposition als Realnullpunkt.

1 = Nach dem Einschalten des FC 300 sowie nach dem Ändern von Achsparametern muss vor einem Fahrbefehl – ob direkt oder durch ein Programm ausgeführt – zwingend zuerst eine Homefahrt erfolgen.

Wenn dieser Parameter auf [1] gesetzt ist, muss eine Homefahrt ausgeführt werden, bevor irgendeine andere Positionierfahrt ausgeführt werden kann.

Bei einem Fahrbefehl ohne erfolgreich ausgeführte Homefahrt wird der Fehler 106 ausgelöst.

ACHTUNG!:
Aus Sicherheitsgründen und zur Vermeidung von Fehlpositionierungen sollte der Parameter immer auf [1] gesetzt und damit eine Homefahrt erzwungen werden. Es muss in diesem Fall jedoch berücksichtigt werden, dass dann alle Programme vor dem ersten Fahrbefehl einen HOME Befehl ausführen müssen, damit sie einwandfrei funktionieren.

33-01 Nullpunkt-Offset bezügl. Home-Position

HOME_OFFSET

Bereich [Einheit]

-MLONG – MLONG [qc] * 0

Funktion

HOME_OFFSET wird benutzt, um einen Offset (Versatz) einzuführen, vergleichbar mit dem Referenzschalter oder Indexpuls. Nach der Homefahrt wird der Antrieb auf HOME_OFFSET positioniert. An dieser Stelle wird auch der Maschinennullpunkt oder Index gesetzt.

33-02 Homefahrt-Rampe

HOME_RAMP

Bereich

1 – Par. 32-83 VELRES * 10

Funktion

Beschleunigung, die für die Fahrt zur Home-Position verwendet wird. Die Angabe bezieht sich auf die minimale Rampe, die in Par. 32-81 *Kürzeste Rampe* definiert ist. Die Einheit ergibt sich durch den Par. 32-83 *Geschwindigkeitsteiler*, standardgemäß in % von der *kürzesten Rampe*; 50 % bedeutet dann halb so schnell, d.h. doppelt so lange.

Für die *Homefahrt-Rampe* ergibt sich folgender Zusammenhang:

$$\text{Homefahrt - Rampe [ms]} = \frac{\text{P. 32 - 83 Geschw.teiler}}{\text{P. 33 - 02 HomefahrtRampe}} * \text{P. 32 - 81 Kürz.Rampe [ms]}$$

ACHTUNG!:
Die *Homefahrt-Rampe* kann nie einen höheren Wert haben als die *Default-Beschleunigung* in Par. 32-85.

33-03 Homefahrt-Geschwindigkeit

HOME_VEL

Bereich

- p. 32-83 – p. 32-83 * 10

Funktion

HOME_VEL bestimmt die Geschwindigkeit, mit der die Fahrt zum Referenzschalter ausgeführt wird. Die Angabe ist auf die Nenngeschwindigkeit bezogen und von dem Parameter 32-83 abhängig.

Ein negatives Vorzeichen heißt, dass die Suche in der anderen Richtung erfolgt.

P. 33 - 03 Homefahrt - Geschwindigkeit [U/Min] =

$$\text{Homefahrt - Geschwindigkeit} * \frac{\text{P. 32 - 80 Max.Geschwindigkeit}}{\text{P. 32 - 83 Geschwindigkeitsteiler}}$$

ACHTUNG!:
Da immer in der gleichen Drehrichtung (abhängig vom Vorzeichen) nach dem Referenzschalter gesucht wird, sollte dieser an den Grenzen des Fahrbereichs angebracht werden. Nur so kann sichergestellt werden, dass sich der Antrieb bei einer Homefahrt aus allen Positionen auch tatsächlich in Richtung des Referenzschalters und nicht von ihm weg bewegt.

Um eine gute Repetierbarkeit der Referenzfahrt zu erhalten, sollte mit höchstens 10 % der maximalen Drehzahl gefahren werden.

33-04 Homefahrt-Verhalten

HOME_TYPE

Option

*	Reversieren und Index	[0]
	Reversieren, kein Index	[1]
	Vorwärts und Index	[2]
	Vorwärts, kein Index	[3]

Funktion

0 = Mit Homefahrt-Geschwindigkeit und Richtung bis zum Referenzschalter fahren, dann Reversieren und langsam den Schalter verlassen; anschließend zum nächsten Indeximpuls fahren.

1 = Wie 0, aber ohne Suchen des Indeximpulses.

2 = Wie 0, aber ohne Reversieren, sondern in gleicher Richtung weiter aus dem Schalter herausfahren.

3 = Wie 1, aber ohne Reversieren.

□ 33-1* Synchronisation

Positions-, Geschwindigkeits- und Winkel/Positionssynchronisation, mit oder ohne Marker und mehr ist mit folgenden Parametern möglich:

33-10 Synchronisationsfaktor Master (M:S)

SYNCFACTM

Bereich

−MLONG – MLONG * 1

−MLONG bis −1 = dreht die Richtung der Synchronisation (Verhältnis zum Master) um

Funktion

Die Synchronisation wird mit einem Verhältnis von Master:Slave in qc beschrieben; SYNCFACTM bestimmt den Synchronisationsfaktor für den Master.

Syncfaktor Master und Par. 33-11 *Syncfaktor Slave* ermöglichen den Ausgleich unterschiedlicher Getriebefaktoren bzw. die Anpassung der Drehzahl des Slaves im Verhältnis zur festgelegten Drehzahl des Masters.

Slave Geschwindigkeit =

$$\text{Master Geschwindigkeit} * \frac{\text{Par. 33 - 11 Syncfaktor Slave}}{\text{Par. 33 - 10 Syncfaktor Master}}$$

In Verbindung mit der Kurvensynchronisation (CAM) werden die Parameter SYNCFACTM und SYNCFACTS zur Umrechnung der qc in MU-Einheiten benutzt.

Dadurch kann der Anwender im CAM-Editor mit sinnvollen Einheiten arbeiten. Siehe Beispiel 2.

$$\frac{\text{Getriebfaktor} * \text{ Drehgeberauflösung} * 4}{\text{Skalierfaktor}} \text{qc} = 1 \text{ MU}$$

vorausgesetzt dass:

$$\text{Getriebefaktor} = \frac{\text{Motorumdrehungen}}{\text{Umdrehungen am Abtrieb}}$$

Drehgeber = Inkrementalgeber (bei Absolutgebern entfällt der Multiplikator 4).

Skalierfaktor = Anzahl der Benutzereinheiten BE [qc], die einer Umdrehung am Antrieb entsprechen.

Beispiel 1

Wenn der Master zweimal so schnell fahren soll wie der Slave, beträgt das Verhältnis 2:1.

Par. 33-10 *Syncfactor Master* = 2
Par. 33-11 *Syncfactor Slave* = 1

Beispiel 2

Transportband:

Die Eingabe soll in 1/10 mm Auflösung möglich sein.

Der Antrieb ist mit dem Transportband mit einer Getriebeübersetzung von 25:11 verbunden; das heißt der Motor macht 25, das Zahnriemenrad 11 Umdrehungen.
Getriebefaktor = 25/11

Inkrementalgeber direkt am Master-Antrieb; Drehgeber-Auflösung = 4096

Das Zahnriemenrad hat 20 Zähne/Umdrehung, 2 Zähne entsprechen 10 mm, daher entspricht 1 Umdrehung = 100 mm Transportbandvorschub. Skalierfaktor ist demnach 1000.

$$\frac{25/11 * 4096 * 4}{1000} qc = \frac{25 * 4096 \times * 4}{1000 \times 11} qc = 1\ MU$$

$$= \frac{2048}{55} qc = 1\ MU = \frac{\text{Par. 33 - 10 Syncfaktor Master}}{\text{Par. 33 - 11 Syncfaktor Slave}}$$

Um mit 1/10 Grad Einteilung zu arbeiten, setzen Sie die Parameter wie folgt:

Par. 33-10 *Syncfaktor Master* = 2048
Par. 33-11 *Syncfaktor Slave* = 55

Beispiel 3

Berechnung des Skalierfaktors bei einem Reibantrieb:

Der Abtrieb sei mit einem Reibrad (Radius 60 mm) versehen; es soll mit einer Auflösung von 1/10 mm gearbeitet werden. Eine Umdrehung am Abtrieb berechnet sich demnach:

Skalierfaktor = 2 Π r * 10 = 2 Π * 60 * 10
= 3969,91

Skalierfaktor = 3970

Da durch das Aufrunden auf jeden Fall ein Fehler entsteht, muss nach jeder vollen Umdrehung ein Markerabgleich durchgeführt werden.

33-11 Synchronisationsfaktor Slave (M:S)

SYNCFACTS

Bereich

–MLONG – MLONG ∗ 1

Funktion

Die Synchronisation wird mit einem Verhältnis von Master:Slave in qc beschrieben; *Syncfaktor Slave* bestimmt dabei den Synchronisationsfaktor für den Slave.

Parameter 33-10 *Synchronisationsfaktor Master* und 33-11 *Syncfaktor Slave* ermöglichen den Ausgleich unterschiedlicher Getriebefaktoren bzw. die Anpassung der Drehzahl des Slaves im Verhältnis zur festgelegten Drehzahl des Masters.

$$\text{Slavegeschwindigkeit} = \text{Mastergeschwindigkeit} * \frac{\text{Par. 33 - 11 Syncfaktor Slave}}{\text{Par. 33 - 10 Syncfaktor Master}}$$

In Verbindung mit der Kurvensynchronisation (CAM) werden die Parameter *Synchronisationsfaktor Master* und *Slave* zur Umrechnung der qc in MU-Einheiten benutzt. Dadurch kann der Anwender im CAM-Editor mit sinnvollen Einheiten arbeiten. Siehe Beispiel in Par. 33-10.

Siehe Voraussetzung für die Formel bei Par. 33-10 *Synchronisationsfaktor Master*.

$$\frac{\text{Getriebfaktor} * \text{ Drehgeberauflösung} * 4}{\text{Skalierfaktor}} qc = 1\ MU$$

Beispiele

Siehe Par. 33-10 *Synchronisationsfaktor Master*.

33-12 Positionsoffset für Synchronisation

SYNCPOSOFFS

Bereich [Unit]

-MLONG/p. 33-11 SYNCFACTS –
MLONG/p. 33-11 SYNCFACTS [qc] ∗ 0

Funktion

Setzt den Offset für die Positionssynchronisation (SYNCP). Dieser Offset ist auch bei einer Positionssynchronisation mit Markerkorrektur gültig (SYNCM).

Der *Positionsoffset* kann während der Synchronisation jederzeit per Befehl online verändert werden.

$$\text{Slave Offset} = \text{P. 33 - 12 SYNCPOSOFFS} * \frac{\text{Par. 33 - 11 Syncfaktor Slave}}{\text{Par. 33 - 10 Syncfaktor Master}}$$

Der Offset für die Positionssynchronisation wird sofort ausgeführt, wenn der Befehl SYNCP folgt.

Beim Start von SYNCM dagegen wird auf die erste Auswertung der Markerpulse gewartet. Erst dann wird der Offset angewandt.

Zur Vermeidung von Kompatibilitätsproblemen sollten Sie mit Par. 33-23 das *Startverhalten* von SYNCM festlegen.

33-13 Genauigkeitsfenster für Positionssync.

SYNCACCURACY

Bereich [Unit]

-MLONG – MLONG [qc] ∗ 1000

0 – MLONG = Ein positives Vorzeichen liefert den absoluten Wert an SYNCERR.

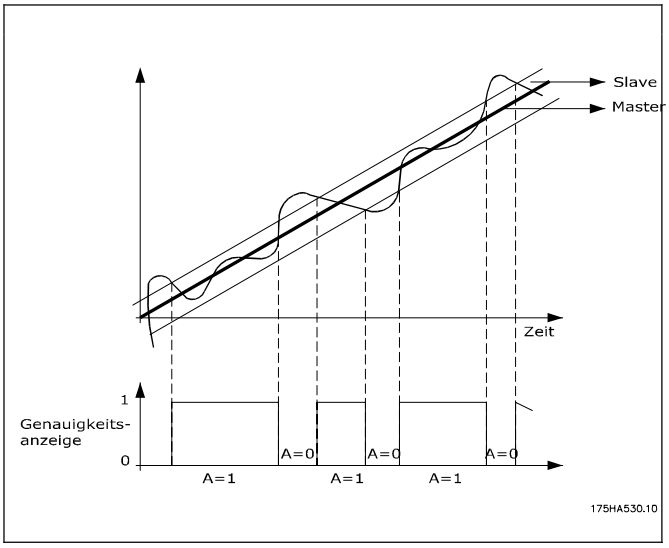
–MLONG to –1 = Ein negatives Vorzeichen liefert den Synchronisationsfehler an SYNCERR mit Vorzeichen. Daraus lässt sich dann erkennen, ob die Synchronisation voraus- oder nachläuft.

Funktion

Der Parameter gibt an, wie groß die Differenz zwischen aktueller Master- und Slave-Position bei einer Positionssynchronisation (SYNCP und SYNCM) sein darf, damit die geforderte Genauigkeit (ACCURACY) noch erfüllt ist. SYNCERR dagegen liefert den tatsächlichen Synchronisationsfehler des Slaves in Benutzereinheiten.

Ob SYNCACCURACY erfüllt wird, können Sie im Programm mit SYNCSTAT abfragen.

SYNCACCURACY ist wichtig für die Markersynchronisation um READY melden zu können, da andernfalls vorher n-mal SYNCERR abgefragt und verglichen werden müsste.



Genauigkeitsfenster gesetzt durch SYNCACCURACY.

Die dunkle Linie zeigt die Positionen, denen Master und Slave folgen. SYNCACCURACY setzt das Fenster auf 100 und somit wird das ACCURACY-Flag gesetzt, wenn der Slave innerhalb des Fensters ist.

33-14 Rel. Geschwindigkeitslimit Slave

SYNCVELREL

Bereich [Einheit]

0 – 100 [%] ∗ 0

0 = Aus, d.h. keine Beschränkung

Syntax

SET SYNCVELREL wert

wert = Prozentwert

Funktion

Tolerierte Abweichung des Folgeantriebs von der Master-Geschwindigkeit in %.

Dieser Parameter gibt an, um wie viel Prozent der Folgeantrieb von der Geschwindigkeit des Masters abweichen darf, während er versucht die Synchronisation wieder herzustellen.

Zum Beispiel bei einer Änderung von Par. 33-12 *Positionsoffset für Sync.* oder beim Start der Synchronisation oder bei der Korrektur der Abweichung bei der Markerauswertung. Dabei gilt Folgendes:

Muss der Slave aufholen, fährt er mit der maximal erlaubten Drehzahl, wobei dies entweder die mit VEL eingestellte Drehzahl ist oder die durch

MAVEL + (MAVEL * SYNCVELREL/100)

berechnete, je nachdem welche von beiden kleiner ist. (MAVEL ist aktuelle Master-Geschwindigkeit).

Muss der Slave abbremsen und auf den Master warten, fährt er mindestens mit der Drehzahl

MAVEL – (MAVEL * SYNCVELREL/100).

Das heißt, wenn SYNCVELREL zum Beispiel 50 ist, wird der Folgeantrieb nicht langsamer fahren als MAVEL/2.

33-15 Markeranzahl Master

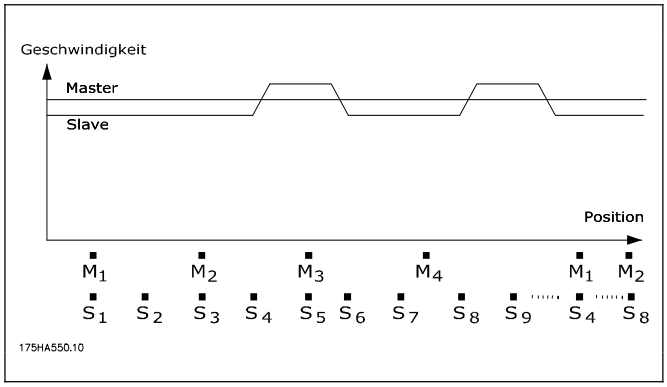
SYNCMARKM

Bereich

1 – 10000 ∗ 1

Funktion

Markeranzahl Master und *Slave* müssen entsprechend dem Verhältnis der Anzahl der Markersignale des Master zum Slave gesetzt werden. Ein Verhältnis von 1:1 bedeutet, dass sich jeder Slave-Marker auf jeden Master-Marker abstimmt. Ein Verhältnis von 2:1 bedeutet, dass sich jeder Slave-Marker auf jeden zweiten Master-Marker abstimmt.



33-16 Markeranzahl Slave

SYNCMARKS

Bereich

1 – 10000 ∗ 1

Funktion

Markeranzahl Master (Par. 33-15) und *Slave* müssen entsprechend dem Verhältnis der Anzahl der Markersignale des Master zum Slave gesetzt werden.

Ein Verhältnis von 1:1 bedeutet, dass sich jeder Slave-Marker auf jeden Master-Marker abstimmt. Ein Verhältnis von 2:1 bedeutet, dass sich jeder Slave-Marker auf jeden zweiten Master-Marker abstimmt.

Beispiel

Der Master-Marker ist ein externes Signal, das meldet, wenn ein Transportgut ankommt; der dazugehörende Slave-Marker ist der Indeximpuls vom Motor. Wenn der Motor immer drei Umdrehungen benötigt bis ein Gut ankommt, dann bedeutet das, dass auch immer drei Indeximpulse vergehen müssen bis ein Marker kommt. Daraus ergibt sich ein Verhältnis von 3:1; es wird nur jeder dritte Slave-Puls ausgewertet.

33-17 Markerabstand Master

SYNCMPULSM

Bereich [Unit]

0 – MLONG ∗ 4096
[qc] or in CAM mode [MU]

Funktion

Markerabstand Master gibt an wie viele qc (Master) zwischen zwei Master-Markern liegen bzw. im CAM-Modus den Abstand zwischen Sensor und Arbeitsposition in MU.

Wenn man den Drehgeber-Indexpuls als Markersignal benutzt, beträgt der Abstand zwischen zwei Markern die Auflösung [qc] des Drehgebers.

Wenn externe Markersignale benutzt werden, können Sie den Markerabstand mit dem Programm „Marker count" (siehe Programmbeispiele) messen, falls er nicht bekannt ist.

Markerabstand Master gilt nur für Synchronisationen mit Markerkorrektur (SYNCM und SYNCCMM).

Bei einer CAM-Synchronisation wird statt des Abstands zwischen zwei Master-Markern der Abstand des Sensors zur Arbeitsposition in MU angegeben. (Der Abstand ergibt sich automatisch durch die Mastertaktlänge [Mt].)

Wenn der Parameter größer als eine Master-Taktlänge [Mt] ist, wird automatisch ein Marker-FIFO-Register für die Handhabung der Markerkorrektur gebildet.

33-18 Markerabstand Slave

SYNCMPULSS

Bereich [Unit]

0 – MLONG ∗ 4096
[qc] or in CAM mode [BE]

Funktion

Der *Markerabstand Slave* gibt an wie viele qc (Slave) zwischen zwei Slave-Markern liegen, bzw. im CAM-Modus den Abstand des Sensors zur Arbeitsposition in BE.

Markerabstand Slave gilt nur für Synchronisationen mit Markerkorrektur (SYNCM und SYNCCMS).

33-19 Markertyp Master

SYNCMTYPM

Option

∗	Drehgeber Z positive Flanke	[0]
	Drehgeber Z negative Flanke	[1]
	Externer Marker positive Flanke	[2]
	Externer Marker negative Flanke	[3]

Funktion

Definiert den Signal- bzw. Markertyp für den Master-Marker: Indexpuls des Drehgebers oder externer Marker.

Markertyp Master gilt nur für Synchronisationen mit Markerkorrektur (SYNCM und SYNCCMM) oder wenn Sie den Befehl MIPOS im Programm verwenden wollen.

Externes Master-Markersignal: Eingang 5.

33-20 Markertyp Slave

SYNCMTYPS

Option

∗	Drehgeber Z positive Flanke	[0]
	Drehgeber Z negative Flanke	[1]
	Externer Marker positive Flanke	[2]
	Externer Marker negative Flanke	[3]

Funktion

Definiert den Signaltyp für den Slave-Marker: Indexpuls des Drehgebers oder externer Marker.

SYNCMTYPS gilt nur für Synchronisationen mit Markerkorrektur (SYNCM und SYNCCMS) oder wenn Sie den Befehl IPOS im Programm verwenden wollen.

Externes Slave-Marker-Signal: Input 6

33-21 Master-Marker Toleranzfenster

SYNCMWINM

Bereich [Unit]

0 – P. 33-17 *Markerabstand Master* ∗ 0

0 = Aus

[qc] oder im CAM Modus [MU]

Funktion

Das *Master-Marker Toleranzfenster* gibt an, wie groß die erlaubte Toleranz für das Auftreten der Marker ist.

Mit der Werkseinstellung [0] wird das Fenster nicht überwacht, das heißt es wird immer auf den nächsten Marker synchronisiert, auch wenn dieser einen wesentlich größeren Abstand hat.

Mit jeder anderen Einstellung werden nur Marker akzeptiert, die innerhalb des Fensters liegen. Wenn innerhalb des Toleranzfensters kein Marker kommt, wird das entsprechende Flag (SYNCSTAT) gesetzt und keine Markerkorrektur durchgeführt. Es wird auch der entsprechende andere Marker ignoriert und erst beim nächsten Mal wieder korrigiert – also kein Aufholen zum nächsten Marker.

Nach dem Start von SYNCM oder SYNCCSTART beginnt die Überwachung erst nachdem der erste Marker gefunden ist.

ACHTUNG!:
Änderungen des Parameters werden sofort aktiv – nicht erst nach dem nächsten SYNCM Befehl.

Beispiel

Par. 33-17 Markerabstand Master = 30000
Master-Marker Toleranzfenster = 1000

Es wird nur der Marker akzeptiert, der innerhalb des Intervalls von 29000 bis 31000 liegt.

33-22 Slave-Marker Toleranzfenster

SYNCMWINS

Bereich [Unit]

0 – P. 33-18 *Markerabstand Slave* ∗ 0

0 = Aus

[qc] oder im CAM Modus [BE]

Funktion

Das *Slave-Marker Toleranzfenster* gibt an, wie groß die erlaubte Toleranz für das Auftreten der Marker ist.

Mit der Werkseinstellung [0] wird das Fenster nicht überwacht, das heißt es wird immer auf den nächsten Marker synchronisiert, auch wenn dieser einen wesentlich größeren Abstand hat.

Mit jeder anderen Einstellung werden nur Marker akzeptiert, die innerhalb des Fensters liegen. Wenn innerhalb des Toleranzfensters kein Marker kommt, wird das entsprechende Flag (SYNCSTAT) gesetzt und keine Markerkorrektur durchgeführt. Es wird auch der entsprechende andere Marker ignoriert und erst beim nächsten Mal wieder korrigiert – also nicht zum nächsten Marker aufgeholt.

Nach dem Start von SYNCM oder SYNCCSTART beginnt die Überwachung erst nachdem der erste Marker gefunden ist.

ACHTUNG!:
Änderungen des Parameters werden sofort aktiv – nicht erst nach dem nächsten SYNCM Befehl.

33-23 Startverhalten für Sync.

SYNCMSTART (mit Markerkorrektur)

Option

∗	Start Funktion 1	[0]
	Start Funktion 2	[1]
	Start Funktion 3	[2]
	Start Funktion 4	[3]
	Start Funktion 5	[4]
	Start Funktion 6	[5]
	Start Funktion 7	[6]
	Start Funktion 8	[1000]
	Start Funktion 9	[1001]
	Start Funktion 10	[1002]
	Start Funktion 11	[1003]
	Start Funktion 12	[1004]
	Start Funktion 13	[1005]
	Start Funktion 14	[1006]
	CAM Master Start	[2000]

Funktion

SYNCMSTART gibt an ob beim Starten der Synchronisation auf den jeweils voreilenden, nachfolgenden oder auf den dichtesten Markerimpuls des Masters aufsynchronisiert werden soll.

SYNCMSTART gilt nur für Synchronisationen mit Markerkorrektur (SYNCM und SYNCCMM).

0 = Der Slave-Marker, der dem ersten Master-Marker (nach SYNCM) folgt, wird mit dem ersten Master-Marker abgeglichen.

1 = Der erste Slave-Marker (nach SYNCM) wird mit dem folgenden Master-Marker abgeglichen.

2 = Nach Erreichen der Master-Geschwindigkeit werden die nächsten zwei Marker abgeglichen. (Korrektur durch Aufholen oder Abbremsen).

3 = Nach Erreichen der Master-Geschwindigkeit wird der nächste Slave-Marker mit dem davor liegenden Master-Marker abgeglichen. (Korrektur durch Aufholen).

4 = Nach Erreichen der Master-Geschwindigkeit wird der nächste Slave-Marker mit dem nachfolgenden Master-Marker abgeglichen. (Korrektur durch Abbremsen).

5 = Nach Erreichen der Master-Geschwindigkeit wird der nächste Slave-Marker mit dem Master-Marker abgeglichen, der am dichtesten folgt. (Korrektur durch Aufholen oder Abbremsen, je nach kürzestem Abstand.)

6 = Nach dem Befehl SYNCM werden die ersten zwei Marker genommen und auf diese aufsynchronisiert.

1000 = wie [0], aber ein *Positionsoffset für Sync.* (Par. 33-12) ist nicht aktiv, bevor die erste Markerkorrektur ausgeführt wurde.

1001 = wie [1], aber ein Offset ist nicht aktiv bevor die erste Markerkorrektur ausgeführt wurde.

1002 = wie [2], aber ein Offset ist nicht aktiv bevor die erste Markerkorrektur ausgeführt wurde.

1003 = wie [3], aber ein Offset ist nicht aktiv bevor die erste Markerkorrektur ausgeführt wurde.

1004 = wie [4], aber ein Offset ist nicht aktiv bevor die erste Markerkorrektur ausgeführt wurde.

1005 = wie [5], aber ein Offset ist nicht aktiv bevor die erste Markerkorrektur ausgeführt wurde.

1006 = wie [6], aber ein Offset ist nicht aktiv bevor die erste Markerkorrektur ausgeführt wurde.

2000 = Das Zählen der Masterpulse in MU beginnt mit dem Master-Marker.

ACHTUNG!:
Der Parameter 2000 wirkt nur bei Kurvensynchronisationen (CAM-Modus).

33-24 Markeranzahl für Fault

SYNCFAULT

Bereich

0 – 10000 ∗ 10

Funktion

Gibt an, wie oft bei einer Markersynchronisation (SYNCM und SYNCCMM) nicht ACCURACY auftauchen darf, bis FAULT eintritt.

Dieser Zustand kann im Programm mit SYNCSTAT abgefragt werden.

33-25 Markeranzahl für Ready

SYNCREADY

Bereich

0 – 10000 ∗ 1

Funktion

Gibt an wie oft bei einer Markersynchronisation (SYNCM und SYNCCMM) eine Bewertung der Synchronisation mit ACCURACY durchgeführt sein muss, bis READY erfüllt ist.

Dabei wird bei jeder Korrektur ACCURACY geprüft. Wenn ACCURACY erfüllt ist, wird 1 addiert, bis die vorgegebene Markeranzahl erreicht ist.

Die Synchronisation wird immer erst nach n Markerimpulsen beim Master Par. 33-15 *Markeranzahl Master* bewertet.

ACCURACY und READY können Sie mit SYNCSTAT abfragen.

33-26 Geschwindigkeitsfilter

SYNCVFTIME

Bereich [Unit]

–MLONG – MLONG [µs] ∗ 0

–999 – 999 = Standardtabelle

Standardtabelle

Drehgeberauflösung	τ_filt [µs]
250	39500
256	38600
500	19500
512	19000
1000	9500
1024	9300
2000	4500
2048	4400
2500	3500
4096	1900
5000	1400

Funktion

Dieser Parameter konfiguriert den Geschwindigkeitsfilter, der für die Geschwindigkeitssynchronisation verwendet wird. Da bei einer Geschwindigkeitssynchronisation nur mit der jeweils aktuellen Master-Geschwindigkeit gearbeitet wird und diese sehr kleine Werte annehmen kann (z.B. 2 qc/ms), wirkt sich eine kleine Schwankung der Geschwindigkeit bereits dramatisch aus. Um dies zu glätten, wird die folgende Filterfunktion verwendet:

Cmdvel =
Old_Cmdvel + (Actvel – Old_Cmdvel) * ms/τ_filt

Hierbei gilt:

Cmdvel	=	Sollgeschwindigkeit
Old_Cmdvel	=	Letzte Sollgeschwindigkeit
Actvel	=	Aktuelle Geschwindigkeit des Masters
ms	=	Abtastzeit (fest 1 ms)
τ_filt	=	Filterzeit Konstante

Dabei wird der Wert für τ_filt standardgemäß aus einer Tabelle genommen, in Abhängigkeit von der Drehgeberauflösung des Masters. Dieser Wert kann durch den Parameter *Geschwindigkeitsfilter* überschrieben werden und wird immer dann verwendet, wenn der *Geschwindigkeitsfilter* ungleich Null ist.

Wird der Geschwindigkeitsfilter mit einer negativen Zahl definiert, gilt der entsprechende Wert auch für eine Winkel-/Positionssynchronisation SYNCP und für eine mit Markerkorrektur SYNCM.

Es wird in diesem Fall ebenso gefiltert wie oben beschrieben, zusätzlich jedoch der gemachte Fehler aufsummiert. Diese Fehlersumme wird jeweils zu 1000/(τ*10) in die Berechnung mit einbezogen, so dass über längere Zeiträume keine Positionsabweichung entstehen kann.

Der von SYNCERR zurückgelieferte Wert enthält immer den gemachten Fehler, so dass dieser auch bei der Bewertung der Synchronität einfließt. Im Fall einer Markerkorrektur wird der Korrekturwert langsamer und mit demselben Faktor wie die Fehlersummen ausgeglichen.

Setzt man zum Beispiel einen Filterfaktor von –100000 (100 ms) wird eine Markerkorrektur innerhalb von 1 Sekunde (100 ms * 10) ausgeglichen. Diese ermöglicht eine „Zähmung" der Synchronisation ohne die Beschleunigung einzuschränken.

33-27 Offset-Filterzeit

SYNCOFFTIME

Bereich [Unit]

0 – MLONG [ms] ∗ 0

Funktion

Geschwindigkeitsausgleich eines Offsets (1. Aufsynchronisieren; 2. neuer Offset).

Die *Offset-Filterzeit* beeinflusst auch die Art, wie ein neuer *Positionsoffset für Synchronisation* (Par. 33-12) gehandhabt wird. Der Offset, der ausgeführt werden muss, wird Schritt für Schritt realisiert. Eine Schrittweite, die pro Abtastperiode (ms) auszuführen ist, wird wie folgt berechnet:

$$\text{Schrittweite} = \frac{\text{P.33 - 17 Markerabstand Master}}{\text{P.33 - 27 Offset Filterzeit (Integer Anteil)}}$$

Daher wird es also *Offset-Filterzeit* dauern, um einen Offset von Par. 33-17 *Markerabstand Master* auszuführen. Die *Offset-Filterzeit* beeinflusst auch die Marker-Startkorrektur und die Korrektur der Markerfehler (siehe Par. 33-29 *Filterzeit für Markerkorrektur*).

33-28 Markerfilter Konfiguration

SYNCMFPAR

Option

∗	Normale Funktion gemäß SYNCMFTIME	[0]
	Fester Wert für Markerfilterkonstante	[1]
	Keine Getriebekorrektur SYNCFACT	[2]
	Zeitkonstante auf Basis der *Filterzeit für Markerkorrektur* SYNCMFTIME	[4]
	Nur Glättung des Korrekturwerts, Zeitkonstante wie [4]	[16]
	Mittelung des Markerabstands immer durchführen	[64]

Funktion

Dieser Parameter wird benutzt, um das Verhalten des Markerfilters zu beeinflussen, siehe Par. 33-29 *Filterzeit für Markerkorrektur*.

Die folgenden Werte sind Bit-Wertigkeiten und können kombiniert werden:

0 = Normale Funktion des Filters, siehe Par. 33-29 *Filterzeit für Markerkorrektur* SYNCMFTIME

1 = Statt der dynamischen Markerfilter-Konstante wird ein fester Wert von SYNCMFTIME / 300 verwendet.

2 = Getriebekorrektur wird nicht durchgeführt.

4 = Zur Berechnung der Zeitkonstante für den Filter des Korrekturwertes (G_Korrektur) wird die *Filterzeit für Markerkorrektur* (Par. 33-29) anstelle der *Offset Filterzeit* (Par. 33-27) verwendet.

16 = Es wird nicht der gefilterte Markenabstand und die Abweichung berechnet, sondern lediglich der Korrekturwert mittels eines PT-Filters geglättet. Zeitkonstante für diesen Filter gemäß Bitwertigkeit 4 (siehe oben).

64 = Marker Mittelung und Marker Check werden auch durchgeführt wenn SYNCM nicht aktiv ist.

33-29 Filterzeit für Markerkorrektur

SYNCMFTIME

Bereich [Unit]

0 – MLONG [ms] ∗ 0

0 = Aus;
wenn Par. 33-26 *Geschwindigkeitsfilter* negativ ist, wird die Markerkorrektur durch SYNCVFTIME /100 gespreizt.

Anwendungsbeispiel

Bei der Zeitungsproduktion wird diese Art des Filters benötigt, um eine Förderkette zu synchronisieren. Da die Zeitungen nicht völlig regelmäßig aus der Druckmaschine kommen, wären die Bewegungen der Kette sehr hart und dynamisch, falls man ohne Filter synchronisieren würde. Mit allen anderen Arten des Filterns würde das System beginnen, in sinusförmigen Wellen zu schwingen.

Benutzt man aber diese komplexe Filtermethode, arbeitet die Synchronisation sehr gut und löst das Problem.

Funktion

SYNCMFTIME wird in ms eingegeben und wie folgt benutzt:

ACHTUNG!:
Der Master-*Geschwindigkeitsfilter* Par. 33-26 wird zur besseren Auflösung in 1/1000 ms eingegeben, der Markerfilter (SYNCMFTIME) dagegen in Einheiten von 1 ms.

Beispiel:

SET SYNCVFTIME –50000
SET SYNCMFTIME 2000

Das heißt, dass die Master-Geschwindigkeit über eine Periode von 50 ms gemittelt wird. Ein Markerfehler wird also innerhalb von 2000 ms korrigiert.

Der aktuelle gefilterte Markerabstand kann mit SYSVAR Index 4238 ausgelesen werden, wenn dieser Filter durch das Setzen von SYNCMFTIME aktiviert wurde.

Die *Filterzeit für Markerkorrektur* und die Parameter 33-27 *Offset-Filterzeit* und 33-28 *Markerfilter-Konfiguration* werden benutzt um das das Verhalten des Markerfilters zu beeinflussen (siehe unten).

Beschreibung

Das Filtern wird wie folgt gehandhabt:

Markerfilter-Berechnung nur wenn SYNCMFTIME > 0

Wenn SYNCMFPAR = 1,

kann jedes Mal, wenn ein echter Master-Marker erkannt wird, die Markerfilter-Konstante als SYNCMFTIME/300 berechnet werden.

Wenn SYNCMFPAR = 0,

kann jedes Mal, wenn ein echter Master-Marker erkannt wird, die Markerfilter-Konstante wie folgt berechnet werden

$$\text{Gefilterte alte MasterGeschwindigkeit} * \frac{\text{SYNCMFTIME}}{\text{SYNCMPULSM} * 3}$$

das heißt, dass der konstante Markerfilter als Zeitkonstante zum Filtern benutzt wird. Dann sollte die Zeit, die benötigt wird um eine Reaktion entsprechend eines konstanten Eingangswerts zu erhalten, nahezu SYNCMFTIME sein.

Die Berechnung ist notwendig, weil der Filter bei jedem Marker ausgeführt wird und nicht jede ms.

Dieser Markerfilter wird nun benutzt, um den Markerabstand zu filtern. Mit dem Ergebnis wird die notwendige Getriebekorrektur wie folgt berechnet:

$$\text{Getriebekorrektur} = \frac{\text{SYNCMPULSM - gefilterten Markerabstand}}{\text{gefilterten Markerabstand}}$$

∗ Standardeinstellung | **[ ] bei Kommunikation über serielle Schnittstelle benutzter Wert**

Filter Master-Geschwindigkeit und Getriebe-korrektur

Pro Abtastperiode wird die Master-Geschwindigkeit neu berechnet (Differenz der aktuellen zur letzten Master-Position).

IF (SYNCVFTIME < 0)

wird die gefilterte alte Master-Geschwindigkeit mit einer Filterzeit-Konstante gleich SYNCVFTIME/1000 berechnet.

Andernfalls wird die gefilterte alte Master-Geschwindigkeit gleich der aktuellen Master-Geschwindigkeit gesetzt.

Wenn
SYNCMFTIME > 0 und
SYNCMFPAR = 2

wird die Getriebekorrektur durchgeführt, indem zur aktuellen Getriebeübersetzung die mit der Übersetzung multiplizierte Master-Geschwindigkeit addiert wird.

Startkorrektur nur wenn SYNCMFTIME > 0

Die Startkorrektur ist jene Korrektur, die ausgeführt werden muss, sobald die Startbedingungen erfüllt sind. Das heißt, entweder mussten die ersten zwei Marker beobachtet werden (Par. 33-23 SYNCMSTART 1,6) oder es mussten die Master-Geschwindigkeit erreicht und zusätzlich die ersten zwei Marker beobachtet werden (SYNCMSTART 2,3,4,5).

Diese Startkorrektur wird so aufgeteilt, dass sie nach Par. 33-27 SYNCOFFTIME erledigt sein wird. (Derzeit wird sie durch die Anzahl der Marker geteilt, die in SYNCOFFTIME mit der aktuellen Master-Geschwindigkeit passiert wurden und der erhaltene Wert wird zur normalen Markerkorrektur addiert.)

Wenn SYNCOFFTIME == 0,

wird die Startkorrektur sofort ausgeführt, das bedeutet, dass sie innerhalb von zwei Markern erledigt ist.

Markerkorrektur SYNCMFTIME > 0

Zuerst wird die verbleibende Startkorrektur vom Markerfehler abgezogen. Dann wird die *Filterzeit Markerkorrektur* entsprechend dem Par. 33-27 *Offset Filterzeit* gesetzt. (Die Master-Geschwindigkeit hängt von der Markeranzahl ab; siehe Startkorrektur).

Nun wird die Summe aller Markerabstandsfehler benutzt, um die gefilterte Summe für einen Markerfilter zu berechnen. Danach wird die gefilterte Summe der Fehler von der ungefilterten abgezogen. Dieses Ergebnis wird schließlich benutzt, um die Markerkorrektur zu korrigieren.

Diese bereinigte Korrektur wird in den Korrekturfilter eingeben Das Ergebnis von diesem Korrekturfilter wird gespeichert (plus dem Anteil der Startkorrektur, falls notwendig).

Schließlich wird diese Korrektur über einen Markerabstand gespreizt. Das wird durch Teilen der Korrektur durch die Anzahl der Samples erreicht, die notwendig ist, um einen Markerabstand mit der aktuellen Master-Geschwindigkeit zu passieren. Der Wert wird gespeichert und bei jeder Abtastperiode benutzt, um die berechnete Slave-Position zu korrigieren.

Folgende Einstellungen in Par. 33-28 SYNCMFPAR verändern das Verhalten:

SYNCMFPAR & 4 →	Korrekturzeit, wird statt Par. 33-27 SYNCOFFTIME benutzt.
SYNCMFPAR & 16 →	Es wird keine Korrektur durchgeführt, die den Markerabstand betrifft.

Markerkorrektur SYNCMFTIME == 0

Im ersten Fall, wenn die Markerkorrektur > 0, wird die Korrektur über ein Zeit von (-SYNCVFTIME / 100) ms gespreizt.

Im zweiten Fall wird die Korrektur sofort zur Sollposition addiert.

In jedem Fall wird die Reaktion durch die aktuelle Beschleunigung und Verzögerung begrenzt.

33-30 Maximale Markerkorrektur

SYNCMMAXCORR

Bereich [Unit]

0 – MLONG [qc] ∗ 0

0 = Aus, d.h. keine Begrenzung

Funktion

SYNCMMAXCORR wird benutzt, um die maximale Korrektur, die durch die Markerkorrektur vorgenommen wird, zu begrenzen. Der Wert wird in qc (Slave) eingegeben. Der Befehl arbeitet mit SYNCM und SYNCC.

Nachdem die PFG_G_KORREKTUR berechnet ist, wird PFG_G_KORREST auf das Minimum der G_KORREKTUR oder SYNCMMAXCORR gesetzt. Dieser Wert wird dann für die Korrektur benutzt.

ACHTUNG!:
Wenn man Par. 33-29 *Filterzeit für Markerkorrektur* oder Par. 33-26 *Geschwindigkeitsfilter* (negativ) gesetzt hat, wird die Korrektur abhängig von diesen Faktoren über eine gewisse Zeit gedehnt.

33-31 Synchronisationstyp

SYNCTYPE

Option

*	Standard	[0]
	Look ahead	[1]

Funktion

Die Art, wie die Synchronisation durchgeführt wird, kann geändert werden:

0 = Die aktuelle Master-Position wird mit der künftigen Slave-Position (wo der Slave in 1 ms sein wird) verglichen.

1 = Die aktuelle Master-Position wird mit der aktuellen Sollposition verglichen.

Im Standardfall (SYNCTYPE = 0), wird die Positionsdifferenz ausgeglichen.

Das bedeutet, dass die aktuelle Master-Position (wo der Master jetzt ist) mit der künftigen Slave-Position (wo der Slave in 1 ms sein wird) verglichen wird. So wird immer hinter dem Master hergefahren, solange man nicht INTEGRAL benutzt.

Wenn SYNCTYPE = 1 gewählt ist, vergleicht das System die aktuelle Master-Position mit der aktuellen Sollposition. Das heißt, dass das System versuchen wird, die Differenz der Positionen auf Null zu bringen, egal wie PID gesetzt ist.

ACHTUNG!:
Vergegenwärtigen Sie sich, dass SYNCERR auf eine aktuelle (neue) Master-Sollposition angleicht, minus der Istposition des Slaves plus unerledigter Filterfehler und Korrekturen.

□ 33-4* Grenzwertbehandlung

Parameter für die Bestimmung des Verhaltens der Endschalter.

33-40 Verhalten bei Endschalter

ENDSWMOD

Option

*	Fehler-Unterprogramm aufrufen	[0]
	Kontrollierter Stopp	[1]

Funktion

Dieser Parameter gibt an, wie sich die Steuerung bei Erreichen des positiven oder negativen Hardware-Endschalters verhalten soll.

Verhalten im Fehlerfall siehe Par. 33-83 ERRCOND.

33-41 Negative Software-Wegbegrenzung

NEGLIMIT

Bereich [Unit]

-MLONG – MLONG [qc]	* -500000

Funktion

NEGLIMIT gibt die negative Wegbegrenzung für alle Fahrbewegungen an. Wird dieser Wert überschritten, wird ein Fehler ausgelöst. NEGLIMIT ist nur aktiv, wenn Par. 33-43 SWNEGLIMACT gesetzt ist.

Ein Positionierbefehl, der außerhalb der eingestellten Grenzen liegt, wird nicht ausgeführt.

ACHTUNG!:
Bei der Verwendung des Befehls DEF ORIGIN wird die Wegbegrenzung automatisch angepasst, sodass die ursprüngliche Lage des Verfahrbereichs erhalten bleibt.

ACHTUNG!:
Die Wegbegrenzung wird immer in Quadcounts angegeben.

33-42 Positive Software-Wegbegrenzung

POSLIMIT

Bereich [Unit]

-MLONG – MLONG [qc]	* 500000

Funktion

POSLIMIT gibt die positive Wegbegrenzung für alle Fahrbewegungen an. Wird dieser Wert überschritten, wird ein Fehler ausgelöst.

POSLIMIT ist nur aktiv, wenn Par. 33-44 SWPOSLIMACT gesetzt ist. Ein Positionierbefehl, der außerhalb der eingestellten Grenzen liegt, wird nicht ausgeführt.

ACHTUNG!:
Bei der Verwendung des Befehls DEF ORIGIN wird die Wegbegrenzung automatisch angepasst, so dass die ursprüngliche Lage des Verfahrbereichs erhalten bleibt.

ACHTUNG!:
Die Wegbegrenzung wird immer in Quadcounts angegeben.

33-43 Negative SW-Wegbegrenzung aktiv

SWNEGLIMACT

Option

*	Inaktiv	[0]
	Aktiv	[1]

Funktion

Durch Setzen dieses Parameters auf [1] wird die Überwachung der *Negativen Wegbegrenzung* im FC 300 aktiviert. Dann wird bei jeder Bewegung überprüft, ob die Zielposition außerhalb des zulässigen Verfahrbereichs liegt. Wenn dies der Fall ist, wird eine Fehlermeldung ausgelöst und die Antriebsregelung abgeschaltet.

Im Positioniermodus bedeutet dies, dass der entsprechende Positioniervorgang nicht gestartet wird und der Fehler durch einen ERRCLR Befehl behoben werden kann.

Im Synchronisations- und Drehzahlmodus kann der Fehler erst beim Überfahren der Wegbegrenzung erkannt werden, wodurch sich der Antrieb beim Auftreten der Fehlermeldung bereits außerhalb des zulässigen Verfahrbereichs befindet. In diesem Fall müssen Sie den Antrieb von Hand wieder in den zulässigen Bereich zurück bewegen und den Fehler löschen, oder im Menü *Steuerung* → *Parameter* → *Achsen* vorübergehend die entsprechende *Software-Wegbegrenzung* abschalten und dann den Fehler löschen.

33-44 Positive SW-Wegbegrenzung aktiv

SWPOSLIMACT

Option

*	Inaktiv	[0]
	Aktiv	[1]

Funktion

Durch Setzen dieses Parameters auf „1" wird die Überwachung der *Positiven Software-Wegbegrenzung* in der Steuerung aktiviert. Dann wird bei jeder Bewegung überprüft, ob die Zielposition außerhalb des zulässigen Verfahrbereichs liegt und im gegebenen Fall eine Fehlermeldung ausgelöst und die Antriebsregelung abgeschaltet.

Im Positioniermodus bedeutet dies, dass der entsprechende Positioniervorgang nicht gestartet wird und der Fehler durch einen ERRCLR Befehl behoben werden kann.

Im Synchronisations- und Drehzahlmodus kann der Fehler erst beim Überfahren der Begrenzung erkannt werden, wodurch der Antrieb beim Auftreten der Fehlermeldung bereits außerhalb des zulässigen Verfahrbereichs ist. Dann müssen Sie den Antrieb von Hand wieder in den zulässigen Bereich zurück bewegen und den Fehler löschen, oder im Menü *Steuerung* → *Parameter* → *Achsen* vorübergehend die entsprechende *Software-Wegbegrenzung* abschalten und den Fehler löschen.

33-45 Messzeit im Zielfenster

TESTTIM

Bereich [Unit]

0 – 10 [ms]	* 0

Funktion

Nach dem Erreichen des Zielfensters wird zweimal die Istposition gemessen und mit dem Par. 33-46 *Zielfenster-Grenzwert* verglichen. Ist das Ergebnis kleiner als dieser Grenzwert, gilt die Position als erreicht, andernfalls wird erneut gemessen. TESTTIM gibt den Zeitabstand zwischen diesen beiden Messungen an.

ACHTUNG!:
Die Einschränkung auf 10 ms ist dadurch begründet, dass die Funktion 'diffval' wirklich wartet und solange auch keine Endschalter- und Schleppfehler-Überwachung aktiv ist. Deswegen sollte diese Zeit nicht zu lange sein.

33-46 Zielfenster-Grenzwert

TESTVAL

Bereich [Einheit]

1 – 10000 [qc]	* 1

Funktion

Nachdem das Zielfenster erreicht ist, wird mit dem in Par. 33-45 TESTTIM festgelegten Abstand zweimal die Position ausgelesen und der Abstand mit diesem *Zielfenster-Grenzwert* verglichen.

Das Ergebnis entscheidet, ob die Position als erreicht gilt oder nicht.

ACHTUNG!:
Bei größeren Zeitabständen muss berücksichtigt werden, dass das Erreichen der Zielposition auf jeden Fall um diese Zeit verzögert wird.

33-47 Zielfenster-Größe

TESTWIN

Bereich [Einheit]

0 – 10000 [qc] ∗ 0
0 = Off

Die *Zielfenster-Größe* muss immer kleiner als Par. 33-46 *Zielfenster-Grenzwert* sein.

Funktion

TESTWIN gibt die Größe des Zielfensters an. Eine Position gilt erst dann als erreicht, wenn die Sollfahrt (Trapez) abgearbeitet ist, die Istposition innerhalb des Fensters liegt und die Geschwindigkeit kleiner als Par. 33-46 *Zielfenster-Grenzwert* ist. (Voraussetzung: TESTWIN und TESTTIM sind aktiviert.) Hierbei ist die Geschwindigkeit TESTVAL in qc/TESTTIM angegeben.

Die Steuerung wartet mit dem Ausführen des jeweils nächsten Befehls, bis die Istposition innerhalb des Zielfensters liegt.

Wenn TESTWIN nicht aktiviert ist [0], gilt das Ziel als erreicht, sobald die Sollposition gleich der Zielposition ist. Diese muss jedoch nicht mit der tatsächlichen Position des Antriebs übereinstimmen.

ACHTUNG!:
Wird das Zielfenster um die Endposition zu klein gewählt, könnte sich der Antrieb in einer sehr kleinen Umgebung um die Endposition bewegen, ohne das Zielfenster zu erreichen, so dass das Programm bei dem entsprechenden Positionierbefehl „hängen" bleibt.

Zielfenster [0] deaktiviert die Überwachung der Istposition und überwacht lediglich die Sollposition.

□ 33-5* I/O Konfiguration

Es gibt je einen Parameter für die Eingänge und Ausgänge. Mit diesem Parameter wird jedem Eingang und Ausgang eine Funktion zugeordnet.

Funktionen der digitalen Eingänge

	Funktionen der digitalen Eingänge	Auswahl	Klemme
∗	Keine Funktion	[0]	X57, X59/7,8*
	Home-Referenzschalter NO	[1]	X57, X59/7,8*
	Home-Referenzschalter NC	[2]	X57, X59/7,8*
	Negativer Endschalter NO	[3]	X57, X59/7,8*
	Negativer Endschalter NC	[4]	X57, X59/7,8*
	Positiver Endschalter NO	[5]	X57, X59/7,8*
	Positiver Endschalter NC	[6]	X57, X59/7,8*
	Fehler löschen NO	[7]	X57, X59/7,8*
	Fehler löschen NC	[8]	X57, X59/7,8*
	Programm abbrechen NO	[9]	X57, X59/7,8*
	Programm abbrechen NC	[10]	X57, X59/7,8*
	Programm fortsetzen NO	[11]	X57, X59/7,8*
	Programm fortsetzen NC	[12]	X57, X59/7,8*
	Programm starten NO	[13]	X57, X59/7,8*
	Programm starten NC	[14]	X57, X59/7,8*
	Programmwahl	[15]	X57, X59/7,8*

X57 = alle

*) X59/7,8 nur wenn Par. 33-60 IOMODE auf [0] gesetzt ist.

Sie können alle digitalen Eingänge 1 – 10 (12) mit diesen Funktionen programmieren:

- **Keine Funktion [0]:** Keine Reaktion auf Signale vom Eingang_n.
- **Home-Referenzschalter NO [1]:** Definiert den digitalen Eingang_n der MCO 305 als Home-Referenzschalter. *Homefahrt-Verhalten* bei Erreichen des Schalters siehe Par. 33-04.
- **Home-Referenzschalter NC [2]:** Definiert den digitalen Eingang_n als inversen Home-Referenzschalter. *Homefahrt-Verhalten* bei Erreichen des Schalters siehe Par. 33-04.
- **Negativer Endschalter NO [3]:** Definiert den digitalen Eingang_n als negativen Endschalter.
- **Negative Endschalter NC [4]:** Definiert den digitalen Eingang_n als inversen negativen Endschalter.
- **Positiver Endschalter NO [5]:** Definiert den digitalen Eingang_n als positiven Endschalter.
- **Positiver Endschalter NC [6]:** Definiert den digitalen Eingang_n als inversen positiven Endschalter.

- **Fehler löschen NO [7]:** Definiert den digitalen Eingang_n, der zum Fehler löschen benutzt wird.
- **Fehler löschen NC [8]:** Definiert den digitalen Eingang_n, der zum Fehler löschen benutzt wird.
- **Programmausführung abbrechen NO [9]:** Definiert den digitalen Eingang_n, der benutzt wird um zu reagieren und ein Programm sofort abzubrechen, sobald er aktiviert wird. Ein solches Programm kann mit CONTINUE fortgesetzt werden.
- **Programmausführung abbrechen NC [10]:** Definiert den digitalen Eingang_n, der benutzt wird um zu reagieren und ein Programm sofort abzubrechen, sobald er aktiviert wird. Ein solches Programm kann mit CONTINUE fortgesetzt werden.
- **Programmausführung fortsetzen NO [11]:** Definiert den digitalen Eingang_n der zum Fortsetzen von abgebrochenen Programmen benutzt wird.
- **Programmausführung fortsetzen NC [12]:** Definiert den digitalen Eingang_n der zum Fortsetzen von abgebrochenen Programmen benutzt wird.
- **Programmausführung starten NO [13]:** Definiert den digitalen Eingang_n der benutzt wird, um den Typ des Programmstarts festzulegen.
  Wenn ein Eingang_n auf [13] gesetzt ist, dann wird zuerst das *Autostart*-Programm ausgeführt und danach gewartet bis der Eingang_n aktiv ist. Dieser wird entsprechend der Programmwahl ausgewertet, um die Nummer des Programms zu bestimmen, das ausgeführt werden soll.
  Wenn kein Eingang für Programmstart gesetzt ist, wird wieder das mit Autokennung versehene Programm gestartet.
- **Programmausführung starten NC [14]:** Definiert den digitalen Eingang_n der benutzt wird, um den Typ des Programmstarts festzulegen.
  Wenn Eingang_n auf [14], gesetzt ist, dann wird zuerst das *Autostart*-Programm ausgeführt und danach gewartet bis der Eingang_n aktiv ist. Dieser wird entsprechend der Programmwahl ausgewertet, um die Programmnummer zu bestimmen, die ausgeführt werden soll.
  Wenn kein Eingang für Programmstart gesetzt ist, wird wieder das mit Autokennung versehene Programm gestartet.
- **Programmwahl [15]:** Definiert den Eingang_n, der für die Programmwahl benutzt wird. Wenn Eingang_n auf [15] gesetzt ist, gibt dieser Parameter die Eingangsnummer an, ab der die Eingänge für die Programmwahl verwendet werden. Dazu gehören alle bis zu I_FUNCTION_14.

  Beispiel

  Wenn I_FUNCTION_3_15 und I_FUNCTION_7_13, werden bei der Aktivierung von Eingang 7 die Eingänge 3, 4, 5, 6 binär ausgewertet und das Ergebnis als Programmnummer verwendet.

Eingang	Level	Binärwert
3	low	0
4	high	2
5	high	2^2
6	low	0
=> zu startendes Programm:		6

  Maximal kann somit zwischen 90 Programmen, die mit den Nummern 0 bis 89 gekennzeichnet sind, ausgewählt werden.

ACHTUNG!:
Die Werte werden nicht automatisch zurückgesetzt, wenn ein neuer Eingang bestimmt wird. Der Anwender muss selbst darauf achten. Das bedeutet, wenn I_FUNCTION_1 auf [1] gesetzt ist (d.h. Eingang 1 ist der Referenzschalter) und der Anwender setzt I_FUNCTION_3 auf [1] (d.h. Eingang 3 ist der Referenzschalter) dann sind zwei Eingänge als Referenzschalter definiert. Die Software nimmt aber immer den ersten und ignoriert den zweiten.

33-50 Klemme X57/1 Digitaler Eingang

I_FUNCTION_1

∗ Keine Funktion [0]

Funktion

Definiert die Funktion des digitalen Eingangs 1 der MCO 305.

33-51 Klemme X57/2 Digitaler Eingang

I_FUNCTION_2

∗ Keine Funktion [0]

Funktion

Definiert die Funktion des digitalen Eingangs 2 der MCO 305.

33-52 Klemme X57/3 Digitaler Eingang

I_FUNCTION_3

∗ Keine Funktion [0]

Funktion

Definiert die Funktion des digitalen Eingangs 3 der MCO 305.

33-53 Klemme X57/4 Digitaler Eingang

I_FUNCTION_4

∗ Keine Funktion [0]

Funktion

Definiert die Funktion des digitalen Eingangs 4 der MCO 305.

33-54 Klemme X57/5 Digitaler Eingang

I_FUNCTION_5

∗ Keine Funktion [0]

Funktion

Definiert die Funktion des digitalen Eingangs 5 der MCO 305.

33-55 Klemme X57/6 Digitaler Eingang

I_FUNCTION_6

∗ Keine Funktion [0]

Funktion

Definiert die Funktion des digitalen Eingangs 6 der MCO 305.

33-56 Klemme X57/7 Digitaler Eingang

I_FUNCTION_7

∗ Keine Funktion [0]

Funktion

Definiert die Funktion des digitalen Eingangs 7 der MCO 305.

33-57 Klemme X57/8 Digitaler Eingang

I_FUNCTION_8

∗ Keine Funktion [0]

Funktion

Definiert die Funktion des digitalen Eingangs 8 der MCO 305.

33-58 Klemme X57/9 Digitaler Eingang

I_FUNCTION_9

∗ Keine Funktion [0]

Funktion

Definiert die Funktion des digitalen Eingangs 9 der MCO 305.

33-59 Klemme X57/10 Digitaler Eingang

I_FUNCTION_10

∗ Keine Funktion [0]

Funktion

Definiert die Funktion des digitalen Eingangs 10 der MCO 305.

33-60 Klemme X59/1 und X59/2 Modus

IOMODE

Option

Eingang [0]
X59/1 = Eingang 11
X59/2 = Eingang 12

∗ Ausgang [1]
X59/1 = Ausgang 1
X59/2 = Ausgang 2

Funktion

Zwei der Klemmen (X59/1 und X59/2) können als digitale Eingänge oder Ausgänge konfiguriert werden.

33-61 Klemme X59/1 Digitaler Eingang

I_FUNCTION_11

∗ Keine Funktion [0]

Funktion

Definiert die Funktion des digitalen Eingangs 11 der MCO 305.

ANMERKUNG: Dieser Parameter wird nur dargestellt, wenn Par. 33-60 IOMODE = [0], d.h. Eingang_11 im Standardmodus benutzt wird.

33-62 Klemme X59/2 Digitaler Eingang

I_FUNCTION_12

∗ Keine Funktion [0]

Funktion

Definiert die Funktion des digitalen Eingangs 12 der MCO 305.

ANMERKUNG: Dieser Parameter wird nur dargestellt, wenn Par. 33-60 IOMODE = [0], d.h. Eingang_12 im Standardmodus benutzt wird.

Funktionen der digitalen Ausgänge

Funktionen der digitalen Ausgänge		Auswahl	Klemme
*	Keine Funktion	[0]	X59
	Fahrbefehl aktiv NO	[1]	X59
	Fahrbefehl aktiv NC	[2]	X59
	Fehler NO	[3]	X59
	Fehler NC	[4]	X59
	Bremssteuerung NO	[5]	X59
	Bremssteuerung NC	[6]	X59

ANMERKUNG: 8 Ausgänge sind nur verfügbar, wenn Par. 33-60 IOMODE auf [1] gesetzt ist.

Sie können alle digitalen Ausgänge 1 – 8 (6) mit folgenden Funktionen programmieren:

- **Keine Funktion [0]:** Keine Reaktion auf Signale vom Ausgang_n.
- **Fahrbefehl aktiv NO [1]:** Definiert den digitalen Ausgang_n der MCO 305 für *Fahrbefehl aktiv*. Der Ausgang ist immer aktiviert (24 V) sobald ein Fahrbefehl aktiv ist, unabhängig in welchem Modus (Positions-, Geschwindigkeits- oder Synchronisationsbefehl).
  Diese Funktion eignet sich nicht für die Motorüberwachung, denn der Motor könnte stillstehen, obwohl die Steuerung in Bewegung ist.
- **Fahrbefehl aktiv NC [2]:** Definiert den digitalen Ausgang_n für *Fahrbefehl aktiv*. Der Ausgang ist immer aktiviert (0 V) sobald ein Fahrbefehl aktiv ist, unabhängig in welchem Modus (Positions-, Geschwindigkeits- oder Synchronisationsbefehl).
  Diese Funktion eignet sich nicht für die Motorüberwachung, denn der Motor könnte stillstehen, obwohl die Steuerung in Bewegung ist.
- **Fehler NO [3]:** Definiert den Ausgang_n für Fehler. Der Ausgang wird gesetzt (24 V), wenn ein Fehler aufgetreten ist. Sobald der Fehler gelöscht ist, wird er wieder zurückgesetzt.

ACHTUNG!:
Die Einstellung des Parameters hat keinen Einfluss auf die Verwendung der Befehle OUT und OUTB.

Mit diesen Befehlen können auch die Ausgänge verändert werden, die vordefinierte Funktionen besitzen.

- **Fehler NC [4]:** Definiert den Ausgang_n für Fehler. Der Ausgang wird gesetzt (0 V), wenn ein Fehler aufgetreten ist. Sobald der Fehler gelöscht ist, wird er wieder zurückgesetzt.

ACHTUNG!:
Die Parametereinstellung hat keinen Einfluss auf die Verwendung der Befehle OUT und OUTB. Mit diesen Befehlen können auch die Ausgänge verändert werden, die vordefinierte Funktionen besitzen.

- **Bremssteuerung NO [5]:** Definiert den digitalen Ausgang_n für die Bremse. Wenn ein Ausgang für die Bremse definiert ist, bleibt diese aktiv, sogar wenn das Programm mit [Esc] abgebrochen wurde.
  Der Ausgang wird bei einem Abbruch oder einem Fehler der Optionskarte aktiviert (24 V), falls der Par. 33-83 ERRCOND auf [1] oder [3] gesetzt ist.

ACHTUNG!:
Der Ausgang Bremse muss immer durch einen OUT Befehl im Programm zurückgesetzt werden.

Beispiel

```
ON ERROR GOSUB err_handle
// p. 33-66 O4 ist auf [6] gesetzt
SET ERRCOND 1
// Hauptprogramm
SUBPROG err_handle
  WAITI 1
  ERRCLR
  OUT 4 1
RETURN
```

- **Bremssteuerung NC [6]:** Definiert den digitalen Ausgang_n für die Bremse. Wenn ein Ausgang für die Bremse definiert ist, bleibt diese aktiv, sogar wenn das Programm mit [Esc] abgebrochen wurde.
  Der Ausgang wird bei einem Abbruch oder einem Fehler der Optionskarte aktiviert (0 V), falls der Par. 33-83 ERRCOND auf [1] oder [3] gesetzt ist.

ACHTUNG!:
Der Ausgang Bremse muss immer durch einen OUT Befehl im Programm zurückgesetzt werden.

33-63 Klemme X59/1 Digitaler Ausgang

O_FUNCTION_1

∗ Keine Funktion [0]

Funktion

Definiert die Funktion des digitalen Ausgangs 1 der MCO 305.

ANMERKUNG: Dieser Parameter wird nur dargestellt, wenn Par. 33-60 IOMODE = 1, d.h. Ausgang 1 wird nicht im Standardmodus benutzt.

33-64 Klemme X59/2 Digitaler Ausgang

O_FUNCTION_2

∗ Keine Funktion [0]

Funktion

Definiert die Funktion des digitalen Ausgangs 2 der MCO 305.

ANMERKUNG: Dieser Parameter wird nur dargestellt, wenn Par. 33-60 IOMODE = 1, d.h. Ausgang 2 wird nicht im Standardmodus benutzt.

33-65 Klemme X59/3 Digitaler Ausgang

O_FUNCTION_3

∗ Keine Funktion [0]

Funktion

Definiert die Funktion des digitalen Ausgangs 3 der MCO 305.

33-66 Klemme X59/4 Digitaler Ausgang

O_FUNCTION_4

∗ Keine Funktion [0]

Funktion

Definiert die Funktion des digitalen Ausgangs 4 der MCO 305.

33-67 Klemme X59/5 Digitaler Ausgang

O_FUNCTION_5

∗ Keine Funktion [0]

Funktion

Definiert die Funktion des digitalen Ausgangs 5 der MCO 305.

33-68 Klemme X59/6 Digitaler Ausgang

O_FUNCTION_6

∗ Keine Funktion [0]

Funktion

Definiert die Funktion des digitalen Ausgangs 6 der MCO 305.

33-69 Klemme X59/7 Digitaler Ausgang

O_FUNCTION_7

∗ Keine Funktion [0]

Funktion

Definiert die Funktion des digitalen Ausgangs 7 der MCO 305.

33-70 Klemme X59/8 Digitaler Ausgang

O_FUNCTION_8

∗ Keine Funktion [0]

Funktion

Definiert die Funktion des digitalen Ausgangs 8 der MCO 305.

□ 33-8* Globale Parameter

33-80 Aktivierte Programmnummer

PRGPAR

Bereich

-1 – 127 ∗ -1

–1 = Programmnummer ist nicht aktiviert, d. h. es wird nach AutoExec kein Programm gestartet.

0 – 127 = Aktivierte Programmnummer (und AutoExec) werden nach dem Einschalten gestartet.

Der *Einschaltstatus* wird mit Par. 33-81 festgelegt.

Funktion

Mit dem PRGPAR können Sie festlegen, welches Programm nach Ablauf eines per *Autostart* (Autokennung) ausgeführten Programms gestartet werden soll. Dieser Parameter kann auch von Programmen oder per Display geändert und gespeichert werden.

Wenn keine Programmnummer aktiviert ist und auch in Par. I_FUNCTION_n [13] oder [14] kein Eingang für einen Programmstart gesetzt ist, wird wieder das mit Autokennung versehene Programm gestartet.

ACHTUNG!:
Wenn kein Autostart-Programm definiert ist, kann auch kein Programm über diesen Parameter gestartet werden, denn dies erfordert immer ein beendetes Autostart-Programm.

33-81 Einschaltstatus

Power-up Status

Option

	Motor aus	[0]
*	Motor an	[1]

Funktion

Der Status der Steuerung nach dem Einschalten kann mit diesem Parameter festgelegt werden.

Wählen Sie *Motor aus* [0] wenn der Motor nach dem Einschalten ungesteuert (FC 300 im Leerlauf) bleiben muss. FC 300 und Positionierregelung müssen mit dem MOTOR ON Befehl abgeschaltet sein, bevor die Bewegung gestartet werden kann.

Wählen Sie *Motor an* [1] wenn der Motor nach dem Einschalten gesteuert werden muss, die Positionierregelung aktiv ist und auf der aktuellen Position bleibt, bis ein anderer Befehl gegeben wird.

33-82 Statusüberwachung Antrieb

STATUSMONITORING

Option

	Aus	[0]
*	Ein	[1]

Funktion

Aus- und einschalten der Überwachung des FC 300 Status während die Positionierregelung der MCO 305 aktiv ist.

Wählen Sie *Aus* [0] wenn die Überwachung abgeschaltet sein muss, d. h. MCO 305 wird versuchen den Motor unabhängig vom FC 300 Status zu regeln. Wenn versucht wird, eine Bewegung zu starten, solange der FC 300 nicht freigegeben ist, wird das normalerweise zu einem Schleppfehler führen (Fehler 108).

Wählen Sie *Ein* [1] wenn die Überwachung eingeschaltet sein muss. Fehler 113 wird aktiviert, falls der FC 300 nicht freigegeben ist (z.B. Trip) während MCO 305 im MOTOR ON Status ist (Positionierregelung).

33-83 Verhalten im Fehlerfall

ERRCOND

Option

*	Freilauf	[0]
	Freilauf und Bremse	[1]
	Geregelter Stopp	[2]
	Geregelter Stopp mit Bremse	[3]
	Ohne automatisches MOTOR OFF in die Fehlerroutine springen	[5]

Funktion

0 = Standard, d.h. Antrieb geht in Freilauf, der Regelkreis wird unterbrochen.

1 = Wie [0], aber Ausgang Bremse (falls definiert) wird aktiviert.

2 = Motorstopp mit max. Verzögerung (Stopprampe), anschließend stillstandgeregelt.

3 = Wie [2], zusätzlich wird der Ausgang Bremse aktiviert (falls definiert), aber erst nach MOTOR STOP.
Alle anderen Aktivitäten wie MOTOR OFF und ähnliche müssen im ON_ERROR Unterprogramm gesetzt werden.

5 = Es wird in das Fehlerunterprogramm gesprungen, die Regelung aber nicht automatisch abgeschaltet. Dies kann bzw. muss bei Bedarf durch das Anwendungsprogramm mit einem MOTOR OFF Befehl im Fehlerunterprogramm ausgelöst werden.

ACHTUNG!:
In den Parametern 33-63 bis 33-70, O_FUNCTION_n muss mit Option 5 oder 6 ein Ausgang Bremse definiert sein.

33-84 Verhalten bei Programmabbruch

ESCCOND

Option

*	Geregelter Stopp	[0]
	Geregelter Stopp + Ausgänge = 0	[1]
	Geregelter Stopp + Ausgänge = 1	[2]

Funktion

ESCCOND definiert wie der FC300 bei einem Programmabbruch mit [Esc] reagieren soll.

0 = Der Motor wird mit maximaler Verzögerung gestoppt, die Ausgang Bremse wird aktiviert (falls definiert), die Master-Simulation wird gestoppt.
Die Ausgänge bleiben im aktuellen Status.

1 = Wie [0], aber alle Ausgänge inklusive des FC 300 Ausgangs (falls durch MCO 305 gesteuert) werden auf [0] gesetzt.
Ausnahme: Der Ausgang Bremse wird – falls definiert – immer aktiviert.

2 = Wie [0], aber alle Ausgänge inklusive des FC 300 Ausgangs (falls durch MCO 305 gesteuert) werden auf [1] gesetzt.
Ausnahme: Der Ausgang Bremse wird – falls definiert – immer aktiviert.

33-85 Externe 24VDC MCO Versorgung

EXTERNAL24V

Option

*	Nein	[0]
	Ja	[1]

Funktion

Definiert ob eine externe 24 V Versorgung angeschlossen ist oder nicht.

□ MCO Datenanzeigen

Um das Lesen und Schreiben der PCD[] Arrays zu unterstützen und weiterhin in Übereinstimmung mit dem ProfiDrive Profil zu bleiben gibt es folgende 20 Parameter in der 34-0* und 34-2* Gruppe:

□ 34-0* PCD Schreib-Parameter

34-01...10 PCD n nach MCO schreiben

n = 1 – 10
- P. 34-01 = PCD 1 nach MCO schreiben
- P. 34-02 = PCD 2 nach MCO schreiben
- P. 34-03 = PCD 3 nach MCO schreiben
- P. 34-04 = PCD 4 nach MCO schreiben
- P. 34-05 = PCD 5 nach MCO schreiben
- P. 34-06 = PCD 6 nach MCO schreiben
- P. 34-07 = PCD 7 nach MCO schreiben
- P. 34-08 = PCD 8 nach MCO schreiben
- P. 34-09 = PCD 9 nach MCO schreiben
- P. 34-10 = PCD 10 nach MCO schreiben

Funktion

Alle 10 Parameter sind als Displayzeilen-Parameter in Par. 0-20 bis 0-24 der LCP-Bedieneinheit wählbar.

Mit Index [n] kann nur „MCO PCD[n] Schreiben" ausgewählt werden. Diese Auswahl definiert die entsprechenden Subindizes, die von der MCO 305 verarbeitet werden. Das macht es auch möglich die Indizes 0 und 1 (STW/ZSW und Sollwert/HIW) auf MCO zu setzen. Allerdings könnte sich dies mit dem ProfiDrive Profil widersprechen.

□ 34-2* PCD Lese-Parameter

34-21...31 PCD n von MCO lesen

n = 1 – 10
- P. 34-21 = PCD 1 von MCO lesen
- P. 34-22 = PCD 2 von MCO lesen
- P. 34-23 = PCD 3 von MCO lesen
- P. 34-24 = PCD 4 von MCO lesen
- P. 34-25 = PCD 5 von MCO lesen
- P. 34-26 = PCD 6 von MCO lesen
- P. 34-27 = PCD 7 von MCO lesen
- P. 34-28 = PCD 8 von MCO lesen
- P. 34-29 = PCD 9 von MCO lesen
- P. 34-30 = PCD 10 von MCO lesen

Funktion

Alle 10 Lese-Parameter sind als Displayzeilen-Parameter in Par. 0-20 bis 0-24 wählbar. Aber als Index [n] kann nur „MCO PCD[n] Lesen" ausgewählt werden. Diese Auswahl legt fest, dass die entsprechenden Sub-Indizes von MCO 305 erzeugt werden.

☐ 34-4* Eingänge & Ausgänge

34-40 Digitale Eingänge

Funktion

Lesen des Status der digitalen Eingänge.

34-41 Digitale Ausgänge

Funktion

Lesen des Status der digitalen Ausgänge.

☐ 34-5* Prozessdaten

In den meisten Standardanwendungen können die 34-xx Displayparameter, die automatisch gehandhabt werden, statt des LINKSYSVAR Befehls genutzt werden.

34-50 Istposition

Funktion

Aktuelle Slave-Position in BE; entspricht dem APOS Befehl.

34-51 Sollposition

Funktion

Slave-Position in BE setzen; entspricht dem CPOS Befehl.

34-52 Istposition Master

Funktion

Aktuelle Master-Position in qc; entspricht dem MAPOS Befehl.

34-53 Indexposition Slave

Funktion

Letzte Indexposition des Slaves in BE; entspricht dem IPOS Befehl.

34-54 Indexposition Master

Funktion

Letzte Indexposition des Masters in qc; entspricht dem MIPOS Befehl.

34-55 Kurvenposition

Funktion

Slave-Kurvenposition, die der aktuellen Master-Position der Kurve entspricht, abfragen; entspricht dem CURVEPOS Befehl.

34-56 Schleppfehler

Funktion

Aktuellen Schleppfehler einer Achse in BE abfragen (mit Berücksichtigung des Vorzeichens); entspricht dem TRACKERR Befehl.

34-57 Synchronisationsfehler

Funktion

Aktuellen Synchronisationsfehler des Slaves abfragen. Das ist der Abstand zwischen der aktuellen Master-Position (umgerechnet mit Getriebefaktor und Offset) und der Istposition des Slaves. Das Ergebnis wird in BE angezeigt und

a) als absoluter Wert, wenn der Wert des Genauigkeitsfensters in Par. 33-13 SYNCACCURACY mit einem positiven Vorzeichen definiert ist;

b) mit Vorzeichen, wenn in Par. 33-13 der Wert des Genauigkeitsfensters mit einem negativen Vorzeichen definiert ist.

Der Parameter entspricht dem SYNCERR Befehl.

34-58 Aktuelle Geschwindigkeit

Funktion

Aktuelle Geschwindigkeit in BE/s; entspricht AVEL.

34-59 Aktuelle Master-Geschwindigkeit

Funktion

Aktuelle Master-Geschwindigkeit in qc/s; entspricht dem MAVEL Befehl.

34-60 Synchronisationsstatus

Funktion

Flag, um den Synchronisationsstatus abzufragen. Der Parameter entspricht dem SYNCSTAT Befehl.

34-61 Achsstatus

Funktion

Informiert über den Status der Programmausführung. Der Parameter entspricht dem AXEND Befehl.

34-62 Programmstatus

Funktion

Zeigt den Status der Achse und der Steuerung in 4-Byte-Werten. Dies entspricht dem STAT Befehl.

□ 34-7* Diagnoseanzeigen

Parameter zum Auslesen der MCO Warnmeldungen.

34-70 MCO Alarmwort 1

Funktion

Zeigt das MCO 305 Alarmwort zum Auslesen von MCO Fehler in MCT 10.

Par. 34-70 kann nicht ausgelesen werden, wenn der Motor läuft.

Bit	Hex	Dezimal	Bedeutung
0	00000001	1	FC nicht freigegeben
1	00000002	2	Fehler nicht zurück-gesetzt
2	00000004	4	HOME nicht ausgeführt
3	00000008	8	Schleppfehler
4	00000010	16	Index nicht gefunden
5	00000020	32	Hardware-Endschalter überschritten
6	00000040	64	Software-Endschalter überschritten
7	00000080	128	Keine externe 24 V
8	00000100	256	Digitaler Ausgang Überlast
9	00000200	512	Drehgeberfehler
10	00000400	1024	Speicherfehler
11	00000800	2048	Parameterspeicher defekt
12	00001000	4096	Programmspeicher defekt
13	00002000	8192	Reset durch CPU
14	00004000	16384	WAITNDX Timeout
15	00008000	32768	Interner MCO Fehler
16	00010000	65536	Homefahrt-Geschwindigkeit Null
..	..	..	nicht benutzt
31	80000000	2147483648	MCO Alarmwort 2

34-71 MCO Alarmwort 2

Funktion

Zeigt das MCO 305 Alarmwort zum Auslesen von MCO Fehler in MCT 10.

Par. 34-71 kann nicht ausgelesen werden, wenn der Motor läuft.

Bit	Hex	Dezimal	Bedeutung
0	00000001	1	Achse nicht vorhanden
1	00000002	2	Unbekannter Befehl
2	00000004	4	Unbekannter Parameter
3	00000008	8	Zu viele LOOP Befehle
4	00000010	16	Zu viele Interrupts
5	00000020	32	Zu viele GOSUB
6	00000040	64	Zu viele RETURN
7	00000080	128	Abbruch durch Benutzer
8	00000100	256	LINK fehlgeschlagen
9	00000200	512	Falsche Array-Größe (DIM)
10	00000400	1024	Array zu klein
11	00000800	2048	Zu viele Zeit-Interrupts
12	00001000	4096	Platz im Speicher reicht nicht aus
13	00002000	8192	Programmspeicher ist schreibgeschützt
14	00004000	16384	CAM-Array falsch
15	00008000	32768	Parameter speichern fehlgeschlagen

.

□ Parameterlisten

Die Parameter werden durch Parameternummern bestimmt. Orientieren Sie sich am besten zuerst in der Übersicht; dann finden Sie die Detail-Informationen ganz schnell anhand der Parameternummer.

Ändern während des Betriebs

„TRUE" (WAHR) bedeutet, dass der Parameter während des Betriebs des Frequenzumrichters geändert werden kann;

„FALSE" (FALSCH) bedeutet, dass er gestoppt werden muss, um Änderungen vorzunehmen.

4-Set-up (4-Parameter-Sätze)

„1-Set-up" (1 Parametersatz): Der Datenwert ist derselbe in allen Parametersätzen.

Umwandlungsindex

Diese Zahl bezieht sich auf eine Umrechnungszahl, die beim Schreiben oder Lesen mit einem Frequenzumrichter benutzt wird.

Schlagen Sie bitte alle anderen Umrechnungs-faktoren im FC 300 Produkthandbuch nach.

Umrechnungsindex	0
Umrechnungsfaktor	1

Datentyp

Schlagen Sie bitte alle andern Datentypen im FC 300 Produkthandbuch nach.

Datentyp	**Beschreibung**	**Typ**
2	Integer 8	Int8
3	Integer 16	Int16
4	Integer 32	Int32
5	Unsigned 8	Uint8
6	Unsigned 16	Uint16
7	Unsigned 32	Uint32

□ Anwendungsparameter, Parameter Liste

Par. Nr. #	Parametername	Parameterbeschreibung	Werkseinstellung	Ändern während des Betriebs	4-Par.-sätze	Umwandlungs-index	Typ
19-0*	**Anwendungsparameter**						
19-00		Anwendungsparameter	0	TRUE			
…				TRUE			
19-89		Anwendungsparameter	0	TRUE			
19-9*	**Nur-Lesen Anwendungsparameter**						
19-90		Anwendungsparameter 90	0	Nur-Lesen	1-set-up	0	Int32
19-91		Anwendungsparameter 91	0	Nur-Lesen	1-set-up	0	Int32
19-92		Anwendungsparameter 92	0	Nur-Lesen	1-set-up	0	Int32
19-93		Anwendungsparameter 93	0	Nur-Lesen	1-set-up	0	Int32
19-94		Anwendungsparameter 94	0	Nur-Lesen	1-set-up	0	Int32
19-95		Anwendungsparameter 95	0	Nur-Lesen	1-set-up	0	Int32
19-96		Anwendungsparameter 96	0	Nur-Lesen	1-set-up	0	Int32
19-97		Anwendungsparameter 97	0	Nur-Lesen	1-set-up	0	Int32
19-98		Anwendungsparameter 98	0	Nur-Lesen	1-set-up	0	Int32
19-99		Anwendungsparameter 99	0	Nur-Lesen	1-set-up	0	Int32

∗ Standardeinstellung | **[ ] bei Kommunikation über serielle Schnittstelle benutzter Wert**

□ MCO Grundeinstellungen, Parameterliste

Par. Nr. #	Parametername	Parameterbeschreibung	Werksein-stellung	Ändern während des Betriebs	4-Par.-sätze	Umwand-lungs-index	Typ
32-0*	**Drehgeber 2 - Slave**						
32-00	ENCODERTYPE	Inkrementalgeber Signaltyp	[1] RS422	TRUE	1-set-up		Uint8
32-01	ENCODER	Inkrementalgeber Auflösung	1024 PPR	TRUE	1-set-up		Uint32
32-02	ENCODER ABSTYPE	Absolutgeber Protokoll	[0] Keiner	TRUE	1-set-up		Uint8
32-03	ENCODER ABSRES	Absolutgeber Auflösung	8192 Pulse/U	TRUE	1-set-up		Uint32
32-05	ENCODER ABSTYPE	Absolutgeber Datenlänge	25 Bit	TRUE	1-set-up		Uint8
32-06	ENCODERFREQ	Absolutgeber Taktfrequenz	262.000 kHz	TRUE	1-set-up		Uint32
32-07	ENCODER CLOCK	Absolutgeber Takterzeugung	[1] On	TRUE	1-set-up		Uint8
32-08	ENCODER DELAY	Absolutgeber Kabellänge	0	TRUE	1-set-up		Uint16
32-09	ENCODER MONITORING	Drehgeber-Überwachung	[0] Aus	TRUE	1-set-up		Uint8
32-10	POSDRCT	Drehrichtung	[1] Keine Funktion	TRUE	1-set-up		Uint8
32-11	POSFACT_N	Benutzerfaktor Nenner	1	TRUE	1-set-up		Uint32
32-12	POSFACT_Z	Benutzerfaktor Zähler	1	TRUE	1-set-up		Uint32
32-3*	**Drehgeber 1 - Master**						
32-30	MENCODER TYPE	Inkrementalgeber Signaltyp	[1] RS422	TRUE	1-set-up		Uint8
32-31	MENCODER	Inkrementalgeber Auflösung	1024 PPR	TRUE	1-set-up		Uint32
32-32	MENCODER ABSTYPE	Absolutgeber Protokoll	[0] Keiner	TRUE	1-set-up		Uint8
32-33	MENCODER ABSRES	Absolutgeber Auflösung	8192 Pulse/U	TRUE	1-set-up		Uint32
32-35	MENCODER DATLEN	Absolutgeber Datenlänge	25 Bit	TRUE	1-set-up		Uint8
32-36	MENCODER FREQ	Absolutgeber Taktfrequenz	262.000 kHz	TRUE	1-set-up		Uint32
32-37	MENCODER CLOCK	Absolutgeber Takterzeugung	[1] Ein	TRUE	1-set-up		Uint8
32-38	MENCODER DELAY	Absolutgeber Kabellänge	0	TRUE	1-set-up		Uint16
32-39	MENCODER MONITORING	Drehgeber-Überwachung	[0] Aus	TRUE	1-set-up		Uint8
32-40	MENCODER TERM	Drehgeber-Abschluss-widerstand	[1] Ein	TRUE	1-set-up		Uint8
32-5*	**Rückführungsquelle**						
32-50	Rückführung Slave		[2] Enc2	TRUE	1-set-up		
32-6*	**PID-Regelung**						
32-60	KPROP	Proportionalfaktor	30	TRUE	1-set-up	0	Uint32
32-61	KDER	Differentialwert für PID-Regelung	0	TRUE	1-set-up	0	Uint32
32-62	KINT	Integralfaktor	0	TRUE	1-set-up	0	Uint32

∗ Standardeinstellung | **[ ] bei Kommunikation über serielle Schnittstelle benutzter Wert**

Par. Nr. #	Parametername	Parameterbeschreibung	Werksein-stellung	Ändern während des Betriebs	4-Par.-sätze	Umwand-lungs-index	Typ
32-63	KILIM	Grenzwert für die Integralsumme	1000	TRUE	1-set-up	0	Uint16
32-64	BANDWIDTH	PID Bandbreite	1000	TRUE	1-set-up	0	Uint16
32-65	FFVEL	Geschwindigkeits-Feed-forward	0	TRUE	1-set-up	0	Uint32
32-66	FFACC	Beschleunigungs-Feed-forward	0 %	TRUE	1-set-up	0	Uint32
32-67	POSERR	Maximal tolerierter Positionsfehler	20000 qc	TRUE	1-set-up		Uint32
32-68	REVERS	Reversierungsverhalten Slave	[0] Rever-sieren	TRUE	1-set-up		Uint8
32-69	TIMER	Abtastzeit für PID-Regelung	1 ms	TRUE	1-set-up		Uint16
32-70	PROFTIME	Abtastzeit für Profilgenerator	[1] 1 ms	TRUE	1-set-up		Uint8
32-71	REGWMAX	Größe des Regelfensters (Aktivierung)	0 qc	TRUE	1-set-up		Uint32
32-72	REGWMIN	Größe des Regelfensters (Deaktivierung)	0 qc	TRUE	1-set-up		Uint32
32-8*	**Geschwindigkeit & Beschleunigung**						
32-80	VELMAX	Maximalgeschwindigkeit (Encoder)	1500 RPM	TRUE	1-set-up		Uint32
32-81	RAMPMIN	Kürzeste Rampe	1 s	TRUE	1-set-up		Uint32
32-82	RAMPTYPE	Rampenform	0	TRUE	1-set-up		Uint8
32-83	VELRES	Geschwindigkeitsteiler	100	TRUE	1-set-up		Uint16
32-84	DFLTVEL	Default-Geschwindigkeit	50	TRUE	1-set-up		Uint16
32-85	DFLTACC	Default-Beschleunigung	50	TRUE	1-set-up		Uint16

□ MCO weitere Einstellungen, Parameterliste

Par. Nr. #	Parametername	Parameterbeschreibung	Werkseinstellung	Ändern während des Betriebs	4-Par.-sätze	Umwand-lungs-index	Typ
33-0*	**Homefahrt**						
33-00	HOME_FORCE	Homefahrt erzwingen?	[0] nein	TRUE	1-set-up		Uint8
33-01	HOME_OFFSET	Nullpunkt-Offset bezüglich Home-Position	0 qc	TRUE	1-set-up		Int32
33-02	HOME_RAMP	Homefahrt-Rampe	10	TRUE	1-set-up		Uint16
33-03	HOME_VEL	Homefahrt-Geschwindigkeit	10	TRUE	1-set-up		Int16
33-04	HOME_TYPE	Homefahrt-Verhalten	[0] Reverse + Index	TRUE	1-set-up		Uint8
33-1*	**Synchronisation**						
33-10	SYNCFACTM	Synchronisationsfaktor Master (M:S)	1	TRUE	1-set-up		Int32
33-11	SYNCFACTS	Synchronisationsfaktor Slave (M:S)	1	TRUE	1-set-up		Int32
33-12	SYNCPOSOFFS	Positionsoffset für Synchronisation	0 qc	TRUE	1-set-up		Int32
33-13	SYNC ACCURACY	Genauigkeitsfenster für Positionssynchronisation	1000 qc	TRUE	1-set-up		Int32
33-14	SYNCVELREL	Relative Geschwindigkeits-begrenzung Slave	0 %	TRUE	1-set-up		Uint8
33-15	SYNCMARKM	Markeranzahl Master	1	TRUE	1-set-up		Uint16
33-16	SYNCMARKS	Markeranzahl Slave	1	TRUE	1-set-up		Uint16
33-17	SYNCMPULSM	Markerabstand Master	4096	TRUE	1-set-up		Uint32
33-18	SYNCMPULSS	Markerabstand Slave	4096	TRUE	1-set-up		Uint32
33-19	SYNCMTYPM	Markertyp Master	[0] Enc. Z pos.	TRUE	1-set-up		Uint8
33-20	SYNCMTYPS	Markertyp Slave	[0] Enc. Z pos.	TRUE	1-set-up		Uint8
33-21	SYNCMWINM	Master-Marker Toleranzfenster	0	TRUE	1-set-up		Uint32
33-22	SYNCMWINS	Slave-Marker Toleranzfenster	0	TRUE	1-set-up		Uint32
33-23	SYNCMSTART	Startverhalten für Marker-synchronisation	[0] Start Funkt. 1	TRUE	1-set-up		Uint16
33-24	SYNCFAULT	Markeranzahl für Fault	10	TRUE	1-set-up		Uint16
33-25	SYNCREADY	Markeranzahl für Ready	1	TRUE	1-set-up		Uint16
33-26	SYNCVFTIME	Geschwindigkeitsfilter	0 µs	TRUE	1-set-up	0	Int32
33-27	SYNCOFFTIME	Offset Filterzeit	0 ms	TRUE	1-set-up		Uint32
33-28	SYNCMFPAR	Markerfilter Konfiguration	[0] Marker Filter 1	TRUE	1-set-up		Uint8
33-29	SYNCMFTIME	Filterzeit für Markerkorrektur	0 ms	TRUE	1-set-up		Int32
33-30	SYNCM MAXCORR	Maximale Markerkorrektur	[0] Off	TRUE	1-set-up		Uint32
33-31	SYNCTYPE	Synchronisationstyp	[0] Standard	TRUE	1-set-up		Uint8

∗ Standardeinstellung | **[ ] bei Kommunikation über serielle Schnittstelle benutzter Wert**

Par. Nr. #	Parametername	Parameterbeschreibung	Werkseinstellung	Ändern während des Betriebs	4-Par.-sätze	Umwand-lungs-index	Typ
33-4*	**Grenzwertbehandlung**						
33-40	ENDSWMOD	Verhalten bei Endschalter	[0] Fehlerunter-programm aufrufen	TRUE	1-set-up		Uint8
33-41	NEGLIMIT	Negative Software-Wegbegrenzung	-500000 qc	TRUE	1-set-up		Int32
33-42	POSLIMIT	Positive Software-Wegbegrenzung	500000 qc	TRUE	1-set-up		Int32
33-43	SWNEGLIMACT	Negative Software-Wegbegrenzung aktiv	[0] Inaktiv	TRUE	1-set-up		Uint8
33-44	SWPOSLIMACT	Positive Software-Wegbegrenzung aktiv	[0] Inaktiv	TRUE	1-set-up		Uint8
33-45	TESTTIM	Messzeit im Zielfenster	0 ms	TRUE	1-set-up		Uint8
33-46	TESTVAL	Zielfenster-Grenzwert	1 qc	TRUE	1-set-up		Uint16
33-47	TESTWIN	Zielfenster-Größe	0 qc	TRUE	1-set-up		Uint16
33-5*	**I/O Konfiguration**						
33-50	I_FUNCTION_1	Klemme X57/1 Digitaler Eingang	[0] keine Funktion	TRUE	1-set-up		Uint8
33-51	I_FUNCTION_2	Klemme X57/2 Digitaler Eingang	[0] keine Funktion	TRUE	1-set-up		Uint8
33-52	I_FUNCTION_3	Klemme X57/3 Digitaler Eingang	[0] keine Funktion	TRUE	1-set-up		Uint8
33-53	I_FUNCTION_4	Klemme X57/4 Digitaler Eingang	[0] keine Funktion	TRUE	1-set-up		Uint8
33-54	I_FUNCTION_5	Klemme X57/5 Digitaler Eingang	[0] keine Funktion	TRUE	1-set-up		Uint8
33-55	I_FUNCTION_6	Klemme X57/6 Digitaler Eingang	[0] keine Funktion	TRUE	1-set-up		Uint8
33-56	I_FUNCTION_7	Klemme X57/7 Digitaler Eingang	[0] keine Funktion	TRUE	1-set-up		Uint8
33-57	I_FUNCTION_8	Klemme X57/8 Digitaler Eingang	[0] keine Funktion	TRUE	1-set-up		Uint8
33-58	I_FUNCTION_9	Klemme X57/9 Digitaler Eingang	[0] keine Funktion	TRUE	1-set-up		Uint8
33-59	I_FUNCTION_10	Klemme X57/10 Digitaler Eingang	[0] keine Funktion	TRUE	1-set-up		Uint8
33-60	IOMODE	Klemme X59/1 und X59/2 Modus	[0] Ausgang	FALSE	1-set-up		Uint8
33-61	I_FUNCTION_11	Klemme X57/11 Digitaler Eingang	[0] keine Funktion	TRUE	1-set-up		Uint8
33-62	I_FUNCTION_12	Klemme X57/12 Digitaler Eingang	[0] keine Funktion	TRUE	1-set-up		Uint8
33-63	O_FUNCTION_1	Klemme X59/1 Digitaler Ausgang	[0] keine Funktion	TRUE	1-set-up		Uint8
33-64	O_FUNCTION_2	Klemme X59/2 Digitaler Ausgang	[0] keine Funktion	TRUE	1-set-up		Uint8

∗ Werkseinstellung **[ ] bei Kommunikation über serielle Schnittstelle benutzter Wert**

Par. Nr. #	Parametername	Parameterbeschreibung	Werkseinstellung	Ändern während des Betriebs	4-Par.-sätze	Umwand-lungs-index	Typ
33-65	O_FUNCTION_3	Klemme X59/3 Digitaler Ausgang	[0] keine Funktion	TRUE	1-set-up		Uint8
33-66	O_FUNCTION_4	Klemme X59/4 Digitaler Ausgang	[0] keine Funktion	TRUE	1-set-up		Uint8
33-67	O_FUNCTION_5	Klemme X59/5 Digitaler Ausgang	[0] keine Funktion	TRUE	1-set-up		Uint8
33-68	O_FUNCTION_6	Klemme X59/6 Digitaler Ausgang	[0] keine Funktion	TRUE	1-set-up		Uint8
33-69	O_FUNCTION_7	Klemme X59/7 Digitaler Ausgang	[0] keine Funktion	TRUE	1-set-up		Uint8
33-70	O_FUNCTION_8	Klemme X59/8 Digitaler Ausgang	[0] keine Funktion	TRUE	1-set-up		Uint8
33-8*	**Globale Parameter**						
33-80	PRGPAR	Aktivierte Programmnummer	-1	TRUE	1-set-up	0	Uint8
33-81	Power-up State	Einschaltstatus	[1] Motor ein	TRUE	1-set-up		Uint8
33-82	STATUS MONITORING	Statusüberwachung Antrieb	[1] Ein	TRUE	1-set-up	0	Uint8
33-83	ERRCOND	Verhalten im Fehlerfall	[0] Leerlauf	TRUE	1-set-up		Uint8
33-84	ESCCOND	Verhalten bei Programmabbruch	[0] Geregelter Stopp	TRUE	1-set-up		Uint8
33-85	EXTERNAL24V	Externe 24VDC MCO Versorgung	[0] Nein	TRUE	1-set-up		Uint8

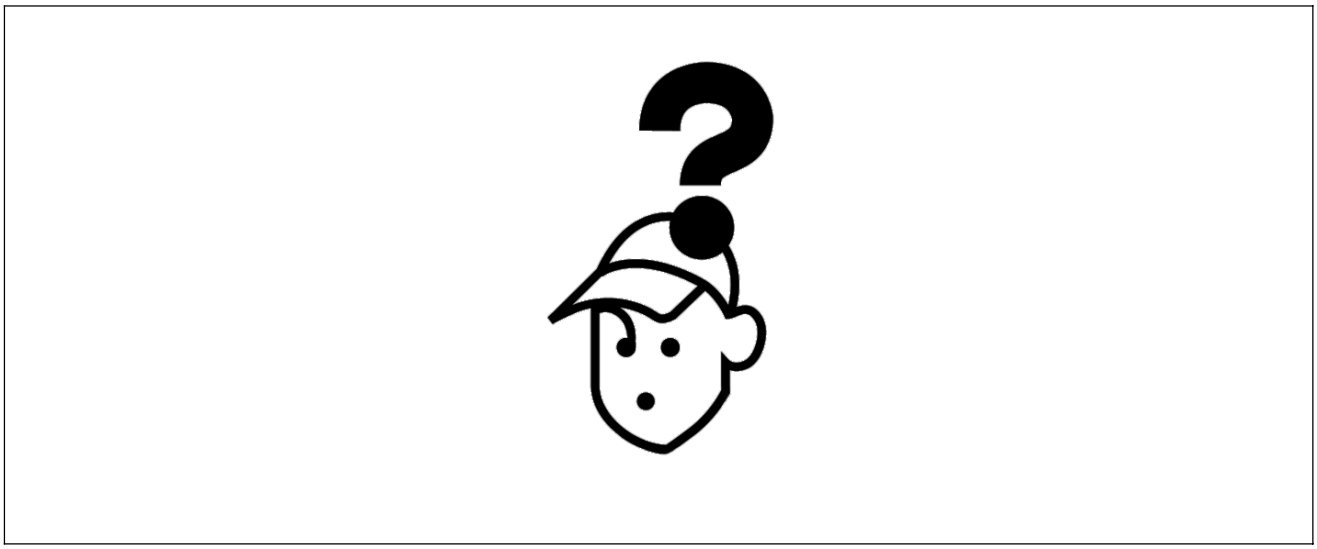
□ MCO Datenanzeigen, Parameterliste

Par. Nr. #	Parametername	Parameterbeschreibung	Werks-ein-stellung	Ändern während des Betriebs	4-Par.-sätze	Umwand-lungs-index	Daten-typ
34-0*	**PCD Schreib-Parameter**						
34-01		PCD 1 nach MCO schreiben			1-set-up		Uint16
34-02		PCD 2 nach MCO schreiben			1-set-up		Uint16
34-03		PCD 3 nach MCO schreiben			1-set-up		Uint16
34-04		PCD 4 nach MCO schreiben			1-set-up		Uint16
34-05		PCD 5 nach MCO schreiben			1-set-up		Uint16
34-06		PCD 6 nach MCO schreiben			1-set-up		Uint16
34-07		PCD 7 nach MCO schreiben			1-set-up		Uint16
34-08		PCD 8 nach MCO schreiben			1-set-up		Uint16
34-09		PCD 9 nach MCO schreiben			1-set-up		Uint16
34-10		PCD 10 nach MCO schreiben			1-set-up		Uint16
34-2*	**PCD Lese-Parameter**						
34-21		PCD 1 von MCO lesen			1-set-up		Uint16
34-22		PCD 2 von MCO lesen			1-set-up		Uint16
34-23		PCD 3 von MCO lesen			1-set-up		Uint16
34-24		PCD 4 von MCO lesen			1-set-up		Uint16
34-26		PCD 6 von MCO lesen			1-set-up		Uint16
34-27		PCD 7 von MCO lesen			1-set-up		Uint16
34-28		PCD 8 von MCO lesen			1-set-up		Uint16
34-29		PCD 9 von MCO lesen			1-set-up		Uint16
34-30		PCD 10 von MCO lesen			1-set-up		Uint16
34-4*	**Eingänge & Ausgänge**						
34-40		Digitale Eingänge			1-set-up		Uint16
34-41		Digitale Ausgänge			1-set-up		Uint16
34-5*	**Prozessdaten**		**Unit**				
34-50		Istposition	BE		1-set-up		Int32
34-51		Sollposition	BE		1-set-up		Int32
34-52		Istposition Master	qc		1-set-up		Int32
34-53		Indexposition Slave	BE		1-set-up		Int32
34-54		Indexposition Master	qc		1-set-up		Int32
34-55		Kurvenposition			1-set-up		Int32
34-56		Schleppfehler	BE		1-set-up		Int32
34-57		Synchronisationsfehler	BE		1-set-up		Int32
34-58		Aktuelle Geschwindigkeit	BE/s		1-set-up		Int32
34-59		Aktuelle Master-Geschwindigkeit	qc/s		1-set-up		Int32

∗ Werkseinstellung | **[ ] bei Kommunikation über serielle Schnittstelle benutzter Wert**

Par. Nr. #	Parametername	Parameterbeschreibung	Werks-ein-stellung	Ändern während des Betriebs	4-Par.-sätze	Umwand-lungs-index	Daten-typ
34-60		Synchronisationsstatus			1-set-up		Int32
34-61		Achsstatus			1-set-up		Int32
34-62		Programmstatus			1-set-up		Int32
34-7*	**Diagnoseanzeigen**						
34-70		MCO Alarmwort 1		FALSE	1-set-up		Uint32
34-71		MCO Alarmwort 2		FALSE	1-set-up		Uint32

Fehlersuche und -behebung



Warnungen und Fehlermeldungen

Alle Meldungen werden im LCP-Display des FC 300 in Kurzform und in der APOSS-Software im Klartext angezeigt.

Informieren Sie sich in der Tabelle in Kürze oder im darauf folgenden Abschnitt im Detail über die Fehlermeldungen.

Die Tabelle enthält die Meldungen in numerischer Reihenfolge. Buchstaben hinter einem %-Zeichen stehen für variable Werte, die im Klartext an den entsprechenden Stellen eingesetzt werden.

Fehler Nr.	Fehlertext	Beschreibung
103	Achse nicht vorh	Achse nicht vorhanden
105	Fehler nicht beseit	Fehler nicht beseitigt
106	HOME nicht angef.	HOME noch nicht angefahren
107	HOME_VEL Null	HOME Geschwindigkeit 0
108	Schleppfehler	Schleppabstand überschritten
109	Index nicht gefund.	Indeximpuls (Encoder) nicht gefunden.
110	Unbekannter Befehl	Unbekannter Befehl
111	SW Endschalter	Software-Endschalter überschritten.
112	Falsche Parameternr	Falsche Parameternummer.
113	FC nicht freigegeben	FC 300 ist nicht bereit aber die PID-Regelung ist aktiv.
114	Zu viele LOOPs	Zu viele verschachtelte LOOP Befehle.
115	Par. speichern fehlgeschl.	Parameter speichern fehlgeschlagen.
116	Param. Speicher	Parameter im Speicher defekt.
117	Progr. Speicher	Programme im Speicher defekt.
118	Reset durch CPU	Reset durch CPU ausgelöst.
119	Benutzer Abbruch	Abbruch durch Benutzer.

Fehler Nr.	Fehlertext	Beschreibung
125	HW Endschalter	HW Endschalter aktiviert.
149	Zu viele Interrupts.	Zu viele Interrupt-Funktionen.
150	Keine ext. 24V	Externe Stromversorgung fehlt.
151	Zu viele GOSUB	Zu viele verschachtelte Unterprogramme.
152	Zu viele RETURN	Zu viele RETURN Befehle.
154	D. Ausgang Überlast	Digitaler Ausgang überlastet.
155	LINK fehlgeschlagen	LINKGPAR Befehl fehlgeschlagen.
162	Speicherfehler	Fehler beim Verifizieren; EEPROM: Adresse % defekt.
170	Array Größe (DIM)	Fehler in der DIM Anweisung.
171	Array zu klein	Feldgrenzen über- oder unterschritten.
179	WAITNDX Timeout	Timeout beim Warten auf Index.
184	Zu viele ONTIME	Zu viele TIME Interrupts.
187	Speicher zu klein	Kein Platz mehr für Variablen.
190	Speicher geschützt	Programmspeicher ist schreibgeschützt.
191	Kurven-Array defekt	Kurven-Array in DIM-Anweisung falsch.
192	Drehgeberfehler	Drehgeberfehler
199	Interner MCO Fehler	Interner MCO Fehler

Fehler 103

Achse nicht vorhanden

Es wurde versucht eine Achse anzusprechen, die in der Steuerung nicht vorhanden ist.

Kontrollieren Sie, ob das Programm Achsbefehle mit einer ungültigen Achsnummer oder allgemeine Achsbefehle (…X(*)) enthält.

Fehler 105

Fehler nicht beseitigt

Es wurde versucht einen Bewegungsbefehl auszuführen, obwohl eine aktuell bestehende Fehlermeldung noch nicht gelöscht wurde.

Fehler 106

HOME noch nicht angefahren

Gemäß dem Achsparameter 33-00 *Homefahrt erzwingen?* wird zwingend eine Fahrt zum Maschinennullpunkt gefordert, bevor andere Bewegungsbefehle ausgeführt werden können. Diese Fahrt zum Maschinennullpunkt wurde noch nicht vorgenommen.

Fehler 107

HOME Geschwindigkeit 0

Es wurde versucht, eine Homefahrt auszuführen, aber der Motor ist in Par. 33-03 *Homefahrt-Geschwindigkeit* auf [0] gesetzt.

Fehler 108

Schleppfehler

Der Abstand zwischen der Soll- und Istposition war größer als in Par. 32-67 *maximal tolerierter Positionsfehler* definiert.

Mögliche Ursachen:

- Mechanisch blockierter oder überlasteter Antrieb,
- zu kleiner Par. 32-67 *max. tolerierter Positionsfehler*,
- Solldrehzahl ist größer als in FC 300 Parameter 4-13 *Maximale Drehzahl* und 3-03 *Maximaler Sollwert*,
- zu große Sollbeschleunigung,
- zu geringer Par. 32-60 *Proportionalfaktor* oder
- FC 300 nicht freigegeben.

Fehler 109

Index nicht gefunden

Bei einer Referenz- bzw. Indexsuche konnte der Indeximpuls des Drehgebers nicht innerhalb einer Motorumdrehung gefunden werden.

Mögliche Ursachen:

- Es wird ein Drehgeber ohne Indeximpuls verwendet,
- der Indeximpuls ist nicht korrekt angeschlossen,
- nicht korrekter Indeximpuls (alle drei Kanäle müssen gleichzeitig low sein) oder
- der Par. 32-01 *Inkrementalgeber Auflösung* ist zu niedrig angegeben.

Fehler 110

Unbekannter Befehl

Ursache: Ein Kommunikations- oder Programmfehler. Das Programm muss neu übersetzt und neu geladen werden.

Fehler 111

SW Endschalter

Durch einen Fahrbefehl wurden die Software-Endschalter überschritten oder würden überschritten werden.

Bei einer Bewegung im Drehzahlmodus wird das Überschreiten der Wegbegrenzung erst erkannt, nachdem die aktuelle Position mit dem Software-Endschalter identisch ist.

In diesem Fall wird die Lageregelung abgeschaltet und der Antrieb muss manuell wieder innerhalb des zulässigen Bereichs bewegt werden. Oder die Überwachung des Software-Endschalters muss kurzzeitig mit Hilfe der Achsparameter *Negative* und *Positive Software-Wegbegrenzung* 33-43 und 33-44 deaktiviert werden. Erst danach kann die Fehlermeldung gelöscht werden.

Bei einer Positionierbewegung wird vor dem Start bereits erkannt, dass die Zielposition außerhalb der Wegbegrenzung liegt. In diesem Fall wird die Bewegung nicht ausgeführt und die Fehlermeldung kann gelöscht werden.

Fehler 112

Falsche Parameternummer

Es wurde versucht, mit einem SET oder SETVLT Befehl einen Parameter zu verändern, den es nicht gibt.

Fehler 113

FC nicht freigegeben

FC 300 ist nicht bereit, aber die PID-Regelung ist aktiv. Das FC Statuswort (Bit 09 und Bit 11) wird alle 20 ms überwacht, wenn die PID-Regelung aktiv ist. Der FC 300 ist im „Nicht bereit" Zustand, wenn:

- eine Alarmmeldung vorliegt,
- er im Hand-Modus ist,
- Hand-LCP-Stopp aktiviert ist.

Fehler 114

Zu viele LOOP

Im ausgeführten Programm sind zu viele ineinander geschachtelte LOOP Befehle.

Fehler 115

Par. speichern fehlgeschlagen

Das Speichern der Optionsparameter ist fehlgeschlagen.

Fehler 116

Param. im Speicher defekt

Die Parameter im EEPROM sind nicht mehr korrekt, weil

- EEPROM defekt oder
- Spannungsausfall während des Speicherns.

ACHTUNG!:
Sie müssen die Parameter mit einem 14-22 *Reset* neu initialisieren und diese anschließend wieder mit Ihren eigenen Anwendungsparametern überschreiben.

Fahrprogramme, die Anwendungsparameter voraussetzen, würden sonst nicht mehr korrekt funktionieren.

Fehler 117

Progr. Speicher

Die im EEPROM abgelegten Programmdaten sind nicht mehr vorhanden oder nicht mehr korrekt, weil

- das EEPROM defekt ist oder
- die Spannung während des Speicherns ausgefallen war.

Sie müssen einen 3-Finger-Reset ausführen, um alle Parameter auf ihre Standardeinstellungen (ab Werk) zurückzusetzen und alle Anwendungsprogramme, Arrays und Anwendungsparameter löschen.

Danach laden Sie wieder die Programme und alle Parameter.

Fehler 118

Reset durch CPU ausgelöst

Der Prozessor wurde angehalten und ein automatisches Reset ausgeführt (Watchdog).

Ursachen könnten sein:

- Kurzzeitiger Spannungsabfall,
- Spannungsspitze oder
- Kurzschluss.

Fehler 119

Benutzer Abbruch

Das Autokennungs-Programm (Autostart) wurde durch den Benutzer abgebrochen.

Oder die [CANCEL]-Taste wurde während des Hochfahrens gedrückt und damit ein Master-Reset ausgelöst.

Fehler 125

HW Endschalter

Ein Fahrbefehl hat den Endschalter einer Achse aktiviert.

Durch diese Aktivierung wurde die Steuerung (abhängig von Par. 33-40 *Verhalten bei Endschalter*) automatisch abgeschaltet und der Antrieb muss manuell aus dieser Position herausgefahren werden, bevor die Fehlermeldung zurückgesetzt werden kann.

Fehler 149

Zu viele Interrupts

Es wurden mehr als die maximal möglichen Interrupt-Funktionen benutzt. Erlaubt sind:

32 ON INT
32 ON STATBIT
32 ON COMBIT
10 ON PARAM
20 ON APOS, ON MAPOS, ON MCPOS

Fehler 150

Keine ext. 24V

Externe Stromversorgung fehlt.

Fehler 151

Zu viele GOSUB

Im ausgeführten Programm wurde zu häufig von einem Unterprogramm direkt in das nächste Unterprogramm gesprungen.

Der Fehler tritt meist dann auf, wenn man rekursiv im Unterprogramm auf eines der Unterprogramme verweist (= Unterprogramme, die sich selbst aufrufen).

Vermeiden Sie zu viele gegenseitige (maximal 10!) und möglichst auch rekursive Unterprogrammaufrufe.

Fehler 152

Zu viele RETURN

Im ausgeführten Programm sind entweder mehr RETURN als entsprechende GOSUB Befehle, oder es wurde direkt mit einem GOTO Befehl in ein Unterprogramm gesprungen.

Pro Unterprogramm ist nur ein RETURN erlaubt.

Es ist immer besser, an den Anfang des Unterprogramms zu springen und dann mit IF... nach einem vorher definierten Label zu springen.

Fehler 154

Dig. Ausgang Überlast

Digitaler Ausgang überlastet.

Fehler 155

LINK fehlgeschlagen

LINKGPAR Befehl fehlgeschlagen.

Fehler 162

Speicherfehler

Nach einem Speichervorgang ins EEPROM (Programm oder Parameter) wurde beim Verifizieren ein Fehler festgestellt.

Löschen Sie das EEPROM mit einem 3-Finger-Reset und versuchen Sie noch einmal das Programm oder die Parameter zu speichern.

Wenn es nicht gelingt, wenden Sie sich bitte an den Service.

Fehler 170

Array Größe (DIM)

Eine Array-Definition in einer DIM-Anweisung stimmt nicht mit den bereits existierenden Arrays in der MCO 305 überein.

Die Felder in der Steuerung könnten von älteren SYNCPOS/APOSS-Programmen stammen und das aktuelle Programm hat andere Definitionen.

Passen Sie entweder das APOSS-Programm an die richtige Array-Größe an oder löschen Sie die alten Arrays.

ACHTUNG!:
Beachten Sie aber die Ratschläge zur Sicherung der Programme und Parameter, bevor Sie das EEPROM löschen.

Fehler 171

Array zu klein

Es wurde versucht ein Array-Element zu beschreiben, das außerhalb der definierten Array-Grenzen liegt. Ursache könnte ein Fehler im APOSS-Programm sein:

- Die Dimensionierung des Arrays stimmt nicht mit dem benötigten Platz überein (z. B. durch eine falsch programmierte Schleife).
- Oder das Array ist für die Anzahl der mit TESTSTART ausgelösten Testfahrten zu klein.
- Prüfen Sie die LOOP Variablen.

Fehler 179

WAITNDX Timeout

Der Befehl WAITNDX wurde ausgeführt und der darin angegebene Timeout überschritten.

Vermutlich ist der Timeout zu kurz gesetzt oder der Indeximpuls konnte nicht gefunden werden (siehe auch Fehler 109).

Fehler 184

Zu viele ONTIME

Im Programm sind zu viele ON TIME oder ON PERIOD Befehle benutzt worden.

Es sind maximal 12 dieser ON TIME und/oder ON PERIOD Befehle innerhalb eines Programms erlaubt**.**

Fehler 187

Var.Speich.zu klein

Kein Platz mehr für Variablen.

Beim Start eines APOSS-Programms wird dynamisch der Platz für die benötigten Variablen reserviert. Dieser Platz ist jetzt nicht mehr vorhanden.

Eventuell haben Sie die maximale Anzahl der Variablen zu groß gewählt. Reduzieren Sie diese in *Einstellungen* → *Compiler* (Standard = 92).

Oder der verfügbare Speicher ist mit Programmen oder Arrays belegt. Löschen Sie die Programme oder beides, die Programme und Arrays durch Löschen des gesamten Speichers

ACHTUNG!:
Beachten Sie aber die Ratschläge zur Sicherung der Programme und Parameter, bevor Sie das EEPROM löschen.

Fehler 190

Memory locked

Der Programmspeicher ist schreibgeschützt und kann nicht verändert werden.

Sie können also Autokennung weder setzen noch löschen und keine Programme sichern oder löschen. Ebenso werden → *RAM speichern* und → *EEPROM löschen* nicht ausgeführt.

Fehler 191

Kurvenarray falsch

In der DIM-Anweisung für SETCURVE wird ein falsches oder altes Array definiert.

Ein alters Array könnte existieren, wenn die CNF-Datei mit allen Parametern und Arrays noch nicht in den *CAM-Editor* geladen wurde.

Ursachen eines falschen Arrays können sein:

- Es wurde nicht vom CAM-Editor erzeugt.
- Es stammt von einer früheren Version des CAM-Editors. Ein solches Array muss erst durch den aktuellen CAM-Editor konvertiert werden (→ *laden* und → *speichern*).
- Oder die Reihenfolge eines Arrays in der DIM-Anweisung stimmt nicht mit der Reihenfolge in der CNF-Datei überein. Sehen Sie dazu auch die Nummer des Arrays in der Titelleiste im *CAM-Editor*.

Fehler 192

Drehgeberfehler

Fehler von der Drehgeberüberwachung: Offener Stromkreis oder Kurzschluss entsprechend der anzeigenden LED.

ACHTUNG!:
Dieser Fehler wird auch angezeigt, wenn kein Drehgeber angeschlossen ist.

Fehler 199

Interner MCO Fehler

Sollte dieser Fehler auftreten, setzen Sie sich bitte mit Ihrem Händler in Verbindung und nennen dem Service die dazu angezeigte Fehlernummer.

□ Meldungen von der APOSS-Software

Die Meldungen von der APOSS-Software sind alphabetisch geordnet. Buchstaben hinter einem %-Zeichen stehen für variable Werte, die im Klartext an den entsprechenden Stellen eingesetzt werden.

Fehlertext
Anschluss %d Pin %d ist nicht erlaubt in Zeile %d Spalte %d
Fehler beim Kompilieren: Programm nicht gespeichert!
Fehler in Datei: Achsparameter
Fehler in Datei: Array-Daten
Fehler in Datei: Globale Parameter
Steuerung führt ein Programm oder Kommando aus!
Timeout: Keine Antwort vom FC
Verbindung zu #%d abgebrochen!
Verbindung zu %d besteht bereits [%s] - Wechsel zum neuen Fenster?

Anschluss-Pin nicht erlaubt

Anschluss %d Pin %d ist nicht erlaubt in Zeile %d Spalte %d

Im OUT Befehl wurde eine ungültige Kombination oder Pin-Nummer verwendet, die so nicht gesetzt werden kann.

Fehler beim Compilieren: Programm nicht gespeichert!

Eine Datei wird immer erst kompiliert und dann gespeichert. Wenn Sie das Programm speichern wollen, zum Beispiel im Menü *Steuerung* → *Programm sichern* und beim Kompilieren ein Syntaxfehler festgestellt wird, erhalten Sie diese Meldung.

Starten Sie die *Syntaxprüfung* im Menü *Entwicklung*, beheben Sie den Syntaxfehler und speichern Sie dann das Programm.

Fehler in Datei: Achsparameter

Um eine Datei zurückspeichern zu können (z.B. mit → *Wiederherstellen aus Datei)*, müssen folgende Bedingungen erfüllt sein:

- Identische Softwareversionen und damit gleiche Anzahl und Reihenfolge der Parameter.
- Gleiche Konfiguration (z.B. gleiche Anzahl der Achsen).

Fehler in Datei: Array-Daten

Beim Zurückspeichern einer Konfiguration (z.B. mit → *Wiederherstellen aus Datei)* wurde erkannt, dass die Array-Daten nicht korrekt formatiert sind. Um eine Datei speichern zu können, müssen folgende Bedingungen erfüllt sein:

- Identische Software-Versionen
- Gleiche Konfiguration

Falls bereits Arrays angelegt sind, müssen diese in Art und Größe zu denen passen, die zurückgespeichert werden sollen.

Fehler in Datei: Globale Parameter

Beim Zurückspeichern einer Konfiguration (z.B. mit → *Wiederherstellen aus Datei)* wurde erkannt, dass die globalen Parameter nicht korrekt formatiert sind. Um eine Datei speichern zu können müssen folgende Bedingungen erfüllt sein:

- Identische Software-Versionen und damit gleiche Anzahl und Reihenfolge der Parameter
- Gleiche Konfiguration

Steuerung führt ein Programm oder Kommando aus!

Während die Steuerung einen Befehl oder Programm ausführt, steht sie nicht für weitere Befehle zur Verfügung. Sie müssen den neuen Befehl → *Abbrechen ESC* und erneut starten, wenn der vorhergehende Befehl vollständig ausgeführt ist.

Timeout: Keine Antwort vom FC

Der FC 300 antwortet nicht; überprüfen Sie die Verbindung.

Verbindung zu ... abgebrochen

Wenn der FC 300 ausgeschaltet, der Stecker gezogen wird, usw. wird das Editier-Fenster vom FC 300 getrennt und die abgebrochene Verbindung gemeldet.

Verbindung zu ... besteht bereits

Verbindung zu %d besteht bereits [%s] - Wechsel zum neuen Fenster?

Die Meldung erscheint beim Öffnen eines neuen Fensters in APOSS oder beim Versuch ein Fenster mit einer Steuerung zu verbinden, mit der bereits ein Fenster verbunden ist.

Ja: Die Steuerung wird vom alten Fenster getrennt und mit dem neuen verbunden.

Nein: Die Steuerung bleibt mit dem alten Fenster verbunden, das neue Fenster hat keine Verbindung zu einer Steuerung.

Anhang

□ Verschaffen Sie sich einen Überblick über alle Programmbeispiele

Einige der Beispiele werden auch mit der Software geliefert. Der übliche Doppelklick auf den Dateinamen öffnet das Beispiel in der APOSS-Software.

Einige der Beispiele sind im „Produkthandbuch" oder im Kapitel „Funktionen und Beispiele" in diesem Projektierungshandbuch beschrieben. Alle anderen Beispiele finden Sie in der Online-Hilfe.

Alle Beispiele können kopiert und in die APOSS-Software eingefügt werden. Bitte lesen Sie die Sicherheitshinweise, bevor Sie die Beispiele benutzen!

Dateiname oder Thema	Sie finden das Beispiel:		Beschreibung
	Online Hilfe	Handbuch oder Software	
Absolute Positionierung	x	Projektierungs-handbuch Seite 20	Absolute Positionierung für das Anwendungsbeispiel „Palettierer".
ACC_01.M	x		Vergleich verschiedener Positioniervorgänge bei konstanter Geschwindigkeit, aber mit unterschiedlicher Beschleunigung.
APOS_01.M	x		Anzeige der aktuellen Positionswerte während eines Positionier-vorgangs.
AXEND_01.M	x		Anzeige des Achsstatus bei verschiedenen Bewegungszuständen.
CAM Box	x	Projektierungs-handbuch Seite 44	Nockenschaltwerk: Nach dem Bedrucken eines Kartons soll der frische Druck sofort im Luftstrom getrocknet werden.
CHR_01.M	x		Bildschirmsteuerzeichen (ASCII) ausgeben.
CMODE_01.M	x		Verschiedene Bewegungsvorgänge im Drehzahlmodus ausführen.
COM_OPT	x	APOSS Software	Programm zum Senden und Empfangen von 8-Byte-Datenworten via Kommunikationsoption und PPO Typ 2.
CPOS_01.M	x		Anzeige der intern errechneten Sollpositionen während einer Bewegung im Drehzahl- oder Positioniermodus.
DELAY_01.M	x		Demonstration einer Wartezeit und des Einflusses von eventuell auftretenden Interrupts. Definition der Interrupt-Quellen und den entsprechenden Unterprogrammen (Interrupthandler).

Dateiname oder Thema	Sie finden das Beispiel:		Beschreibung
	Online Hilfe	Handbuch oder Software	
DIM_01.M	x		Aufzeichnung einer Geschwindigkeitskurve und anschließende Ausgabe als horizontales Balkendiagramm.
DORIG_01.M	x		Definieren des Realnullpunktes an verschiedenen Positionen.
Drehg-S.m	x	APOSS Software	Drehgeber-Testprogramm.
ERROR_01.M	x		Auswertung eines Fehlerzustands mittels der Fehlernummer.
EXIT_01.M	x		Ratespiel: Eine Ziffer muss erraten werden.
Feed-forward Berechnung	x	MCO 305 Produkthandbuch	Feed-forward Berechnung.
GETP_01.M	x		Ausgabe der aktuellen Filterparameter.
GOSUB_01.M	x		Zeit- und Weginformationen während einer Bewegung anzeigen.
GOTO_01.M	x		Innerhalb einer Endlosschleife wird alle 5 Sekunden die aktuelle Zeit seit dem Programmstart ausgegeben.
HOME_01.M	x		Anfahren der HOME-Position (Referenzschalter + INDEX).
IF_01.M	x		Auswertung eines Fehlerzustands anhand der Fehlernummer.
IN_01.M	x		Anzeige der Signalpegel an den digitalen Eingängen.
INB_01.M	x		Anzeige der Signalpegel an den digitalen Eingängen.
INB_02.M	x		Anzeige des Signalpegels an den digitalen Eingängen. Bei jeder Pegeländerung an einem der Eingänge wird die Anzeige neu aufgebaut und die Signalzustände der Eingänge auf die Ausgänge übernommen.
INCL_01.M	x		Zeigt verschiedene Systeminformationen an.
INCIN01.M	x		Include-Datei: Zustand der Eingänge grafisch anzeigen.
INCPOS01.M	x		Include-Datei: Anzeige der aktuellen, sowie der Sollposition.
INCSTA01.M	x		Include-Datei: Unterprogramm zum Anzeigen des Systemstatus in Klartext.
INDEX_01.M	x		Anfahren der Indexposition des Drehgebers.
INKEY_01.M	x		Siehe EXIT_01.M
LOOP_01.M	x		Darstellung zufälliger horizontaler Balken.
Marker.m	x	APOSS Software	Kurvenscheibensteuerung (CAM): Kartons bedrucken mit Markerkorrektur.
Marker count	x		Messen des Markerabstands in Verbindung mit dem Befehl SYNCM.
Marker Synchronisation	x	Projektierungs-handbuch Seite 33	Markersynchronisation für das Anwendungsbeispiel „Verpacken mit variierenden Produktabständen und Schlupf”.
Mechanische Bremssteuerung	x	Produkthandbuch Seite 45	Relative Positionierung mit mechanischer Bremse.
MOTOR_01.M	x		Manuelle, überwachte Bewegung des Antriebs bei abgeschalteter Lageregelung.
Fahrtst.m	x	APOSS Software	Fahrtestprogramm.
MSTOP_01.M	x		Positioniervorgang unterbrechen und nach Warten wieder fort-setzen.

Dateiname oder Thema	Sie finden das Beispiel:		Beschreibung
	Online Hilfe	Handbuch oder Software	
NOWAI_01.M	x		Ausgabe von Informationen während eines Positioniervorgangs.
ONINT_01.M	x		Abarbeitung von Interrupt-Signalen während Positioniervorgängen.
ORIG_01.M	x		Anfahren des Realnullpunktes nach einer undefinierten Bewegung im Drehzahlmodus.
OUT_01.M	x		Setzen verschiedener Ausgänge in Abhängigkeit von der aktuellen Position.
OUTB_01.M	x		Zustand der digitalen Eingänge auf digitale. Ausgänge übertragen.
POS_01.M	x		Relatives und absolutes Positionieren des Antriebs.
Positions-Synchronisation	x	Projektierungs-handbuch Seite 28	Positions-Synchronisation für das Anwendungsbeispiel „Verpacken mit festen Produktabständen”.
Relative Positionierung	x	Projektierungs-handbuch Seite 21	Relative Positionierung für das Anwendungsbeispiel „Palettierer”.
REPEA_01.M,	x		Abarbeitung eines Countdowns und Bewegung im Drehzahlmodus.
slavesync.m	x	APOSS Software	Kurvenscheibensteuerung (CAM): Slave-Synchronisation mit Marker.
stempel.m	x	APOSS Software	Kurvenscheibensteuerung (CAM): Kartons mit Haltbarkeitsdatum stempeln.
STAT_01.M	x		Lesen und Auswerten der Statusinformation.
syncc_msim.m	x		Master-Simulation per Software.
TORIG_01.M	x		Verschiedene temporäre Nullpunkte definieren und Auswirkungen auf die aktuelle Position anzeigen.
Touch-Probe Positioning	x	Projektierungs-handbuch Seite 23	Touch-Probe Positionierung für das Anwendungsbeispiel „Palettierer”.
Geschwindigkeits-synchronisation	x	Projektierungs-handbuch Seite 25	Geschwindigkeitssynchronisation für das Anwendungsbeispiel „Koffertransportband”.
VEL_01.M	x		Variation der Geschwindigkeit während eines Positioniervorgangs.
WAIT_01.M	x		Vorführung verschiedener Wartevorgänge: Zeitverzögerung, Warten auf Positionierung, Warten auf Eingang.
WHILE_01.M	x		Auf einen High-Pegel am Eingang 4 warten und anschließend eventuell vorhandene Eingaben ausgeben.

□ SYNCPOS > MCO 305 Parameter

Es gibt einige neue Parameter in dieser Version. Wegen des Redesigns der Parameternummern sind hier alle neuen und die SYNCPOS-Parameter aufgelistet. Die neuen Parameter sind markiert: „neu".

Einige Parameter wurden entfernt, z.B. I_BREAK oder O_BRAKE. Stattdessen gibt es neue Parameter I_FUNCTION_n und O_FUNCTION_n, die eine Nummer enthalten, mit der festgelegt wird welche Funktion mit diesem Eingang oder Ausgang verbunden ist. Die entfernten Parameter sind mit „ – " markiert, obwohl sie weiterhin kompatibel sind. Sie finden die entsprechenden neuen Parameter in der Spalte „neue Par."

□ Neue Parameter und alle Parameter in alphabetischer Reihenfolge

Parameter-kennung	Parametername	Par. Nr.	SYNC-POS Par. Nr.	neue Par.	Einheit	Werksein-stellung
BANDWIDTH	PID Bandbreite	32-64	(35) 706		%	100
DFLTACC	Default-Beschleunigung	32-85	(34)		%	50
DFLTVEL	Default-Geschwindigkeit	32-84	(33)		%	50
ENCODER	Inkrementalgeber Auflösung	32-01	(2)		Pulse/U	1024
ENCODERCLOCK	Absolutgeber Takterzeugung	32-07	(73)	neu		
ENCODER DATLEN	Absolutgeber Datenlänge	32-05	(71)	neu	Bit	25
ENCODERDELAY	Absolutgeber Kabellänge	32-08	(80)	neu		0
ENCODERFREQ	Absolutgeber Taktfrequenz	32-06	(74)	neu	kHz	262,000
ENCODERTYPE	Inkrementalgeber Signaltyp (Slave)	32-00	(27)		–	0
ENDSWMOD	Verhalten bei Endschalter	33-40	(44)		–	0
ERRCOND	Verhalten im Fehlerfall	33-83	(43)		–	0
ESCCOND	Verhalten bei Programmabbruch	33-84	(70)		–	0
EXTERNAL24V	Externe 24VDC MCO Versorgung	33-85	(110)	neu	–	0
FFACC	Beschleunigungs-Feed-forward	32-66	(37) 708		%	0
FFVEL	Geschwindigkeits-Feed-forward	32-65	(36) 707		%	0
HOME_FORCE	Homefahrt erzwingen?	33-00	(3)		–	0
HOME_OFFSET	Nullpunkt-Offset bezüglich Home-Position	33-01	(42)		qc	0
HOME_RAMP	Homefahrt-Rampe	33-02	(41)		%	10
HOME_VEL	Homefahrt-Geschwindigkeit	33-03	(7)		%	10
HOME_TYPE	Homefahrt-Verhalten	33-04	(40)		–	0
I_BREAK	Eingang für Abbruch	–	(105)	33-50 …	–	0
I_CONTINUE	Programm fortsetzen	–	(106)	33-50 …	–	0
I_ERRCLR	Fehler löschen	–	(107)	33-50 …	–	0
I_FUNCTION_n	Klemme X57/n Digitale Eingänge	33-50...59, 33-61, 33-62	(45-47, 103-107)	neu	–	0
I_NEGLIMITSW	Endschalter negativ	–	(47)	33-50 …	–	0
I_POSLIMITSW	Endschalter positiv	–	(46)	33-50 …	–	0
I_PRGCHOICE	Eingang für Programmwahl Anfang	–	(104)	33-50 …	–	0
I_PRGSTART	Eingang für Programmstart	–	(103)	33-50 …	–	0

Parameter-kennung	Parametername	Par. Nr.	SYNC-POS Par. Nr.	neue Par.	Einheit	Werksein-stellung
I_REFSWITCH	Eingang für Referenzschalter	–	(45)	33-50 …	–	0
IOMODE	Klemme X59/1 und X59/2 Modus	33-60	(113)	neu		0
KDER	Differentialwert für PID-Regelung	32-61	(12) 703		–	1
KILIM	Grenzwert für die Integralsumme	32-63	(21) 705		–	0
KINT	Integralfaktor	32-62	(13) 704		–	0
KPROP	Proportionalfaktor	32-60	(11) 702		–	30
MENCODER CLOCK	Absolutgeber Takterzeugung	32-37	(77)	neu	–	[1] ein
MENCODER	Inkrementalgeber Auflösung	32-31	(30)		Pulse/U	1024
MENCODER DATLEN	Absolutgeber Datenlänge	32-35	(75)	neu	Bit	25
MENCODER DELAY	Absolutgeber Kabellänge	32-38	(81)	neu		0
MENCODERFREQ	Absolutgeber Taktfrequenz	32-36	(78)	neu	kHz	262,000
MENCODERTERM	Drehgeber-Abschlusswiderstand	32-40	(76)	neu	–	1
MENCODERTYPE	Inkrementalgeber Signaltyp (Master)	32-30	(67)		–	1
NEGLIMIT	Negative Software-Wegbegrenzung	33-41	(4)		qc	–500000
O_AXMOVE	Ausgang für Fahrbefehl aktiv	–	(64)	33-63…	–	0
O_BRAKE	Ausgang für mechanische Bremse	–	(48)	33-63…	–	0
O_ERROR	Ausgang für Fehler	–	(108)	33-63…	–	0
O_FUNCTION_n	Klemme X59/n Digitaler Ausgang	33-63…70	(48, 64, 108)	neu	–	0
POSDRCT	Drehrichtung	32-10	(28)		–	1
POSERR	Maximal tolerierter Positionsfehler	32-67	(15)		qc	20000
POSFACT_N	Benutzerfaktor Nenner	32-11	(26)		–	1000
POSFACT_Z	Benutzerfaktor Zähler	32-12	(23)		–	1000
POSLIMIT	Positive Software-Wegbegrenzung	33-42	(5)		qc	500000
PRGPAR	Aktivierte Programmnummer	33-80	(102) 701		–	–1
PROFTIME	Abtastzeit für Profilgenerator	32-70	(29)		1	ms
RAMPMIN	Kürzeste Rampe	32-81	(31)		s	1
RAMPTYPE	Rampenform	32-82	(32)			0
REGWMAX	Größe des Regelfensters (Aktivierung)	32-71	(38)		qc	0
REGWMIN	Größe des Regelfensters (Deaktivierung)	32-72	(39)		qc	0
REVERS	Reversierungsverhalten Slave	32-68	(63)		–	0
STATUS MONITORING	Statusüberwachung Antrieb	33-82	(79) 700	neu	–	1
SWNEGLIMACT	Negative Software-Wegbegrenzung aktiv	33-43	(19)		–	0

Parameter-kennung	Parametername	Par. Nr.	SYNC-POS Par. Nr.	neue Par.	Einheit	Werksein-stellung
SWPOSLIMACT	Positive Software-Wegbegrenzung aktiv	33-44	(20)		–	0
SYNCACCURACY	Genauigkeitsfenster für Positions-synchronisation	33-13	(55)		qc	1000
SYNCFACTM	Synchronisationsfaktor Master (M:S)	33-10	(49)		qc	1
SYNCFACTS	Synchronisationsfaktor Slave (M:S)	33-11	(50)		qc	1
SYNCFAULT	Markeranzahl für Fault	33-24	(57)		–	10
SYNCMARKM	Markeranzahl Master	33-15	(52)		–	1
SYNCMARKS	Markeranzahl Slave	33-16	(53)		–	1
SYNCMFTIME	Filterzeit für Markerkorrektur	33-29	(18)		1 ms	0
SYNCMFPAR	Markerfilter Konfiguration	33-28	(17)		–	1
SYNCMMAXCORR	Maximale Markerkorrektur	33-30	(6)		qc	0
SYNCMPULSM	Markerabstand Master	33-17	(58)		qc	500
SYNCMPULSS	Markerabstand Slave	33-18	(59)		qc	500
SYNCMSTART	Startverhalten für Synchronisation	33-23	(62)		–	0
SYNCMTYPM	Markertyp Master	33-19	(60)		–	0
SYNCMTYPS	Markertyp Slave	33-20	(61)		–	0
SYNCMWINM	Master-Marker Toleranzfenster	33-21	(68)		qc	0
SYNCMWINS	Slave-Marker Toleranzfenster	33-22	(69)		qc	0
SYNCOFFTIME	Offset-Filterzeit	33-27	(16)		ms	0
SYNCPOSOFFS	Positionsoffset für Synchronisation	33-12	(54)		qc	0
SYNCREADY	Markeranzahl für READY	33-25	(56)		–	1
SYNCTYPE	Synchronisation Type	33-31	(51)		–	0
SYNCVELREL	Relative Geschwindigkeits-begrenzung Slave	33-14	(66)		%	0
SYNCVFTIME	Geschwindigkeitsfilter	33-26	(65) 709		τ_filt (μs)	0
TESTTIM	Messzeit im Zielfenster	33-45	(24)		ms	0
TESTVAL	Zielfenster-Grenzwert	33-46	(25)		qc	1
TESTWIN	Zielfenster-Größe	33-47	(8)		qc	0
TIMER	Abtastzeit für PID-Regelung	32-69	(14)		ms	1
VELMAX	Maximalgeschwindigkeit (Encoder)	32-80	(1)		U/Min	3000
VELRES	Geschwindigkeitsteiler	32-83	(22)			100

□ Neues in der aktuellen Version

□ Neue und erweiterte Parameter

Par. 32-09 und Par. 32-39 Drehgeber-Überwachung

Mit [1] und [2] kann man nun wählen, ob alle 3 Kanäle, also A, B und Index oder nur 2 Kanäle, also A, B überwacht werden sollen.

Par. 32-50 Rückführung Slave	Wählen Sie die Rückführungsquelle für den Slave.
Par. 32-82 Rampenform	Weiterer Rampentyp '2' für alle Fahrbewegungen mit Ruckbegrenzung.

Ruckdauer JERKMIN
Definiert die Zeitspanne [ms] in der die definierte Maximalbeschleunigung erreicht werden soll. Vier verschiedene JERKMIN Parameter sind möglich; siehe auch „Ruckbegrenzung” im Kapitel „Funktionen und Beispiele”.

□ Neue SYSVAR Indizes

4275	PFG_G_JSTATE	Enthält den Status des Bewegungsprofils der Ruckbegrenzung.
4277	PFG_G_JERKSTOPPATH	Liefert die Dauer des Bremswegs.

□ Neue Befehle

VLTALARMSTAT	Gibt an, ob ein Alarm vorliegt oder nicht.
VLTCONTROL	Setzt das VLT Steuerwort im MOTOR OFF Status.
VLTERRCLR	Löscht einen VLT-Alarm.

□ Technische Referenz

Dieser Abschnitt dokumentiert Datenstrukturen und Compilerdetails, die der Anwender nur in Ausnahmefällen benötigt. Zum Beispiel wenn eine automatisch erzeugte Programmierung, wie ein CAM-Profil verändert werden soll.

Da der Abschnitt für erfahrene Programmier vorgesehen ist, wird die Referenz nur in Englisch dokumentiert.

□ Array Structure of CAM Profiles

Header

The header contains general information like

- Identification for curve array
- Version number for curve structure
- Type of curve
- Name of curve
- Index to curve information section
- Index to start/stop point section
- Index to fixed point section
- Index to interpolation point section
- Index to start/stop point indices (in interpolation section)
- Index to start/stop velocities (times 100000)
- Index to start path interpolation points
- Index to stop path interpolation points

Curve Information Section

This section of the array contains all information about the type of curve like

- Length of curve (master)
- Length of curve (slave)
- Number of fix points
- Number of Interpolation points (this gives the resolution)
- Type of interpolation
- Slave stop point, point where slave is positioned, when Synchronisation is stopped
- Correction start point (only valid for marker synchronization)
- Correction end point (only valid for marker synchronization)
- Maximum correction which is allowed (only valid for marker synchronization)
- Maximum start/stop path length (Size of start/stop path area)(min. 2)
- Number of start/stop point pairs
- Maximum number of cycles per minute (Application information)

Curve Start/Stop Point Section

This section contains the start/stop points. Because the use of this point is up to the user, we just speak of a path, which can be a start or a stop sequence. Every path consists of 2 points. If we are moving forward, the path starts (start or stop) with the a-point and ends with the b-point. If we are moving backward, the path starts with the b-point and ends with the a-point. So the user is able to tell us in the program, which pair of points to use for starting or stopping, when he uses a STARTCURVE or STOPCURVE command.

- Path 1 (a – point)
- Path 1 (b – point)
- Path 2 (a – point)
- Path 2 (b – point), …

These points have to lie on interpolation points, so possibly the PC software has to adjust them according to the interpolation resolution. This should not be a real restriction, because the interpolation points are normally very dense. So for example if we have rotating master which makes one revolution per cycle and we choose a cycle length of 3600 MU (1 MU = 1/10 degree). Let us further assume, that we choose the number of interpolation points as 1200, than you have a resolution of 3 MU = 3/10 degree for defining your start and stop points.

Fixed Point Section

This section contains the fix points, which were the basis for the interpolation calculation. These points always consist of the following triple

- Master coordinate
- Slave coordinate
- Type of point (tangent, curve)

These points are defined by the user in MU units (see internal description). If you want to avoid, that the real interpolation curve misses your fix points, you have to choose them in such a manner that they lay on an interpolation point (see above). This can be forced through a snap function within the PC software.

Interpolation Point Section

This section contains a list of slave coordinates. They belong to master coordinates which are of equal distance, given by the interpolation resolution.

Indices of Start/Stop Points

Here we have the indices of the start/stop points (see above) within the interpolation array. These are necessary for the ease of start and stop recognition. We are waiting until start index for example equals the actual index and direction of movement is correct. If both are true synchronization will be started.
The same is true for stopping.

Start Stop Velocities

To be able to calculate an appropriate starting or stopping path, we need the velocity we have to reach at end (start) or we will have at the beginning (stop) in UU/MU units (Slave units per Master units).

Start / Stop Paths

This is the place for the interpolation points of the actual start and stop path. These points are calculated when a SYNCCSTART or SYNCCSTOP command is executed, but we have to reserve the room right now.

CAM Array Definition

Index	Name	Unit	Value	Description
General				
1	Identification	(dec)	999.000.001	Number to identify array
2	VersioNumber	(dec)	100	Version as decimal (1.00 = 100)
3	CurveType	(dec)	0	0 = symmetrical; 1 = compatible
4	CurveName 1	(4char)	Nona	Name of curve total 16 char.
5	CurveName 2	(4char)	meCu	default is:
6	CurveName 3	(4char)	rve0	NonameCurve00001
7	CurveName 4	(4char)	0001	
8	IndexCIF	(dec)	16	Index to Curve Information Part
9	IndexSTP	(dec)	27	Index to Start/Stop point Part

Index	Name	Unit	Value	Description
10	IndexFIP	(dec)	IndexSTP + STPno*2	Index to Fix point Part
11	IndexINP	(dec)	IndexFIP + FixPointNo * 3	Index to Interpolation Point Part
12	IndexSTPInd	(dec)	IndexINP + InterpolPointNo	Index to StartStop Interpolation Indices
13	IndexSTPVel	(dec)	IndexSTPInd +STPno*2	Index to StartStop Velocities
14	IndexSTIP	(dec)	IndexSTPVel +STPno*2	Index to Startpath interpolation points
15	IndexSTPIP	(dec)	IndexSTIP + MaxStartStopLen	Index to Stoppath interpolation points
Curve Information				
1	MasterCycleLen	MU	-	Length of Curve in CurveMaster units
2	SlaveCycleLen	UU	-	Slave max. travel distance in CurveSlave units
3	FixPointNo	(dec)	4	Number of fix points (minimum 4)
4	InterpolPointNo	(dec)	-	Number of interpolation points (including first and last, which correspond to the same location)
5	InterpolType	(dec)	0	0 = cubic spline, 1 = periodic cubic spline
6	SlaveStopPosition	UU	0	Position, where slave stands after stopping
7	CorrectionStartPoint	MU	0	Position, where Correction may start
8	CorrectionStopPoint	MU	MasterCycleLen	Position, where Correction has to befinished
9	MaximumCorrection	UU	-	Maximum Correction which is allowed in one cycle
10	MaxStartStopLen	(dec)	0	Maximum length of start/stop path (number of int. points)
11	StartStopNo	(dec)	0	Number of start stop point pairs (n) (see below)
12	MMaxCycles	(dec)	0	Max. number of cycles per minute (application info)
13	MMarkerPos	CM	0	Master Marker Position in curve
14	SMarkerPos	CS	0	Slave Marker Position in curve
Start/Stop Point				
1	STPoint_1.a	MU	0	Start (forward) / Stop (backward) point no. 1
2	STPoint_1.b	MU	0	Stop (forward) / Start (backward) point no. 1
3	STPoint_2.a	MU	0	Start (forward) / Stop (backward) point no. 2
4	STPoint_2.b	MU	0	Stop (forward) / Start (backward) point no. 2
5	...	MU	0	
6	...	MU	0	
2*n-1	STPoint_n.a	MU	0	Start (forward) / Stop (backward) point no. n
2*n	STPoint_n.b	MU	0	Stop (forward) / Start (backward) point no. n

Index	Name	Unit	Value	Description
Fix Point				
1	FixPoint_1.master	MU	0	Fix point no. 1 - master coordinate
2	FixPoint_1.slave	UU	-	Fix point no. 1 - slave coordinate
3	FixPoint_1.type	(dec)	C	Fix point no. 1 - type of point (C = Curve Point, T = Tangent Point
4	...			
5	...			
6	...			
3*n-2	FixPoint_n.master	MU	MasterCycleLen	Fix point no. n - master coordinate
3*n-1	FixPoint_n.slave	UU	-	Fix point no. n - slave coordinate
3*n	FixPoint_n.type	(dec)	C	Fix point no. n - type of point (C = Curve Point, T = Tangent Point)
Interpolation Point				
1	IntPoint_1	UU	0	Interpolation Point no. 1 - slave coordinate
...				
n	IntPoint_n	UU	-	Interpolation Point no. n - slave coordinate
StartStop Indices				
1	STPoint_1.a-index	(dec)	0	Index in Interpolation Array, corresponding to Start point
2	STPoint_1.b-index	(dec)	0	Index in Interpolation Array, corresponding to Start point
3	..			
StartStop Velocities				
1	STPoint_1.a-veloc.	(dec)	(*100000)	Velocity (UU/MU * 100000) in Start point
2	STPoint_1.b-veloc.	(dec)	(*100000)	Velocity (UU/MU * 100000) in Start point
...				
StartPath Interpolation Points				
1	StartPoint_1	UU	0	Interpolation Point no. 1 - for start path
...				
n				
StopPath Interpolation Points				
1	StopPoint_1	UU	0	Interpolation Point no. 1 - for stop path
...				
n				

□ Stichwortverzeichnis

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z