Inhaltsverzeichnis

■ **Projektierungshandbuch lesen 3**

☐ Projektierungshandbuch lesen 3

☐ Verfügbare Literatur für FC 300, MCO 305 und MCT 10 Motion Control Tool 4

☐ Symbole und Konventionen 5

☐ Abkürzungen 5

☐ Definitionen 5

☐ Definitionen 6

■ **Einführung in VLT Motion Control Option MCO 305 ..... 11**

☐ Was ist eine VLT Motion Control Option MCO 305? 11

☐ Systemüberblick 12

☐ Konfigurationsbeispiele 13

☐ Schnittstellen zwischen MCO 305, FC 300 und anderen Options-Modulen 14

☐ PID-Regelung 14

☐ Drehgeber 15

☐ Programmausführung 15

■ **Funktionen und Beispiele 17**

☐ Positionierung 17

☐ Synchronisation 24

☐ Kurvenscheibensteuerung (CAM-Modus) 35

☐ Nockenschaltwerk 47

☐ Mechanische Bremssteuerung 48

☐ Ruckbegrenzung 50

■ **PC Software Benutzeroberfläche 57**

☐ APOSS Benutzeroberfläche 57

☐ Das Editierfenster 60

☐ Menü Datei 63

☐ Menü Bearbeiten 64

☐ Menü Entwicklung 66

☐ Menü Steuerung 72

☐ Menü Tools 81

☐ Menü Einstellungen 82

☐ Menü Fenster 86

☐ Menü Hilfe 86

☐ Menü Download 87

☐ Programme debuggen 89

■ **APOSS Tools 93**

☐ CAM-Editor 93

☐ Array-Editor 109

☐ APOSS Oszilloskop 120

■ **Programmieren mit APOSS 163**

☐ MCO mit der APOSS Makrosprache programmieren 163

☐ Grundlagen 163

☐ Debugging 167

☐ Preprozessor 178

☐ APOSS Befehlsgruppen 181

☐ Verschaffen Sie sich einen Überblick über alle Programmbeispiele 195

■ **Parameter-Referenz 199**

☐ FC 300, MCO 305 und Anwendungsparameter 199

☐ Übersicht FC 300 Parameter 201

☐ Einstellungen für die Anwendung 203

☐ MCO Parameter 204

☐ MCO Grundeinstellungen 204

☐ MCO weitere Einstellungen 220

☐ MCO Datenanzeigen 243

☐ Parameterlisten 246

■ **Fehlersuche und -behebung 255**

☐ Warnungen und Fehlermeldungen 255

☐ Meldungen von der APOSS-Software 264

■ **Index 265**

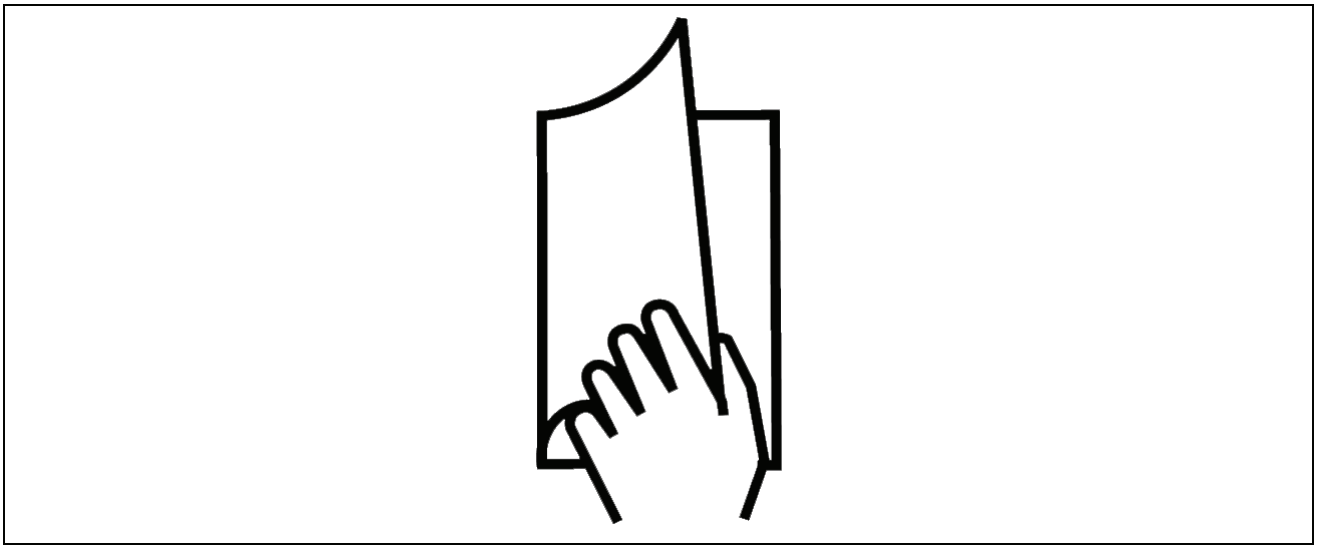
Copyright © Danfoss A/S, 2010

Warenzeichen VLT ist ein eingetragenes Warenzeichen von Danfoss.

Hiperface® ist eingetragenes Warenzeichen der Sick Stegmann GmbH, Max Stegmann GmbH Antriebstechnik-Elektronik.

Microsoft, Windows 2000 und Windows XP sind entweder eingetragene Warenzeichen oder Warenzeichen der Microsoft Corporation in den USA und/oder anderen Ländern.

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.

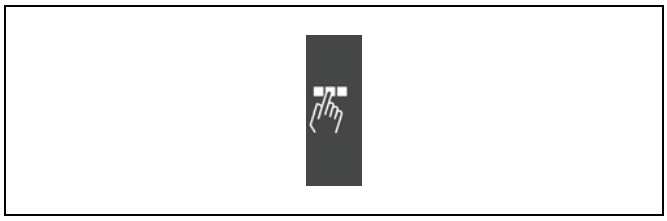
Seitenteiler für das Kapitel „Einführung".

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".

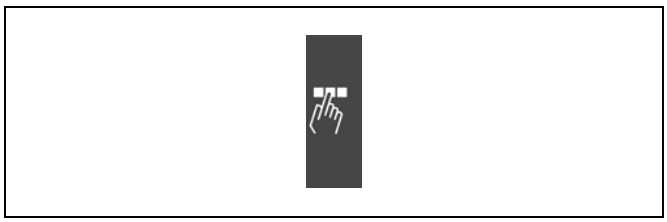
__ Projektierungshandbuch lesen __

Das Kapitel **PC Software Benutzeroberfläche** informiert über die APOSS-spezifischen Menüs und Funktionen. Für mehr Details klicken Sie bitte auf → *Hilfe* in der APOSS Menüleiste. Das Kapitel **APOSS Tools** bietet detaillierte Informationen über den CAM-Editor, Array-Editor sowie das APOSS Oszilloskop.



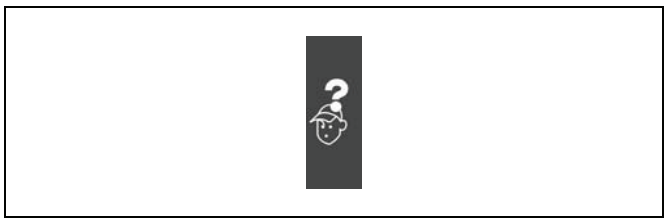
Seitenteiler für „PC Software Benutzeroberfläche".

Das Kapitel **Programmieren** zeigt wie man Steuerungen für den Frequenzumrichter mit MCO 305 programmiert. Dieses Kapitel erläutert alle Befehle nach Gruppen geordnet und alle Parameter in der 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 der PC-Benutzeroberfläche.



Seitenteiler für „Fehlersuche und -behebung".

Das Handbuch schließt mit einem **Stichwortverzeichnis**.

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 MCO 305 Projektierungshandbuch enthält alle technischen Informationen über die Optionskarte sowie Informationen für die Realisierung kundenspezifischer Designs und Anwendungen.
- Die MCO 305 Befehlsreferenz ergänzt das MCO 305 Projektierungshandbuch mit der detaillierten Beschreibung aller Befehle.
- 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.

MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

□ 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

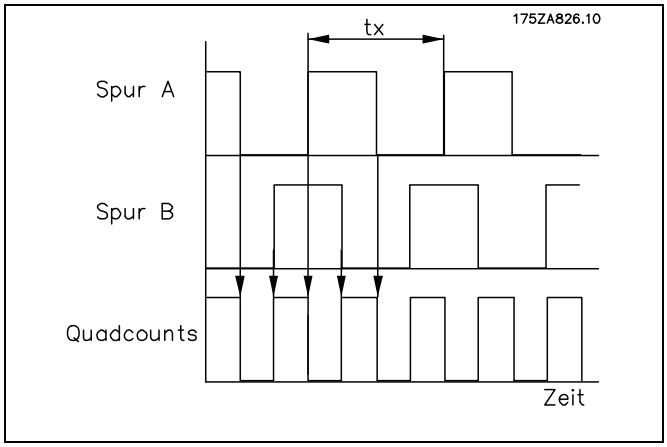
□ Online / Offline Parameter

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.

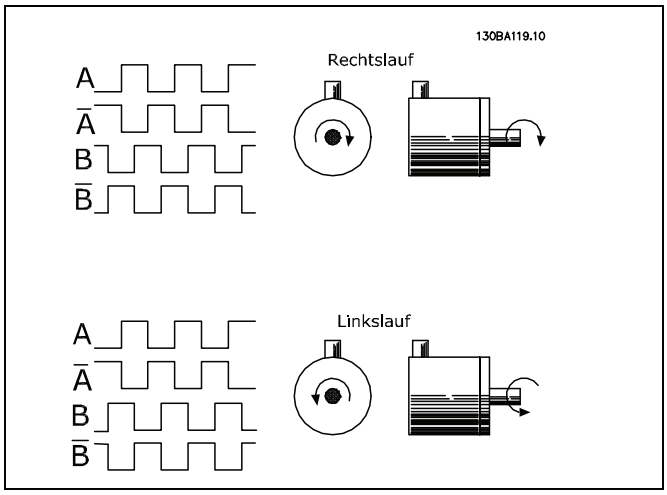
□ Drehgeber-Drehrichtung

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.

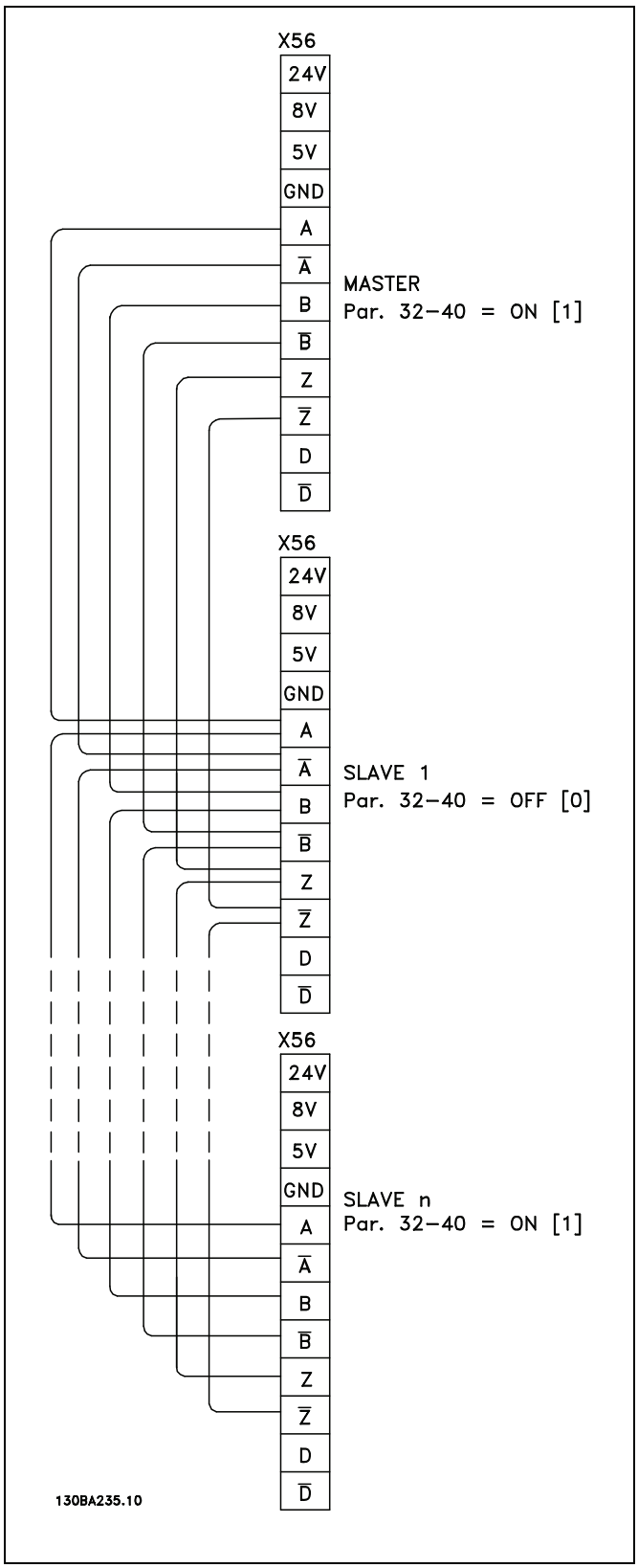
Linksdrehend heißt, dass Kanal B 90° (elektrische Grad) vor Kanal A liegt.

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



☐ 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.

Die Faktoren SYNCFACTM / SYNCFACTS, POSFACT_Z / POSFACT_N sind ab Version MCO 5.00 nicht mehr auf kleine Werte begrenzt.

Intern werden sie wie folgt behandelt: Wann immer ein Wert mit einem Getriebefaktor multipliziert werden muss (d.h. Master Inkremente per ms), wird zuerst geprüft, ob eine Multiplikation einen Überlauf erzeugt. Falls dies der Fall wäre, wird ein Faktor (64 Bit) benutzt, bestehend aus

SYNCFACTS/SYNCFACTM zum Multiplizieren von delta_master.

Falls kein Überlauf entsteht, wird zuerst mit SYNCFACTS multipliziert und dann durch SYNCFACTM geteilt.

Der Fehler wird wie folgt behandelt:

Normaler Fall

Die Multiplikation mit SYNCFACTS erzeugt keinen Fehler, aber die Division durch SYNCFACTM besagt, dass das Ergebnis um $1/2^{32}$ falsch sein kann. Das bedeutet, dass (im schlimmsten Fall) solch ein Fehler jede ms auftritt, d.h. dass nach 1193 Stunden (49,71 Tage) ein Fehler von 1 qc (Slave) gemacht wird.

Hohe Faktor-Werte

In diesem Fall könnte der benutzte Faktor selbst (SYNCFACTS/SYNCFACTM) um $1/2^{32}$ falsch sein. Das heißt, dass im schlimmsten Fall jede ms ein Fehler von delta_master * $1/2^{32}$ auftritt. Angenommen, es wird ein Drehgeber mit 1000 Strichen (4000 qc) pro Umdrehung eingesetzt. Weiter angenommen, dass mit 2000 U/min gefahren wird, d.h. mit einer Geschwindigkeit von 133 qc/ms. Das bedeutet, dass ein Fehler von 133 * $1/2^{32}$ per ms gemacht wird. Daraus folgt, dass im schlimmsten Fall (maximaler Fehler jede ms immer in der gleichen Richtung) ein Fehler von 1 qc nach 9 Stunden entstehen könnte.

Das sollte in den meisten Anwendungen nicht relevant sein.

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.

MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

Master Units [MU]

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

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

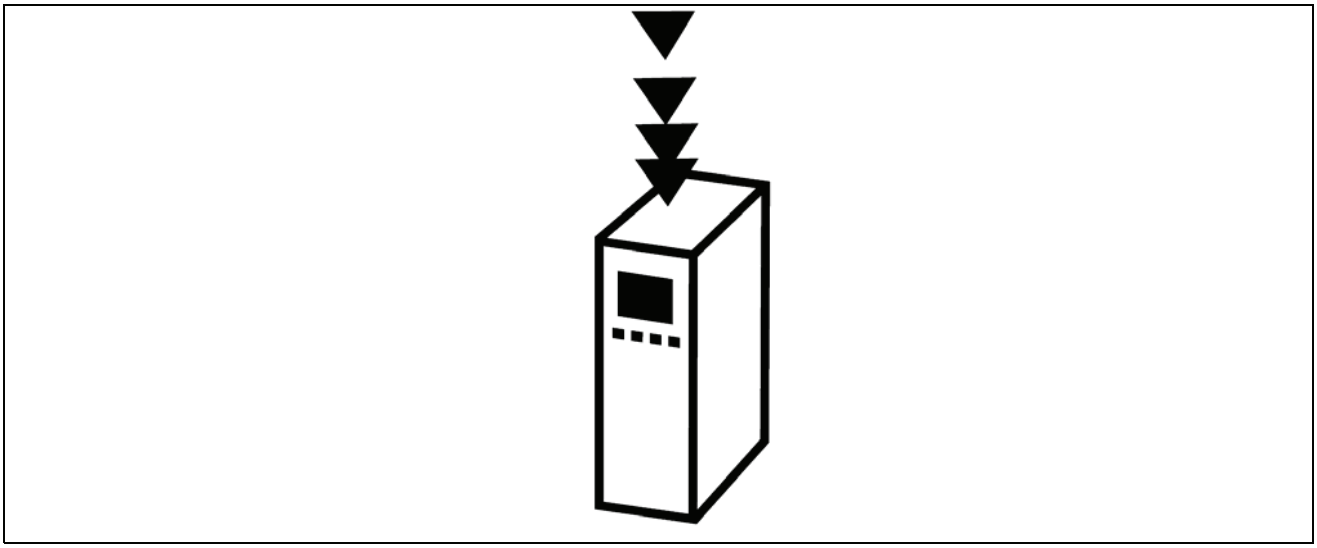
Par. 33-10 *Synchronisationsfaktor Master* SYNCFACTM
Par. 33-11 *Synchronisationsfaktor Slave* SYNCFACTS

□ 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: 5 V Inkremental RS422 und SSI absolut Single- und Multiturn, Gray Code, einstellbare Taktfrequenz und Datenlänge.
- 3 Versorgungsspannungen: 5 V, 8 V und 24 V.

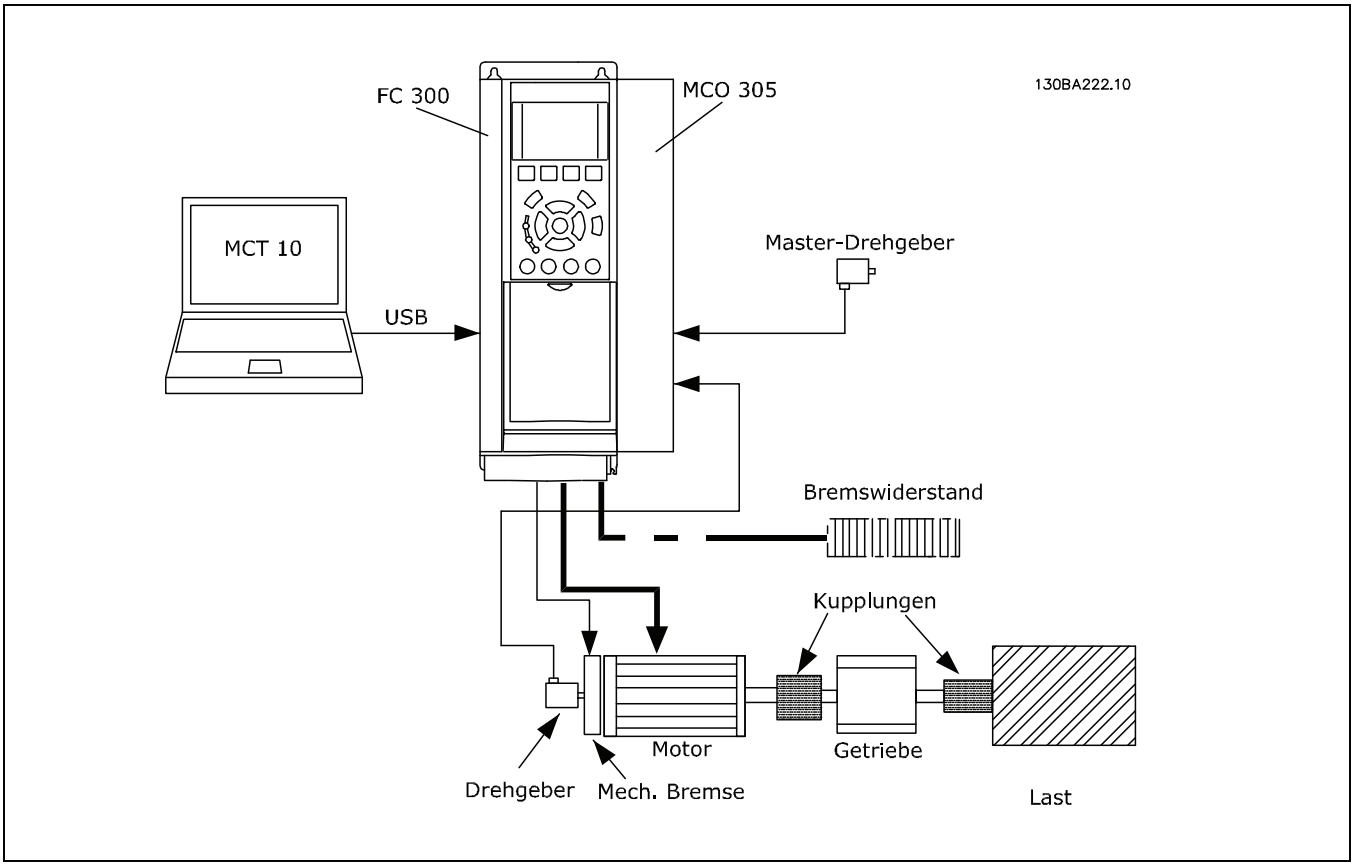
□ Systemüberblick

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:

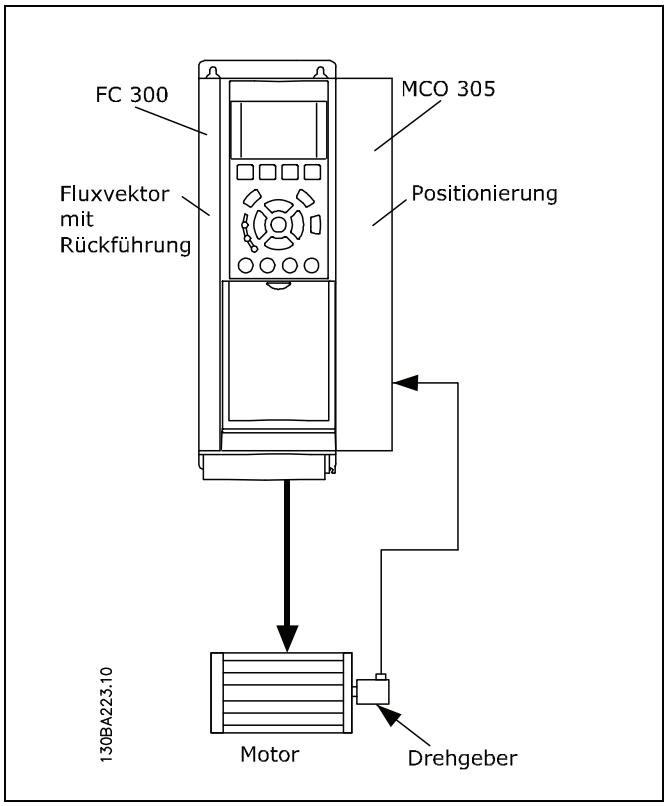
- Bremswiderstand für elektrische Bremsung
- Mechanische Bremse.



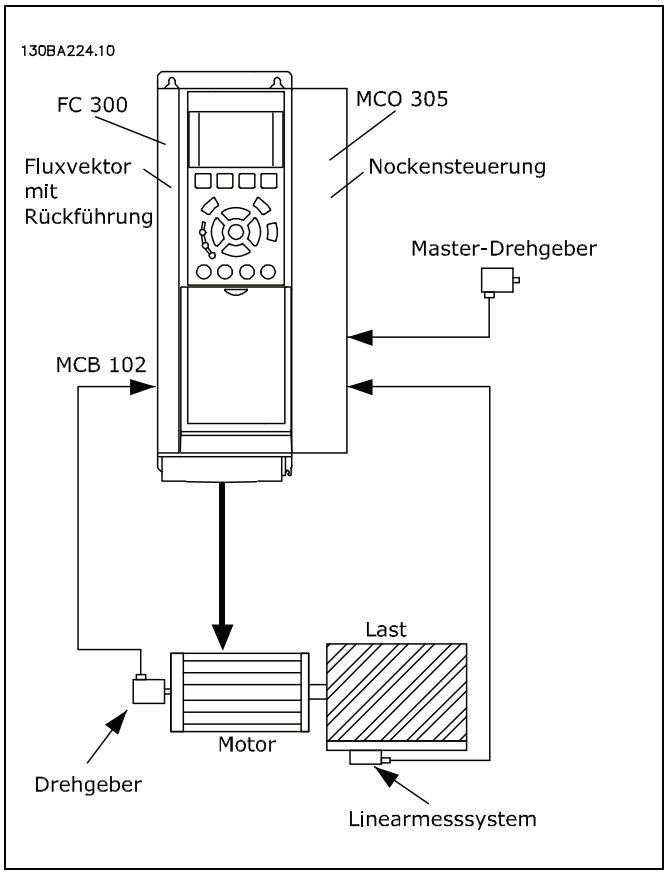
MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

□ 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

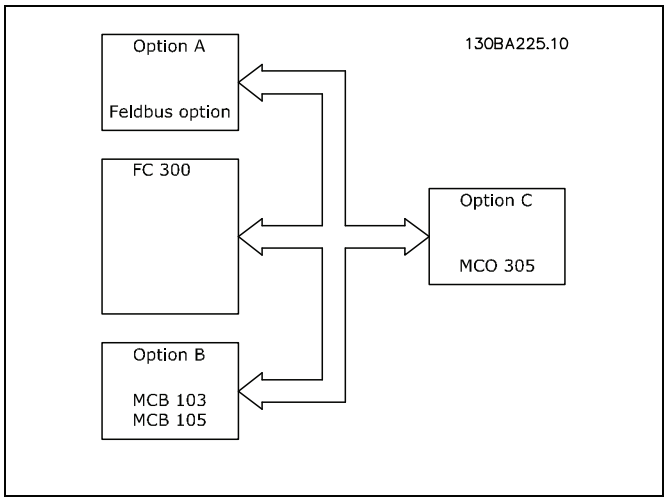
Die Schnittstelle zwischen einer MCO 305 und der FC 300 Steuerkarte ermöglicht sowohl das Lesen und Schreiben von allen Parametern als auch das Lesen des Status von allen Eingängen sowie die Steuerung von allen Ausgängen. Zusätzlich können verschiedene Prozessdaten wie das Statuswort und der aktuelle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

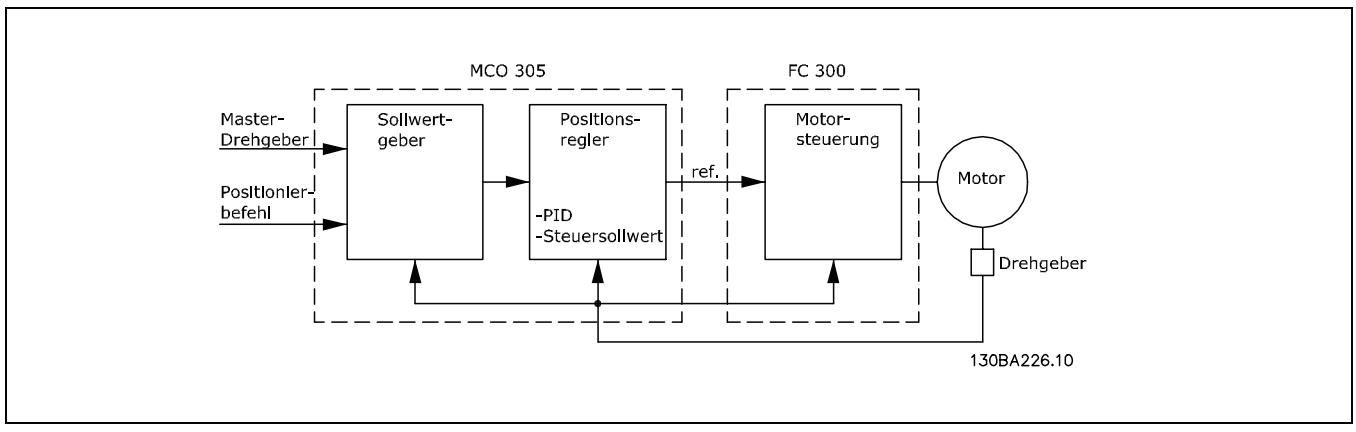
Mehrzweck-I/O-Option MCB 103: Mit dem MCO 305 Anwendungsprogramm kann der Status der Eingänge gelesen und können die Ausgänge gesteuert werden.



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.

MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

□ 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.

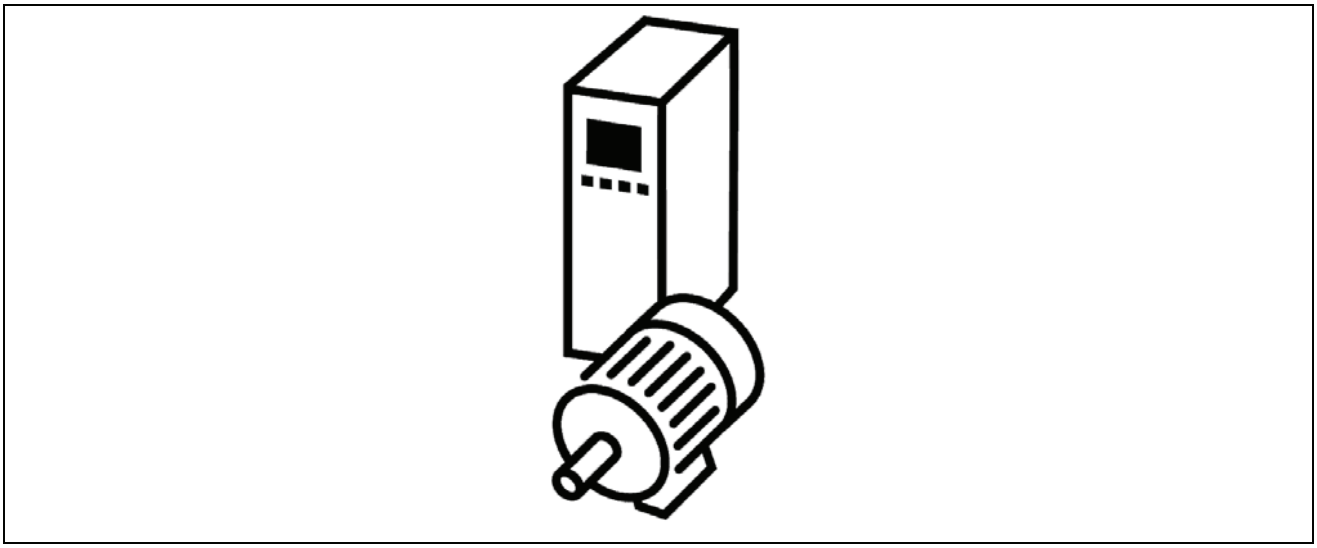
Ein temporäres Programm kann aus dem Editor (MCT10/APOSS) heraus ausgeführt werden. Temporäre Programme werden nur im RAM gespeichert und sind daher nach dem Ausschalten verloren. Das temporäre Programm kann auch in einem speziellen Debug-Modus ausgeführt werden, in dem es möglich ist, die Programmausführung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

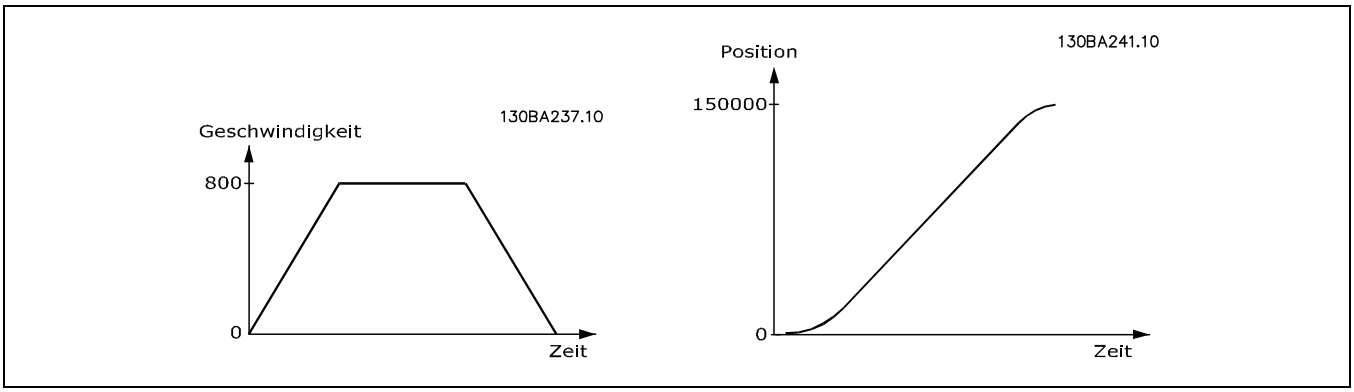
MCO 305 bietet drei Hauptpositionierungsarten:

- 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.

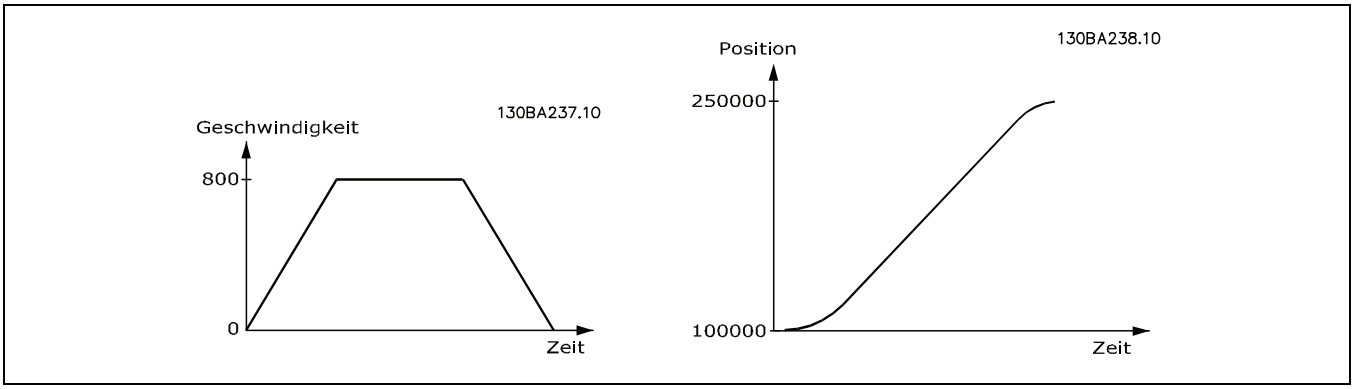
Wenn die Startposition 0 ist und bei einer absoluten Positionierung auf 150.000 die Zielposition 150.000 ist, wird der Antrieb also eine Distanz von 150.000 zurücklegen. Falls andererseits die Startposition 100.000 ist, bleibt bei einer absoluten Positionierung auf 150.000 die Zielposition weiterhin 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.

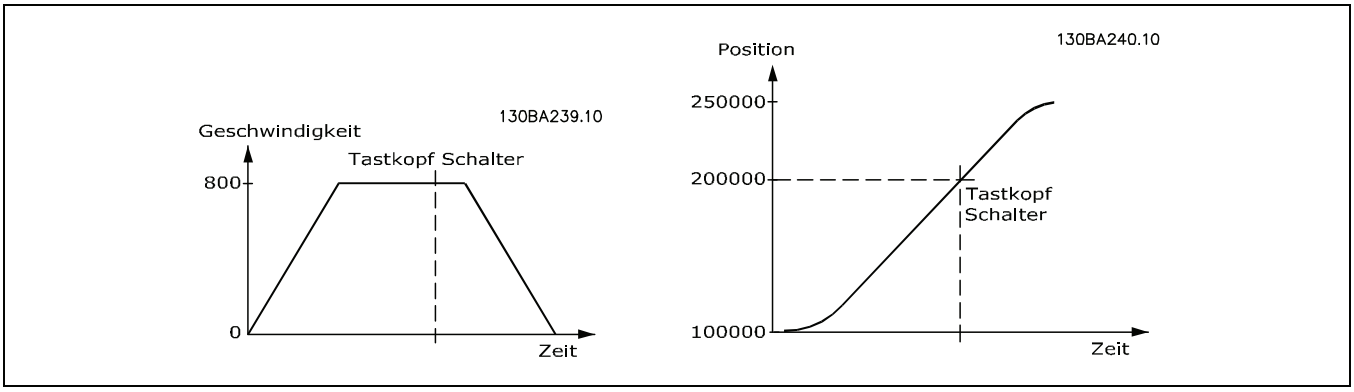


Touch-Probe Positionierung

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.



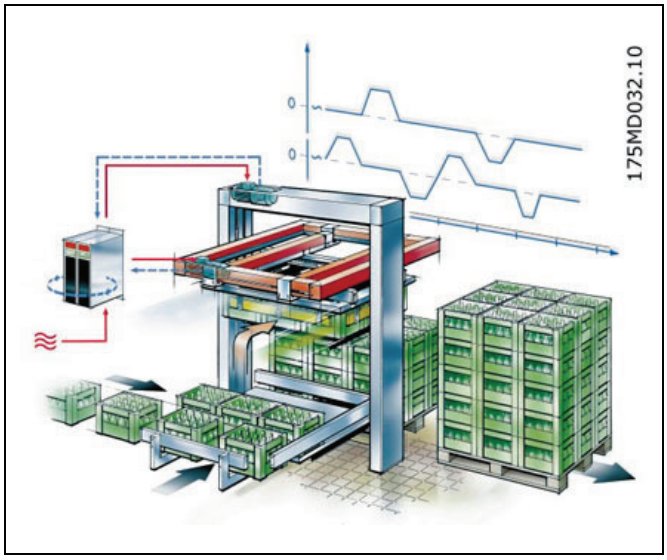
MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

□ 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:

32-0*	Drehgeber 2 – Slave	Seite 204
32-6*	PID-Regelung	Seite 214
32-8*	Geschwindigkeit & Beschleunigung	Seite 217
33-0*	Homefahrt	Seite 220
33-4*	Grenzwertbehandlung	Seite 233

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  ***********************************/
```


MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

◻ 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:

32-0*	Drehgeber 2 – Slave	Seite 204
32-6*	PID-Regelung	Seite 214
32-8*	Geschwindigkeit & Beschleunigung	Seite 217

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 204
32-6*	PID-Regelung	Seite 214
32-8*	Geschwindigkeit & Beschleunigung	Seite 217
33-4*	Grenzwertbehandlung	Seite 233

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	–	–

MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

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)

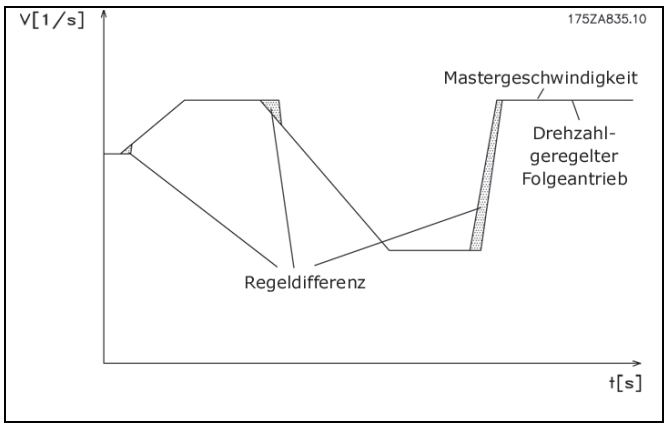
Die Geschwindigkeitssynchronisation (SYNCV) ist eine Geschwindigkeitssteuerung im geschlossenen Regelkreis, bei der die Mastergeschwindigkeit multipliziert mit dem Getriebefaktor der Positions-Sollwert ist und die aktuelle Geschwindigkeit durch den Slave-Drehgeber gemessen wird; Positionsabweichungen 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:

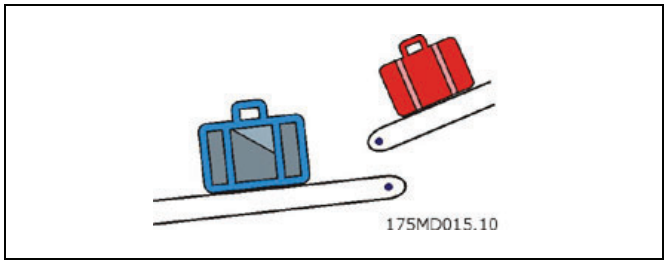
- Synchronisieren von zwei oder mehr Transportbändern
- Strecken von Materialien
- Mischen



Regelungsverhalten bei Geschwindigkeitssynchronisation.

MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

☐ 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:

32-0* Drehgeber 2 – Slave	Seite 204
32-3* Drehgeber 1 – Master	Seite 209
32-6* PID-Regelung	Seite 214
32-8* Geschwindigkeit & Beschleunigung	Seite 217
33-1* Synchronisation	Seite 221

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 ********************************/
```


MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

□ Position/Winkel-Synchronisation (SYNCP)

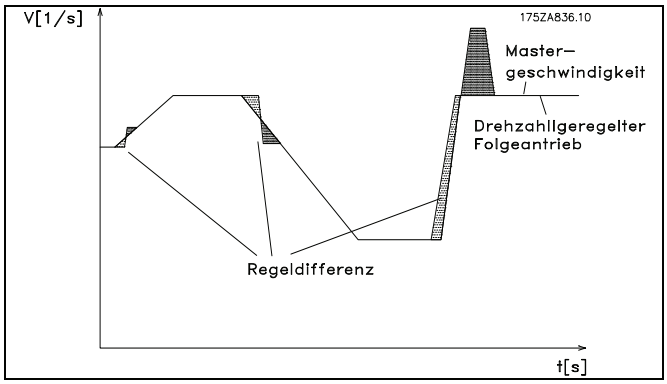
Eine Positionssynchronisation (SYNCP) ist eine Positionsregelung mit Rückführung eines bewegten Ziels, wobei der Sollwert (Sollposition) die Master-Position multipliziert mit der Getriebeübersetzung ist und ein jeder Positionsoffset berücksichtigt wird. Die Slave-Position wird basierend auf diesen Sollwert 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.

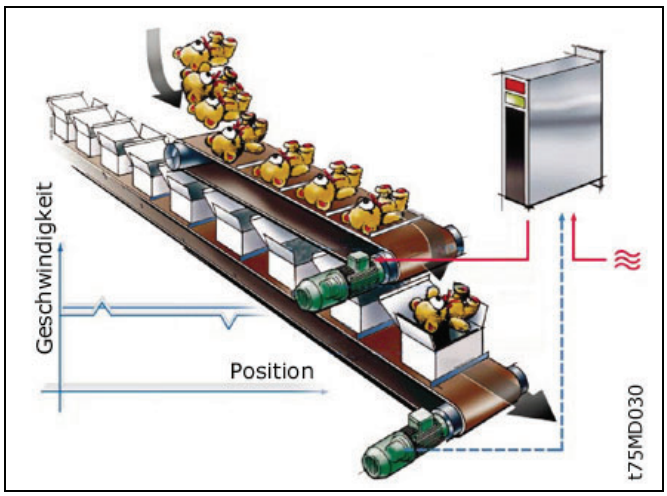


Regelungsverhalten bei Positionssynchronisation

□ 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:

32-0* Drehgeber 2 – Slave Seite 204
32-3* Drehgeber 1 – Master Seite 209
32-6* PID-Regelung Seite 214
32-8* Geschwindigkeit & Beschleunigung Seite 217
33-1* Synchronisation Seite 221

Befehl	Beschreibung	Syntax	Parameter
DEFSYNCORIGIN	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  ******************************/
```


MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

```
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 *********************************/
```


MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

☐ 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.

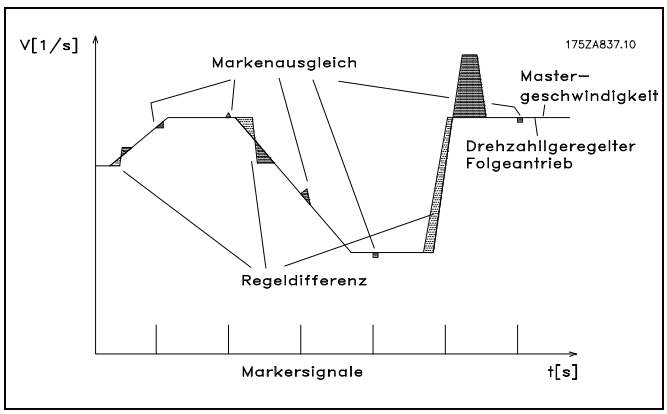
Eine Markersynchronisation verhält sich nach dem Starten anfangs wie eine Positionssynchronisation, aber sobald der erste Satz der Marker erkannt wurde, startet die Markerkorrektur. Welche Marker für die erste Markerkorrektur benutzt werden, wird in Parameter 33-23 festgelegt. Durch die Definition des 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

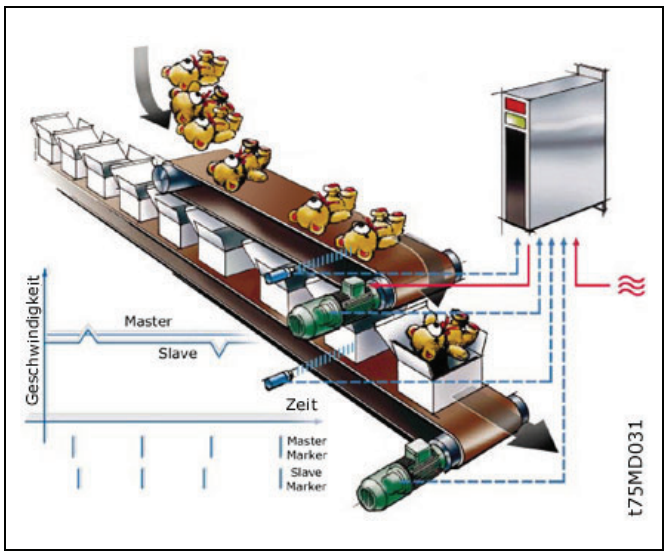
Diese Anwendung besteht aus zwei Transportbändern, eines befördert leere Kartons und das andere die Teddybären. Aufgabe der Anlage ist es, die Teddybären in die Kartons zu packen. Beide, Kartons und Teddys werden durch Reibung befördert und können sich daher auf dem Transportband bewegen. Das bedeutet, 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:

32-0* Drehgeber 2 – Slave	Seite 204
32-3* Drehgeber 1 – Master	Seite 209
32-6* PID-Regelung	Seite 214
32-8* Geschwindigkeit & Beschleunigung	Seite 217
33-1* Synchronisation	Seite 221

Befehl	Beschreibung	Syntax	Parameter
DEFSYNCORIGIN	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 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	–

MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

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 ********************************/
```


MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

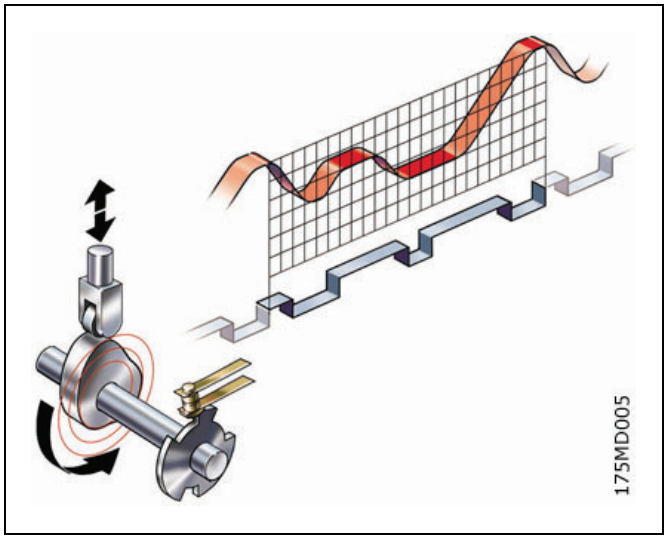
□ Kurvenscheibensteuerung (CAM-Modus)

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

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 in der Registerkarte → *Kurvendaten* zwischen mehreren 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.

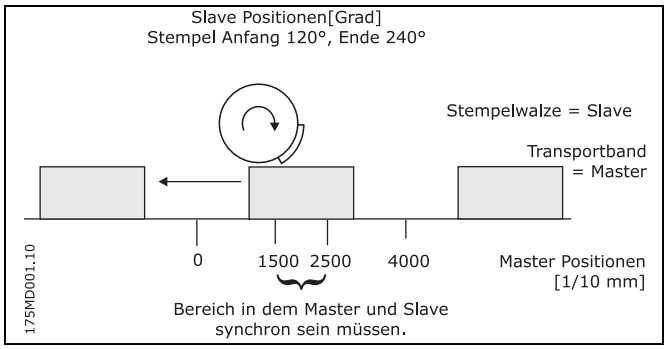
Interne Realisation als Array

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 zbc- (oder cnf)-Datei aus; APOSS und damit die ausgewählte Datei werden daraufhin automatisch im *CAM-Editor* geöffnet.

   APOSS stand-alone: Starten Sie den → *CAM-Editor* und öffnen diese zbc- (oder cnf)-Datei.
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

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

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

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

   Tragen Sie diese ganzzahligen Werte ein in die Registerkarte → *Gebersystem*:
   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 → *Kurvendaten* 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.

Fix Punkte | Einfügen

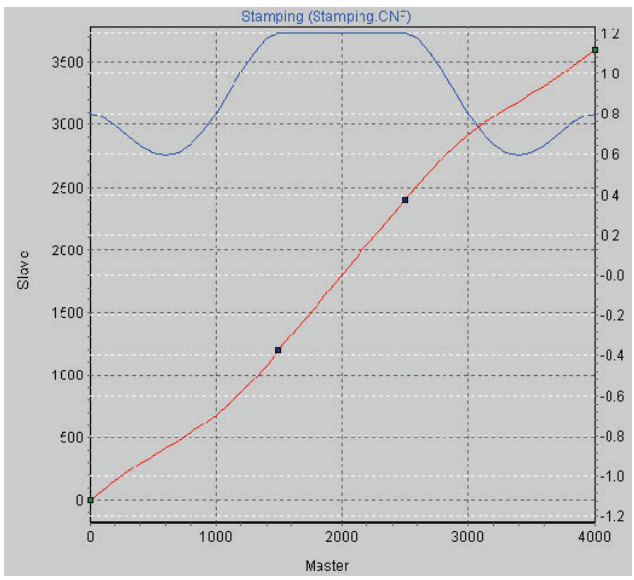
Punkt	Master	Slave	Typ
1	0	0	K
2	1500	1200	K
3	2500	2400	K
4	4000	3600	K

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 → *Tangente* aus. 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 → *Kurveninfo* 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 zbc-Datei speichern; klicken Sie dazu auf → *Sichern als*.
    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 zbc-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
```

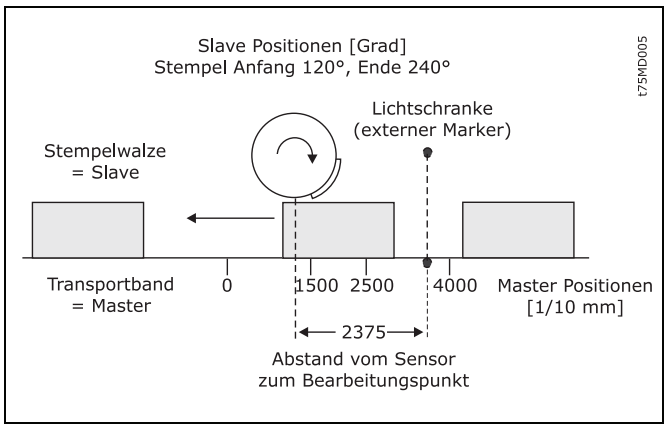
MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

□ 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.

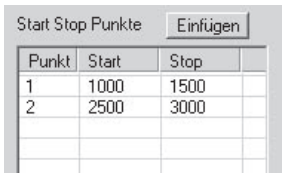
Im Folgenden wird beschrieben, wie Sie die Kurve des vorgehenden Beispiels für diese Anwendung anpassen.

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 werden und bis zum Ende des Kartons ausgekuppelt sein.



11. Bestimmen Sie in der Registerkarte → *Kurvendaten* 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.

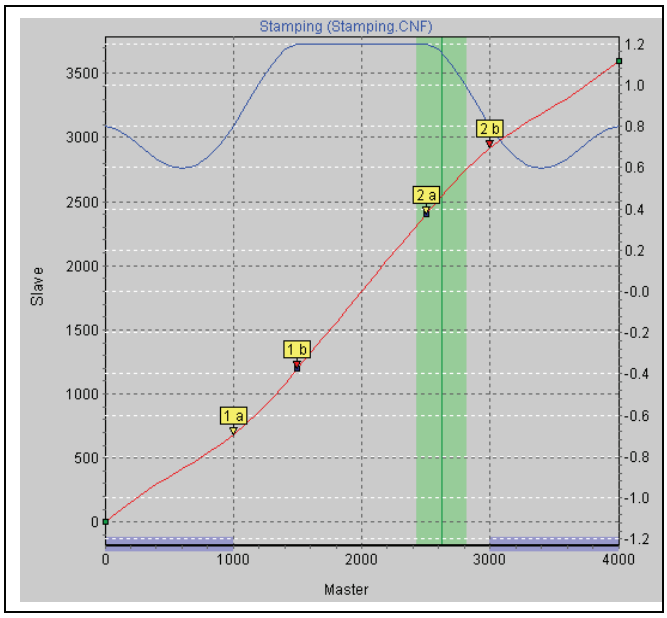
Par. 33-17	*Markerabstand Master*	= 2375
Par. 33-21	*Master-Marker Toleranzfenster*	= 200
Par. 33-19	*Markertyp Master*	= 2

    Tragen Sie die Master-Position in der *Registerkarte* → *Kurvendaten* 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 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 Karton beschädigen. Und die Korrektur muss vollständig beendet sein, wenn der nächste Karton den Bearbeitungspunkt erreicht.



In diesem Beispiel sind die Master-Positionen Ende und Anfang eines Kartons gut geeignet:

Korrektur Start = 3000

Korrektur Ende = 1000Tragen Sie die Werte in die Registerkarte → *Kurvendaten* 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* um die Datei zu speichern, zum Beispiel „marker”.
16. Laden Sie die zbc-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:

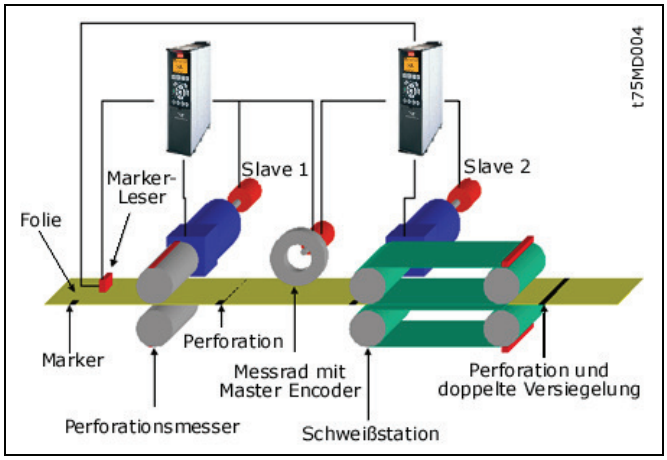
```
DIM marker[112]     // Anzahl der Elemente aus Titelleiste des CAM-Editors
HOME          // Slave Achse führt eine Homefahrt durch (Schalter für Nullstellung oben)
              // Danach befindet sich der Slave in der Nullposition (0 Grad)
              // (entfällt, falls ein Absolutgeber eingesetzt 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
```


MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

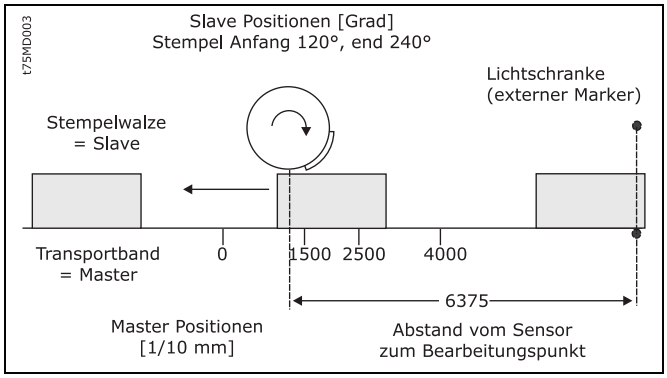
☐ Wenn der Abstand des Sensors größer als eine Masterzykluslänge ist

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.



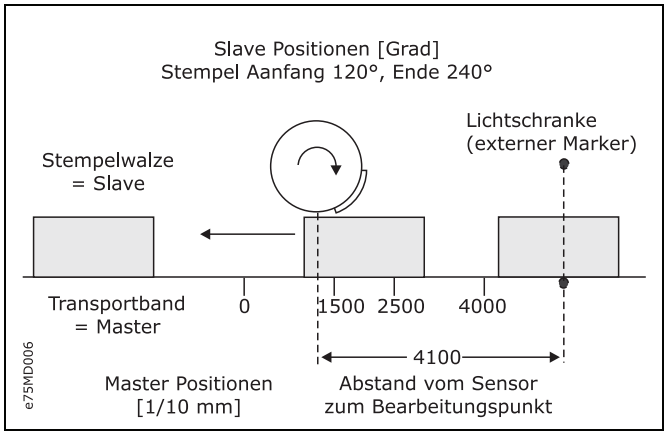
Um zu beurteilen, in welchem Bereich korrigiert werden darf, subtrahieren Sie so oft die Masterzykluslänge, bis der Wert < 1 Masterzykluslänge ist. Dies ist der maximal erlaubte Abstand zum Korrigieren. In diesem Beispiel ist dieser also 6375 – 4000 = 2375 und damit der gleiche Korrekturbereich wie im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



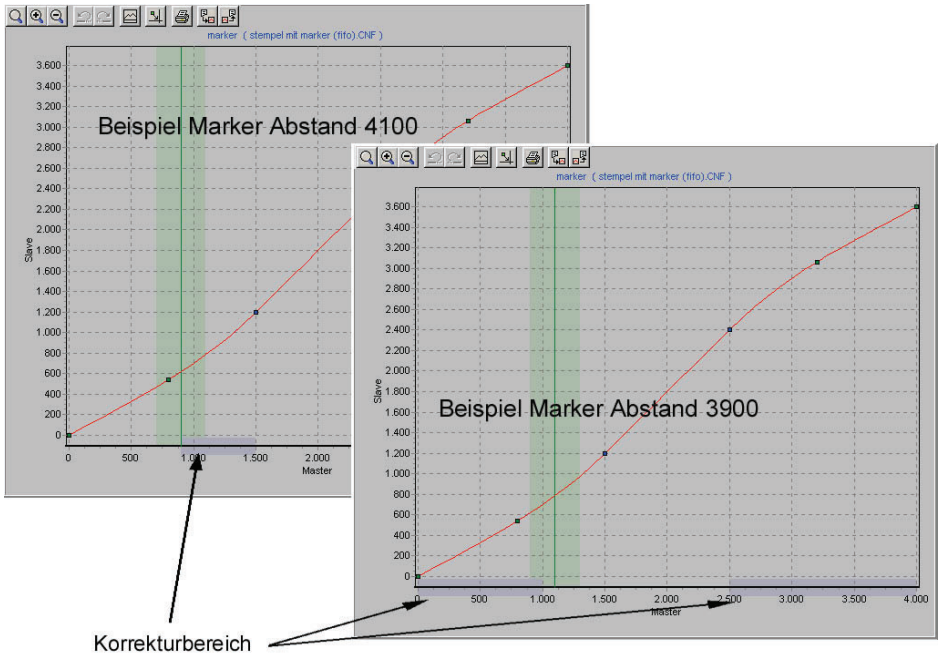
Problemfälle bei der Festlegung des Markerabstandes

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 → *Kurvendaten* ein:

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

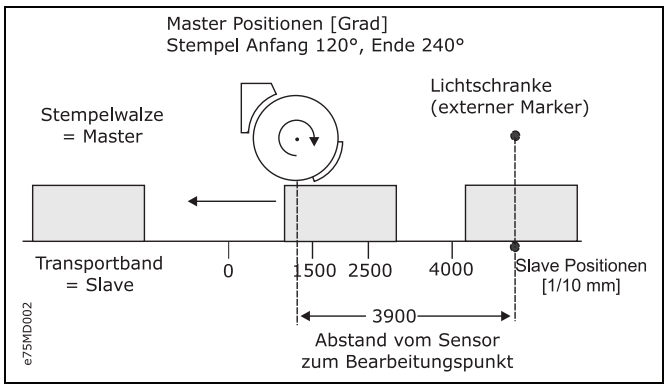
Da nun kein Puffer erzeugt wird, könnte man zum Beispiel von 2500 bis 800 korrigieren.

MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

☐ 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 „zbc" sichern.
2. Diese zbc-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 → *Kurvendaten* 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.

Fix Punkte | Einfügen

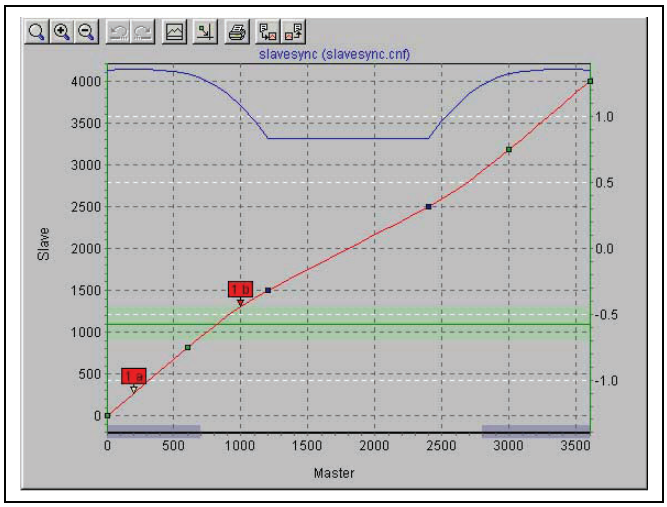
Punkt	Master	Slave	Typ
1	0	0	K
2	1200	1500	K
3	2400	2500	K
4	3600	4000	K

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.

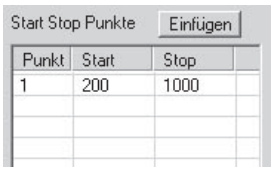
Fix Punkte | Einfügen

Punkt	Master	Slave	Typ
1	0	0	K
2	1200	1500	T
3	2400	2500	T
4	3600	4000	K

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



8. Tragen Sie in der Registerkarte → *Kurveninfo* 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 → *Kurvendaten* 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 → *Kurvendaten* 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 → *Kurvendaten* 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* zum Speichern.
16. Laden Sie die zbc-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-Stopp-Punktepaar 1 einkuppeln
```


MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

◻ Nockenschaltwerk

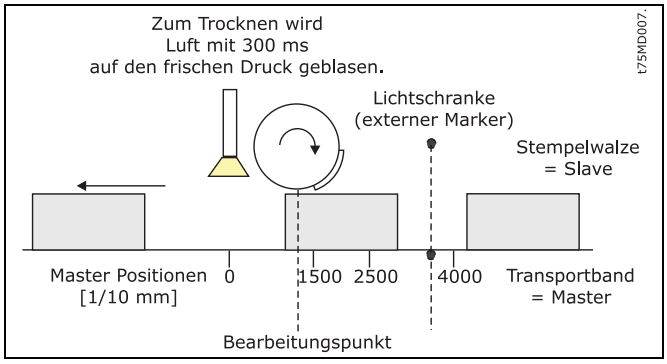
Die mechanische Nockenwelle wird ebenfalls durch eine (oder mehrere) Kurven nachgebildet. Um ein Nockenschaltwerk zu realisieren, muss es möglich sein, den Slave immer wieder an bestimmten Master-Positionen ein- und auszukuppeln.

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-Stopp-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:

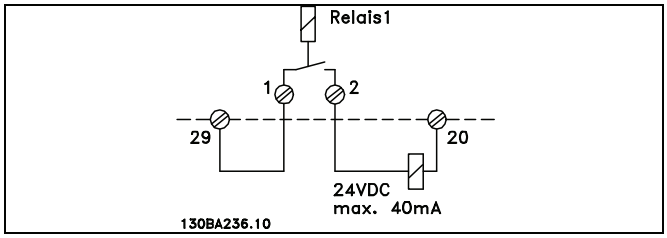



```
ON MCPOS 2500 GOSUB trocknen          // Unterprogramm aufrufen, wenn die
                    // Master-Position 2500 in positiver Richtung  passiert wurde
SUBMAINPROG
    SUBPROG trocknen
    OUT 1 1                           // Trockner einschalten
    DELAY 300                         // 300 ms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


MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

```
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

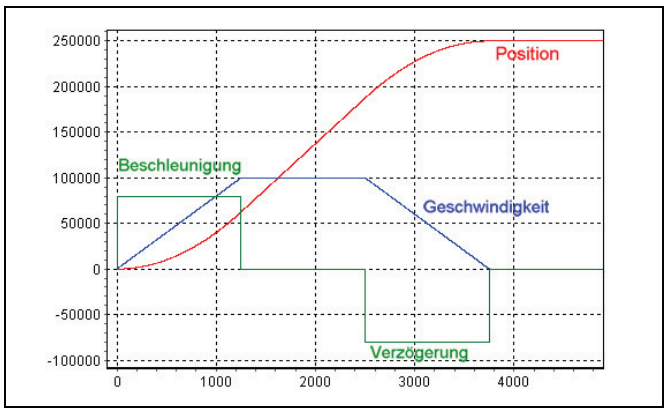
□ Wie ruckbegrenzte Bewegungen funktionieren

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.

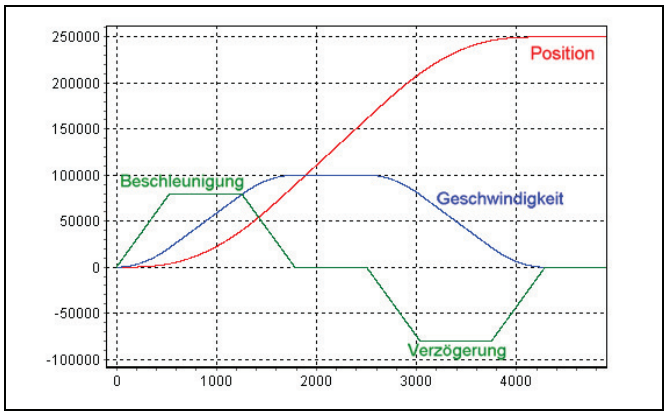
Typische Anwendungen, die ruckfreie Bewegungen erfordern, sind:

- 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.

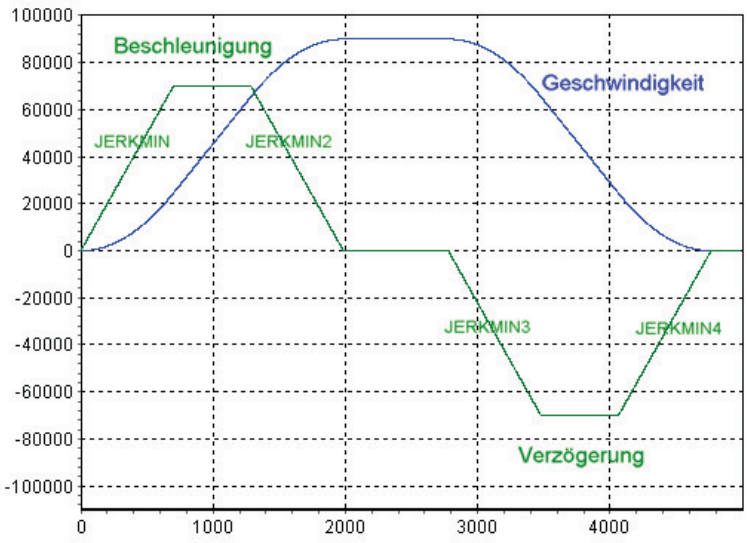
JERKMIN2: Konstante Rücknahme der Beschleunigung. Dies definiert die Zeitspanne [ms] in der die maximale Beschleunigung auf 0 Beschleunigung reduziert werden soll (d.h. normalerweise auf konstante maximale Geschwindigkeit). Wenn „0" gesetzt ist, wird der gleiche Wert wie bei JERKMIN verwendet.

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.

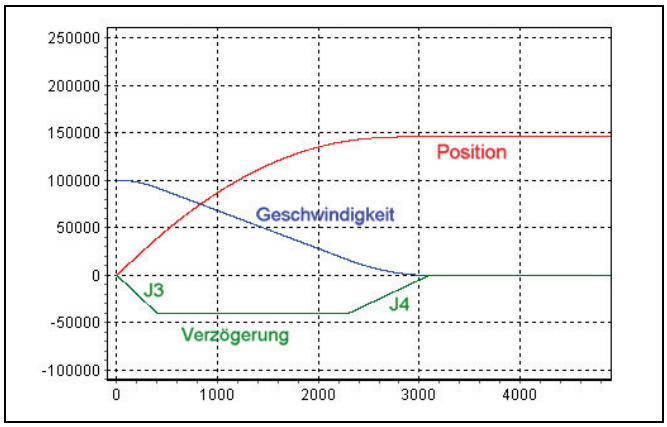
□ Beispiele

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

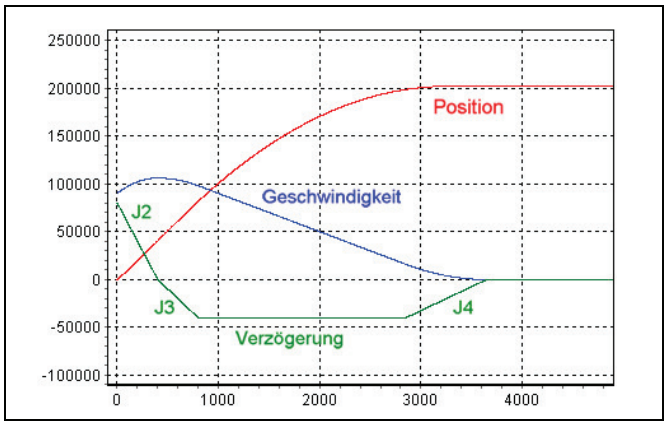
Dieses Diagramm zeigt eine Stopp-Bewegung, die mit einer positiven konstanten Geschwindigkeit beginnt.

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).



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).

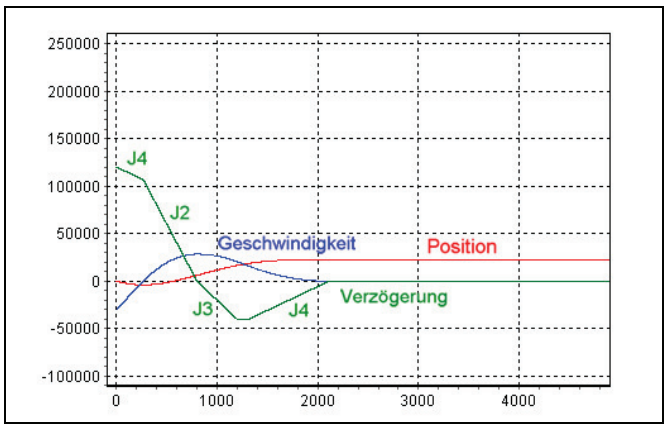


MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

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.

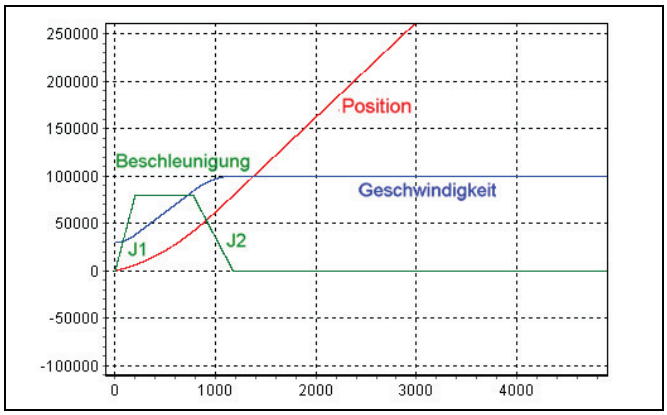
Demzufolge wird die Kurve mit einer Rücknahme der Beschleunigung fortgesetzt (JERKMIN2) bis Beschleunigung 0 erreicht ist. Der Motor fährt nun mit einer konstanten positiven Geschwindigkeit und daher wird die Kurve ganz normal mit einer Verzögerungsrampe (JERKMIN3), einem Segment konstanter Verzögerung (sehr kurz in diesem Beispiel) und einer Zunahme der Verzögerung auf Geschwindigkeit 0 (JERKMIN4) beendet.



In eine konstante Geschwindigkeit wechseln

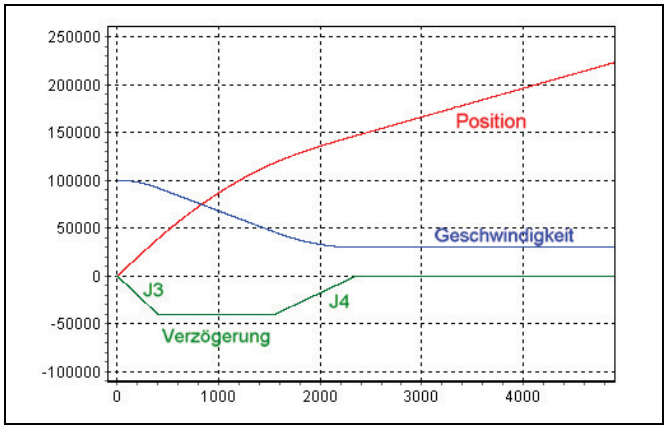
Das folgende Diagramm zeigt eine Bewegung, die mit einer positiven konstanten Geschwindigkeit beginnt und diese auf eine höhere positive konstante Geschwindigkeit steigert.

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.



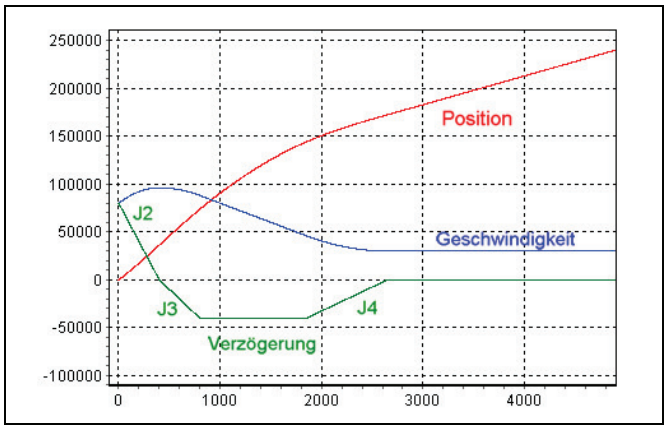
Das nächste Diagramm zeigt eine Bewegung, die mit einer hohen positiven konstanten Geschwindigkeit beginnt und seine Geschwindigkeit auf eine niedrigere positive konstante Geschwindigkeit verringert.

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.



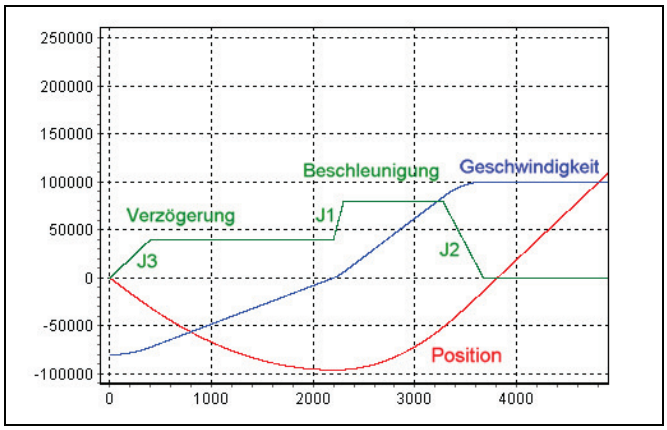
Das nächste Diagramm ist ähnlich dem vorhergehenden, außer dass es mit einer positiven Beschleunigung beginnt.

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.



Das folgende 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).



Auf eine definierte Position fahren

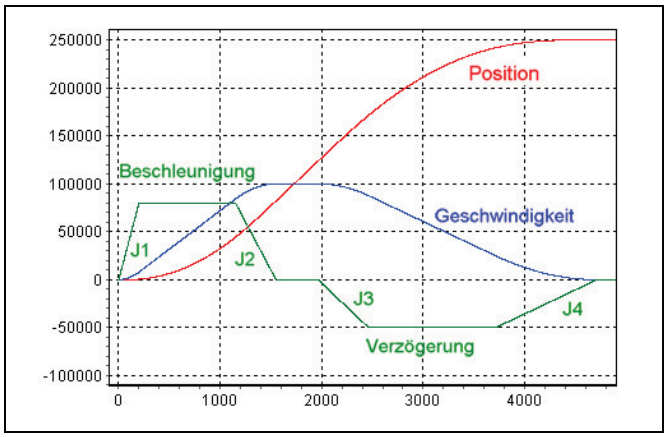
Das folgende Diagramm zeigt eine „normale" Bewegung, die von einer Position, an der sie gestoppt hatte, vorwärts fährt, um an einer anderen Position anzuhalten. Die Kurve startet mit einer Verzögerung auf maximale Geschwindigkeit. Dieser Teil der Kurve ist ähnlich der ersten in *„In eine konstante Geschwindigkeit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

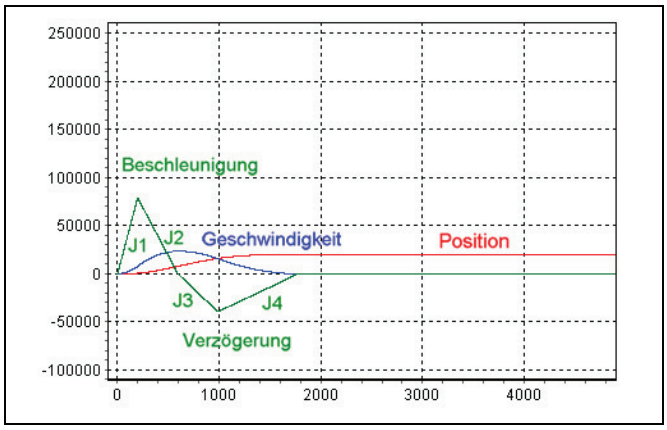
MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

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.



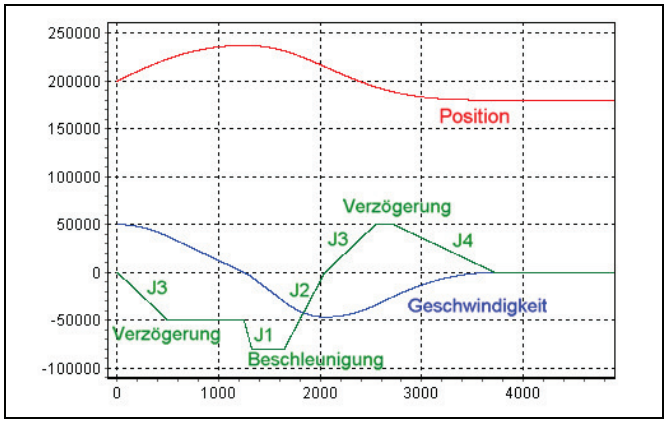
Das nächste Diagramm zeigt eine typische „kurze" Bewegung, bei der die maximale Geschwindigkeit nicht erreicht werden kann. In diesem Fall wird so lange wie möglich mit einer Beschleunigung gefahren (JERKMIN). Abhängig davon, wie weit entfernt die Zielposition ist, kann dabei die maximale Beschleunigung 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.



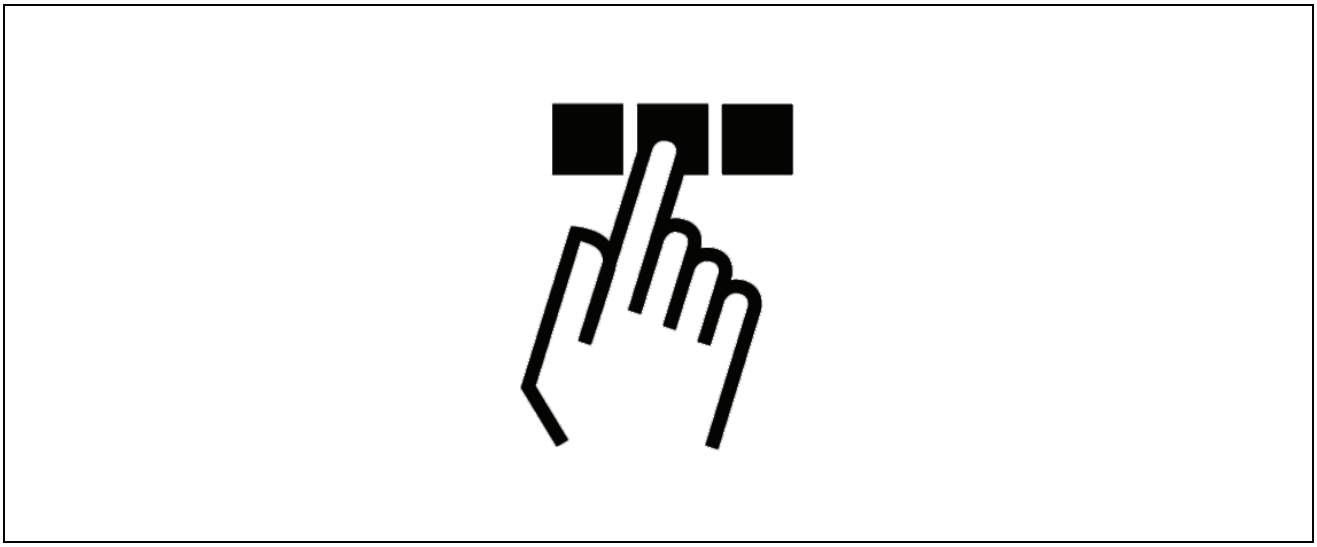
Das nächste Diagramm zeigt ein Beispiel bei dem der Motor anfänglich in die „falsche" Richtung fährt, umgedreht werden und „zurück" auf die Zielposition fahren muss. Weil er „umgedreht" werden muss, startet die Kurve mit einer Verzögerungsrampe (JERKMIN3) bis zur maximalen Verzögerung. Dadurch wird die 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 besteht aus folgenden Segmenten:

- Beschleunigungsrampe (Rückwärtsfahrt)
- ggf. konstante Beschleunigung
- Beschleunigungsrampe
- ggf. konstante Geschwindigkeit
- Verzögerungsrampe
- ggf. konstante Verzögerung
- Verzögerung zum Stoppen auf der Zielposition.



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.

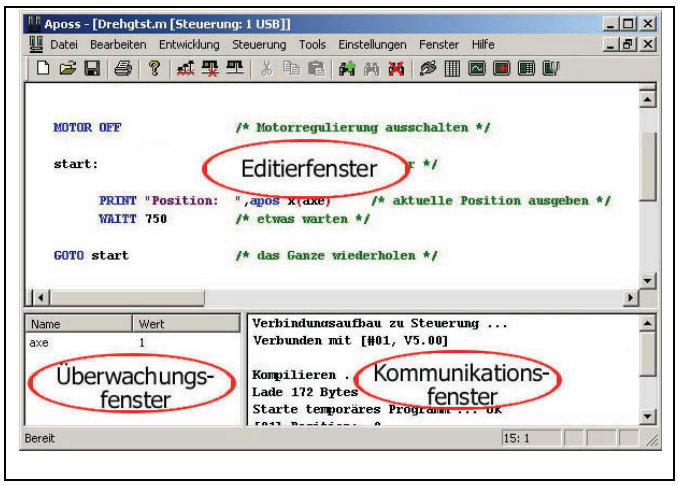
Wenn APOSS von MCT 10 aus gestartet wird, wird nur ein Antrieb verbunden. Daher sind alle Funktionen, mit denen APOSS Antriebe steuern oder mehrere Antriebe verbinden kann gesperrt.

Wenn MCT 10 weder online noch offline benutzt wird, wird APOSS stand-alone betrieben.

□ Das APOSS Fenster

Das APOSS-Fenster erlaubt den gleichzeitigen Zugriff zu einem APOSS-Programm und einer Steuerung. Sie können mehrere APOSS-Fenster öffnen und jedes mit einem anderen APOSS-Programm und Steuerung verbinden.

Das *Editierfenster* im oberen Bereich zeigt das APOSS-Programm während des Editierens und bietet alle üblichen Funktionen eines Text-Editors. Verschiedene Farben erleichtern Ihnen die Unterscheidung zwischen Kommentaren, Programmteilen, Operatoren, Ziffern usw. Sie können die Farbzuordnung mit *Einstellungen* → *Editor* ändern.



Das *Kommunikationsfenster* zeigt sowohl die Meldungen der APOSS-IDE (z.B. Meldungen des Compilers) als auch Meldungen der Steuerung (z.B. programmierte PRINT Befehle). Den Meldungen der Steuerung wird die ID-Nummer der Steuerung (z.B. „[01]" wie in obigen Beispiel) vorangestellt.

Das *Überwachungsfenster* links unten im APOSS-Fenster ist besonders nützlich beim Debuggen. Damit kann sowohl die Steuerung als auch das ausführende Programm während die Steuerung läuft überwacht werden. Was überwacht werden soll, können Sie mit *Entwicklung* → *Überwachung hinzufügen* einstellen.

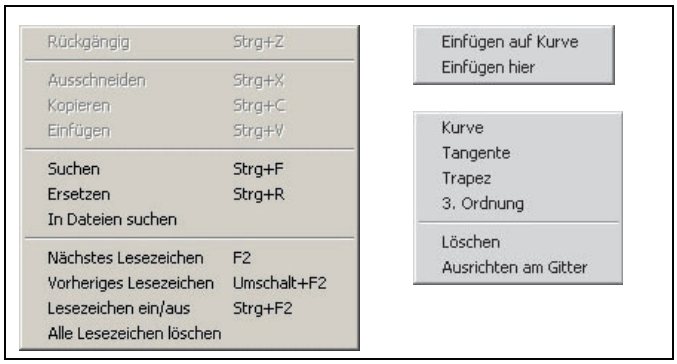
Titelleiste und Symbolleiste

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.

Die *Symbolleiste* bietet einen schnellen Zugriff auf häufig benutzte Funktionen:

Kontextmenüs

An manchen Programmstellen werden Kontextmenüs angeboten, wenn Sie auf die rechte Maustaste klicken. Zum Beispiel im *Editierfenster* oder im *CAM-Editor* zum Einfügen oder Löschen von Fixpunkten. Die Kontextmenüs 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.



MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

Funktionstasten

Häufig benötigte Funktionen sind auf die Funktionstasten gelegt:

[F1] Online-Hilfe

[F2] Zum nächsten Lesezeichen springen.

[F3] Weitersuchen (beim Suchen)

[F4] Syntax des Programms prüfen

[F5] Programm ausführen

[F9] Programm zeilenweise ausführen (nur im Debug Modus)

[F11] System-Prozessdaten in der Online-Hilfe aufrufen

[F12] Befehlshilfe aufrufen

Die anderen Funktionstasten werden an den entsprechenden Stellen erwähnt.

[Esc]-Taste

In Standard Windows-Anwendungen schließt die [Esc]-Taste normalerweise das aktive Fenster. Im APOSS-Fenster jedoch öffnet die [Esc]-Taste automatisch eine Verbindung zu einer Default-Steuerung, wenn keine Steuerung angeschlossen ist.

Wenn schon eine Steuerung angeschlossen ist, beendet die [Esc]-Taste alle gerade laufenden Programme der Steuerung.

ACHTUNG!:
Wenn ein laufendes Programm abgebrochen wird, während sich der Antrieb dreht, wird der Antrieb mit der maximal zulässigen Geschwindigkeit abgebremst!

Eine unterbrochene Verbindung wiederherstellen

Wenn eine aktive Verbindung zu einer Steuerung unterbrochen ist (zum Beispiel, wenn die Steuerung ausgeschaltet oder die Kommunikation getrennt wurde), meldet die Titelleiste des APOSS-Editierfensters, das mit dieser Steuerung verbunden war, „Verbindung unterbrochen". Aber das Editierfenster bleibt mit dieser Steuerung „verbunden". Wenn die [Esc]-Taste gedrückt wird (oder irgendein Befehl ausgeführt werden soll, der eine Kommunikation zur Steuerung erfordert) wird APOSS versuchen, die gleiche Steuerung wieder zu verbinden. Dieses Verhalten ist nur relevant, wenn mehrere Steuerungen in einem Kommunikationsstrang vorhanden sind, z.B. CAN-Bus.

□ Das Editierfenster

Das Editierfenster zeigt das Programm, das gerade bearbeitet wird. Hierfür stehen viele Mausfunktionen sowie Shortcuts zur Verfügung.

Wenn Sie mit den Mauscursor auf einen Befehl oder eine Funktion zeigen, wird die Syntax des Befehls oder der Funktion in einem kleinen Popup-Fenster eingeblendet, Wenn Sie **F1** drücken, erhalten Sie das entsprechende Hilfethema dazu.

Maus-Funktionen

Das Editierfenster unterstützt folgende Maus-Funktionen:

Linksklick	Cursor-Position ändern und vorherige Auswahl aufheben.
Rechtsklick	Kontextmenü *Bearbeiten* öffnen.
Links-Doppelklicken	Das Wort unter dem Cursor markieren.
Linke Maustaste drücken und ziehen	Text markieren
[Alt] + linke Maustaste drücken und ziehen	Textspalte markieren.
Mit linker Maustaste auf die Auswahl zeigen, drücken und ziehen.	Markierten Text verschieben.
[Strg] + mit linker Maustaste auf die Auswahl zeigen, drücken und ziehen.	Markierten Text kopieren.
Linksklick im linken Rand	Die ganze Zeile markieren.
Linksklick im linken Rand, drücken und ziehen	Mehrere Zeilen markieren.
Scrollrad drehen	Fenster vertikal scrollen
Klicken mit Scrollrad	Das Wort unter dem Cursor markieren.
Doppelklicken mit Scrollrad	Die ganze Zeile markieren.
Linksklick auf die Trennleiste (Splitter), drücken und ziehen.	Das Fenster in mehrere Ansichten teilen oder die aktuelle Ansicht eines geteilten Fensters ausrichten.
Links-Doppelklicken auf die Trennleiste.	Das Fenster in zwei Hälften teilen oder, wenn bereits geteilt, die Teilung wieder aufheben.

Tastatur-Funktionen

Das Editierfenster unterstützt folgende Funktionen mit Shortcuts. Beachten Sie, dass viele dieser Funktionen nur mit Tastatur-Shortcuts verfügbar sind.

ACHTUNG!:
Manche der Tastatur-Funktionen hängen von spezifischen Einstellungen Ihrer Tastatur ab. Bei unerwünschten Effekten fragen Sie bitte Ihren Systemadministrator.

Gehe zu	
[Pos1]	Gehe zum Zeilenanfang
[Ende]	Gehe zum Zeilenende
[Strg]+[Pos1]	Gehe zum Programmanfang
[Strg]+[Ende]	Gehe zum Programmende
[Strg]+[←]	Gehe zum Wortanfang
[Strg]+[□]	Gehe zum Wortende
[Strg]+[Alt]+[□]	Gehe zum Anfang der nächsten leeren Zeile
[Strg]+[Alt]+[←]	Gehe zum Ende der vorhergehenden leeren Zeile
[Strg]+[G]	Gehe zu Zeile … (Dialog öffnen)
[Strg]+[B]	Gehe zu Klammerpaaren (("{([" oder "})]")

MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

[F2]	Gehe zum nächsten Lesezeichen
[Umschalt]+[F2]	Gehe zum vorherigen Lesezeichen
[Strg]+[F2]	Lesezeichen in der aktuellen Zeile aus-/einschalten.
Text Auswahl	
[Umschalt]+[←]	Auswahl nach links erweitern
[Umschalt]+[□]	Auswahl nach rechts erweitern
[Umschalt]+[↑]	Auswahl nach oben erweitern
[Umschalt]+[↓]	Auswahl nach unten erweitern
[Strg]+[Umschalt]+[←]	Auswahl zum Anfang des Wortes erweitern
[Strg]+[Umschalt]+[□]	Auswahl zum Ende des Wortes erweitern
[Umschalt]+[Pos1]	Auswahl zum Zeilenanfang erweitern
[Umschalt]+[Ende]	Auswahl zum Zeilenende erweitern
[Umschalt]+[Bild↑]	Auswahl um eine Seite nach oben erweitern
[Umschalt]+[Bild↓]	Auswahl um eine Seite nach unten erweitern
[Strg]+[Umschalt]+[Pos1]	Auswahl zum Programmanfang erweitern
[Strg]+[Umschalt]+[Ende]	Auswahl zum Programmende erweitern
[Strg]+[Alt] +[F8]	Markiert die Zeile, in der der Cursor steht.
Ausschneiden / Kopieren / Einfügen	
[Strg]+[C]	Auswahl in Zwischenspeicher kopieren
[Strg]+[Einfg]	Auswahl in Zwischenspeicher kopieren
[Umschalt]+[Entf]	Auswahl löschen und in Zwischenspeicher kopieren
[Strg]+[X]	Auswahl löschen und in Zwischenspeicher kopieren
[Strg]+[Y]	Markierte Zeile löschen und in Zwischenspeicher kopieren
[Strg]+[V]	Einfügen aus dem Zwischenspeicher
[Umschalt]+[Einfg]	Einfügen aus dem Zwischenspeicher
[Strg]+[Alt]+[K]	Alle Zeilen von der vorhergehenden Leerzeile bis zur nächsten Leerzeile löschen und in den Zwischenspeicher kopieren.
Suchen / Ersetzen	
[Alt]+[F3]	Suchen
[Strg]+[F]	Suchen
[F3]	Weitersuchen.
[Umschalt]+[F3]	Weitersuchen nach oben.
[Strg]+[F3]	Nächstes Wort wie unter dem Cursor suchen.
[Strg]+[Umschalt]+[F3]	Vorheriges Wort wie unter dem Cursor suchen.
[Strg]+[R]	Suchen und ersetzen.
Ändern	
[Einfg]	Zwischen „Einfügen" und „Überschreiben" hin- und herschalten.
[Strg]+[Z]	Rückgängig: Letzte Änderung
[Alt]+[Rücktaste]	Rückgängig: Letzte Änderung
[Tab] (mit markierten Zeilen)	Rückt die ausgewählten Zeilen ein.
[Umschalt]+[Tab]	Rückt die ausgewählten Zeilen wieder aus.
[Strg]+[Backspace]	Löscht bis zum Beginn des Wortes.
[Strg]+[Entf]	Löscht bis zum Ende des Wortes.
[Strg]+[U]	Auswahl in Kleinbuchstaben ändern.
[Strg]+[Umschalt]+ U	Auswahl in Großbuchstaben ändern.
[Strg]+[Umschalt]+ N	Neue Zeile oberhalb einfügen.

[Strg]+[Alt]+[R]	Wiederholt den nächsten Befehl mehrere Male (öffnet den Dialog).
[Strg]+[Umschalt]+[R]	Startet die Makro-Aufzeichnung.
Fenster scrollen	
[Strg]+[↑]	Fenster nach oben scrollen
[Strg]+[↓]	Fenster nach unten scrollen
[Strg]+[Bild↑]	Fenster nach links scrollen
[Strg]+[Bild↓]	Fenster nach rechts scrollen

Funktion Rückgängig

Mit [Alt]+[Rücktaste], [Strg]+[Z] oder *Bearbeiten* → *Rückgängig* können Sie die letzte Aktion im Editierfenster rückgängig machen.

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

Tabulatoren

Benutzen Sie Tabulatoren und Einzüge um Ihr Programm visuell zu strukturieren. Die Tab-Inkremente können in *Einstellungen* → *Editor* individuell definiert werden.

Zeilennummer

Innerhalb des Programms können Sie sich an den Zeilennummern orientieren. Die Syntaxprüfung zum Beispiel stellt nicht nur den Cursor in die Zeile mit einem Fehler, sondern nennt auch die Zeilennummern, die Fehler enthalten im Kommunikationsfenster.

Die aktuelle Zeilennummer finden Sie in der Statuszeile, zum Beispiel 13:1. Der Cursor steht dann in der Zeile 13 vor dem ersten Zeichen.

Makro Aufzeichnen und Abrufen

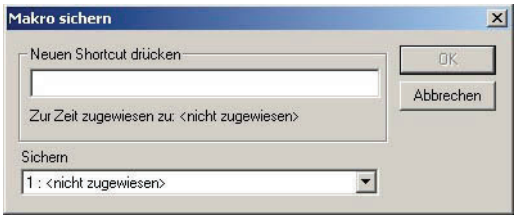
Häufig benutzte Befehle oder Befehlsketten können als Makro aufgezeichnet werden. Diesen Makros können Sie Funktionstasten zuordnen und dann die Befehle beliebig wiederholen. Bis zu 10 verschiedene Makros können definiert werden. Sie werden gesichert und wieder geladen, sobald APOSS-IDE das nächste Mal gestartet wird.

Drücken Sie [Strg]+[Umschalt]+[R] und das kleine Makro-Aufzeichnungsfenster zeigt den Start der Aufzeichnung. Dann geben Sie ein, was Sie aufzeichnen wollen: Das können normale Tastatureingaben, Shortcuts und Menü-Befehle sein.

Beenden Sie die Aufzeichnung durch Klicken auf das schwarze Schließen-Symbol im „Makro-Aufzeichnen”-Dialog.

Der „Makro sichern” Dialog wird geöffnet:

Drücken Sie die Tasten für das Shortcut, das Sie für das Makro benutzen wollen (z.B. [Strg]+[Umschalt]+[M]). Jedes Shortcut kann bis zu 2 Zeichen lang sein. Dann wählen Sie eine der beiden „Sichern als” Alternativen.



MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

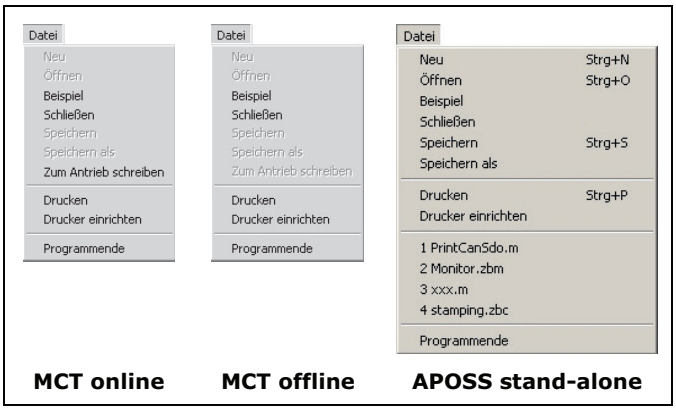
ACHTUNG!:
Sie können zum Festlegen des Makros fast alle Tasten oder Funktionstasten verwenden. Benutzen Sie aber nicht Funktionstasten oder Shortcuts, die bereits eine Funktion haben, denn dann wäre die ursprüngliche Funktion verloren und in einigen Fällen würde dies zu unvorhersehbaren Ergebnissen führen. Wenn Sie zum Beispiel die [↑]-Taste belegen, können Sie diese nie mehr zum Bewegen des Cursors einsetzen!
Die [Alt]-Taste dagegen (z.B. [Alt]+[A], [Alt]+[X] usw.) eignet sich sehr gut für Shortcuts, da die meisten Kombinationen nicht mit vordefinierten Shortcuts in Konflikt geraten.

Mit *Bearbeiten → Alle Makros löschen* können alle derzeit definierten Makros gelöscht werden.

□ Menü Datei

Das Menü *Datei* enthält alle notwendigen Befehle zum → *Öffnen, Schließen, Speichern, Speichern als, Drucken* und *Drucker einrichten* eines Programms; sie werden wie üblich benutzt.

Je nach Modus – MCT online, MCT offline oder APOSS stand-alone – sind verschiedene Funktionen enabled:



Datei → Neu

Für neue Dateien benutzen Sie MCT 10 oder in APOSS stand-alone *Datei → Neu*.

Datei → Öffnen

Wählen Sie die Datei mit MCT 10 aus. Damit wird automatisch APOSS und die Datei geöffnet. In APOSS stand-alone 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 in APOSS stand-alone → *Speichern als*.

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.

Beispiel

APOSS enthält viele Programmbeispiele. Jedes dieser Programmbeispiele kann vom Anwender beliebig benutzt, verändert oder in andere Programme eingebunden werden. Mit *Datei → Beispiel* kann jedes dieser Programmbeispiele editiert werden. Sehen Sie auch die Übersicht auf Seite 195 oder die „Programmbeispiele" in der Onlinehilfe.

Programmende

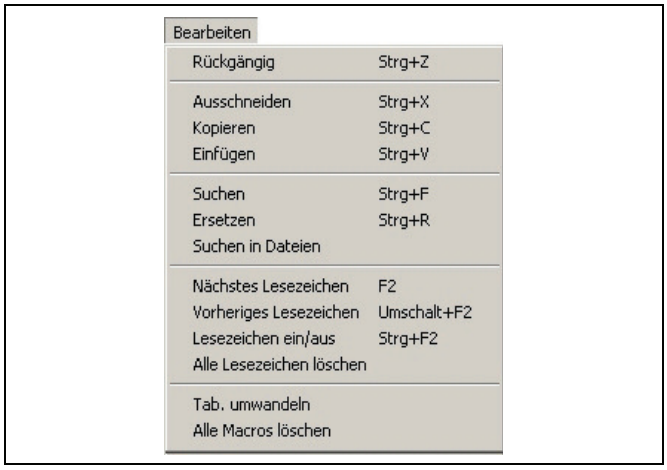
Das Programm APOSS kann durch Klicken auf → *Programmende* oder auf das ☒ Symbol beendet werden. Falls Sie eine neue oder geänderte Datei noch nicht gespeichert haben, haben Sie jetzt die Möglichkeit, dies zu tun.

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

ACHTUNG!:
Das Trennen der Verbindung von sehr viel älteren Steuerungen (zum Beispiel bei *Datei* → *Programmende*), kann dazu führen, dass die Steuerung die Programmausführung stoppt, wenn von der Steuerung PRINT Befehle an den PC geschickt werden. Dies passiert, wenn der interne Print-Puffer der Steuerung voll ist. Dies ist kein Problem für alle neueren Steuerungen. Diese können die Programmausführung fortsetzen, auch wenn der Print-Puffer voll ist; die Print-Meldungen werden einfach verworfen, sobald der Puffer voll ist.

□ Menü Bearbeiten

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



□ 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* statt auf *Suchen* 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. alle Zeichen von a bis c: [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-Umschaltzeichen	Abläufe wie \t, etc. werden durch ein äquivalentes einzelnes Zeichen ersetzt. \\ stellt den Backslash dar.

Beispiele:

10	„10” suchen
10+	Suchen nach „1” gefolgt von mindestens einer „0” (z.B. 10, 100, 1000, etc.).
10*	Suchen nach „1” gefolgt von keiner oder mehreren „0” (z.B. 1, 10, 100, 1000, etc.).
vel[xyz]	Suchen nach „velx”, „vely” oder „velz”.
vel[1-3]	Suchen nach „vel1”, „vel2” oder „vel3”.
(vel)|(acc)	Suchen nach „vel” oder „acc”.
vel[\t]*=	Suchen nach „vel”, gefolgt von einer beliebigen Anzahl von Leerzeichen oder Tabs, gefolgt von „=” (d.h. Suchen nach Anweisungen der Variablen „vel”).

□ Suchen in Dateien

Klicken Sie auf → *Suchen in Dateien* um bestimmte Textpassagen in Dateien zu finden. Alle gefundenen Vorkommen werden im Dialogfenster angezeigt. Doppelklick auf eine Fundstelle öffnet die Datei und positioniert den Cursor in der Zeile. Beachten Sie, dass *Suchen in Dateien* nur Vorkommen eines Textes in Dateien sucht, die auf der Festplatte gespeichert sind. Alle (noch) nicht gespeicherten Änderungen einer gerade bearbeiteten Datei können so nicht gefunden werden.

□ Lesezeichen

Mit Lesezeichen kann der Anwender bestimmte für ihn interessante Zeilen markieren und dann schnell zwischen diesen zu springen.

Wenn Lesezeichen im Editor gesetzt wurden, werden diese mit der Programmdatei gespeichert und mit dieser wieder geladen und gesetzt.

Nächstes Lesezeichen [F2]

Wenn Lesezeichen im Editor gesetzt wurden, dann scrollt → *Nächstes Lesezeichen* oder [F2] durch das Programm und positioniert den Cursor auf die nächste Zeile, die ein Lesezeichen enthält.

Vorheriges Lesezeichen

Wenn Lesezeichen im Editor gesetzt wurden, dann positioniert → *Vorheriges Lesezeichen* oder [Umschalt]+[F2] den Cursor auf die vorherige Zeile, die ein Lesezeichen enthält.

Lesezeichen ein/aus

Diese Funktion oder [Strg]+[F2] schaltet das Lesezeichen in der aktuellen Zeile ein- bzw. aus. Wenn in dieser Zeile kein Lesezeichen ist, wird eines gesetzt. Falls die Zeile bereits ein Lesezeichen enthält, wird es gelöscht.

Alle Lesezeichen Löschen

Klicken Sie auf *Bearbeiten* → *Alle Lesezeichen löschen* um alle existierenden Lesezeichen im Editor zu löschen.

□ Tab umwandeln

Klicken Sie auf → *Tab. Umwandeln* und alle Tab-Zeichen werden während des Editierens in Leerzeichen umgewandelt.

Dazu wird der Tab-Wert benutzt, der in *Einstellungen* → *Editor* gesetzt ist.

□ Alle Makros löschen

Diese Menü-Funktion löscht alle Makros im Editor und deren Shortcuts, die mit [Strg]+[Umschalt]+[R] erzeugt wurden. Siehe auch „Makro aufzeichnen und abrufen”.

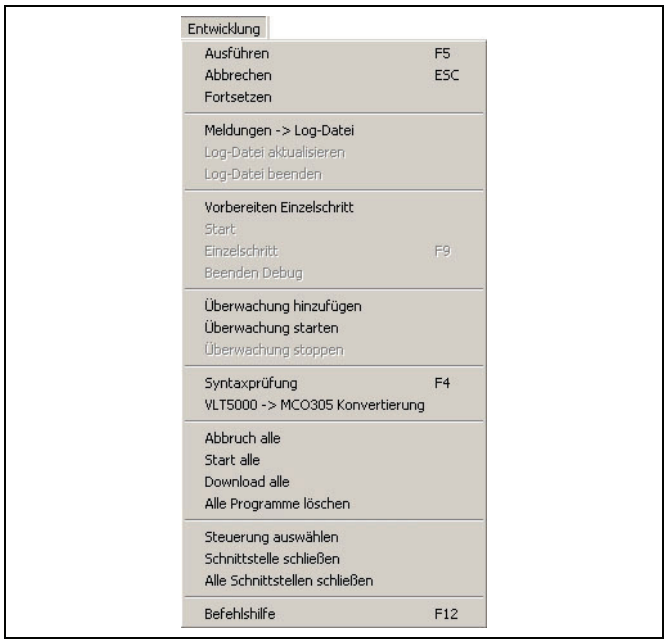
□ Menü Entwicklung

Dieses Menü bietet verschiedene Funktionen für die Phase der Entwicklung von Anwendungsprogrammen. Dies beinhaltet Funktionen zum Kompilieren, Ausführen, Abbrechen und für die Fehlersuche. Es enthält außerdem Funktionen, um Steuerungen zu verbinden und Schnittstellen zu schließen.

Bevor Sie beginnen, müssen Sie immer eine Steuerung bzw. einen FC 300 auswählen. Eine Steuerung kann entweder mit [Esc] mit der Default-Steuerung verbunden werden, oder mit *Entwicklung → Steuerung auswählen* mit einer bestimmten Steuerung, wenn mehrere Steuerungen vorhanden sind. Falls keine Steuerung verbunden ist wenn eine Funktion ausgeführt wird, werden die meisten Funktionen automatisch eine Verbindung zur Default-Steuerung herstellen.

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.

Informationen zu Debugging-Programmen finden Sie in *Programme Debuggen* am Ende dieses Kapitels.



□ Ausführen [F5]

Es wird das Programm gestartet, das geöffnet und im Editor dargestellt ist. Dies beinhaltet folgende Schritte:

1. Falls aktuell keine Steuerung offen und mit dem APOSS-Fenster verbunden ist, wird versucht, eine Verbindung mit der als Default definierten Steuerung herzustellen. Ist keine Steuerung angeschlossen, wird *Ausführen* abgebrochen.
2. Es wird geprüft, ob eine Steuerung bereits ein Programm ausführt. Falls dies der Fall ist, wird der Anwender gefragt, ob das existierende Programm gestoppt werden kann, da die Steuerung nur ein Programm zur gleichen Zeit ausführen kann. Wenn die Steuerung nicht im Leerlauf ist, wird *Ausführen* abgebrochen.
3. Wenn der Anwender während des Editierens Änderungen im Programm vorgenommen hat, wurden diese automatisch auf der PC-Festplatte gesichert. Falls dies ein „neues" Programm ist, kann man jetzt einen Dateinamen für das Programm eingeben. Es ist aber zu diesem Zeitpunkt nicht notwendig, denn ohne Dateinamen wird eine temporäre Datei benutzt.
4. Das editierte Programm wird dann kompiliert und eine für die eingesetzte Steuerung passende Version des Programms mit Maschinencode erzeugt. Beachten Sie, dass der „Maschinencode" nicht editiert werden kann. Falls das Kompilieren aus irgendeinem Grund fehlschlägt, wird *Ausführen* abgebrochen.
5. Dieser Maschinencode wird dann in den temporären Speicher der Steuerung heruntergeladen. Beachten Sie, dass dieser temporäre Speicher verloren geht, sobald die Steuerung ausgeschaltet wird. Um ein Programm dauerhaft in der Steuerung zu sichern, benutzen Sie *Steuerung → Programme.*
6. Sobald der Download beendet ist, wird das Programm in der Steuerung ausgeführt.

Jedes Mal wenn *Ausführen* benutzt wird, wird das vorher heruntergeladene Programm mit dem erneut heruntergeladenen überschrieben. Dies erlaubt es, sehr schnell und leicht Änderungen vorzunehmen und wieder zu testen.

Programme in mehreren FC 300 ausführen

Sie können ganz einfach verschiedene Programme in verschiedenen Steuerungen zur gleichen Zeit ausführen: Dazu öffnen Sie jedes Programm in einem anderen APOSS Fenster und wählen in jedem Fenster mit *Entwicklung → Steuerung auswählen* den jeweiligen FC 300 für dieses Programm aus. Dann starten Sie mit *Entwicklung → Ausführen* in jedem Fenster den Download und das Programm.

Beachten Sie, dass Sie zuerst eine Steuerung *Auswählen* müssen, bevor Sie *Ausführen* starten, weil *Ausführen* immer die Default-Steuerung öffnet, wenn keine Steuerung ausgewählt ist.

Jedes APOSS Fenster kann zur gleichen Zeit mit nur einer Steuerung verbunden sein und jedes APOSS Programm kann zur gleichen Zeit nur in einem APOSS Fenster editiert werden. Falls Sie also das gleiche Programm in mehreren Steuerungen ausführen wollen, können Sie eine der folgenden Methoden nutzen:

1. Verbinden Sie mit *Entwicklung → Steuerung auswählen* die erste Steuerung. Dann downloaden Sie mit *Entwicklung → Ausführen* das Programm und starten die Programmausführung in der Steuerung. Sobald es ausgeführt wird, verbinden Sie mit *Entwicklung → Steuerung auswählen* die zweite Steuerung. Das trennt zwar die Verbindung mit der ersten Steuerung, lässt aber das Programm weiter laufen. Downloaden Sie mit *Entwicklung → Ausführen* wieder das Programm und starten Sie die Programmausführung. Wiederholen Sie diese Vorgehensweise für jede der Steuerungen, in denen das Programm laufen soll.
2. Falls das Programm in mehreren Steuerungen laufen soll, die alle im gleichen Netzwerk sind (zum Beispiel in einem Feldbus), können Sie *Entwicklung → Download alle* nutzen. Dann können die oben genannten Schritte mit einem einzigen Befehl ausgeführt werden.

Wenn Sie das Programm in mehrere Steuerungen laden wollen, verbinden Sie das Programm mit der jeweiligen Steuerung und klicken auf → *Ausführen.*

□ Abbrechen [Esc] und Fortsetzen

Klicken Sie auf *Entwicklung → Abbrechen* oder drücken Sie [Esc] um alle Programme, die in der Steuerung ausgeführt werden, abzubrechen.

ACHTUNG!:
Falls eine Programmausführung abgebrochen wird, während sich der Antrieb dreht, wird der Antrieb mit der maximal zulässigen Verzögerung abgebremst.

Falls das Programm in mehreren Steuerungen gleichzeitig läuft, können Sie mit *Entwicklung → Abbruch alle* laufenden Programme abbrechen.

Klicken Sie auf *Entwicklung → Fortsetzen*, um das eben abgebrochene Programm fortzusetzen. Dabei werden auch die unterbrochenen Positioniervorgänge 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 aller Meldungen, die im Kommunikations-Fenster dargestellt werden, in eine Datei; mit *Log-Datei beenden* wird sie beendet.

Die Meldungen werden gecached und nicht immer sofort in die Datei geschrieben. Wenn also die Datei bearbeitet oder kopiert wird, während noch die Protokollierung läuft, werden die allerletzten Meldungen nicht immer enthalten sein. Sie können dies vermeiden, wenn Sie unmittelbar bevor dem Kopieren oder Bearbeiten die → *Log-Datei aktualisieren*. Dann werden auch alle zwischengespeicherten Meldungen bis zu diesem Zeitpunkt in die Datei geschrieben.

Beachten Sie, dass *Log-Datei beenden* und *Log-Datei aktualisieren* erst aktiviert werden, wenn die Protokollierung gestartet wurde.

☐ Debug Befehle

Die folgenden Befehle wurden entwickelt, um den Anwender bei der Fehlersuche, also beim Debuggen neu entwickelter Programme zu unterstützen. Detaillierte Information über Debuggen und den Gebrauch der Befehle finden Sie in „Programme debuggen" am Ende dieses Kapitels.

Vorbereiten Einzelschritt

Dieser Befehle bereitet sowohl die APOSS-IDE als auch die Steuerung für das Debuggen vor. Dies beinhaltet das Kompilieren des Programms im „Debug"-Modus, das Einfügen von Breakpoints und das Downloaden des Programms.

Start

Dieser Befehl startet die Ausführung des Programms in der aktuellen Programmzeile und führt die Ausführung bis zum nächsten vom Anwender gesetzten Breakpoint fort.

Einzelschritt [F9]

Dieser Befehl führt eine einzelne Programmzeile aus und stoppt vor der nächsten Zeile. Die nächste Zeile wird nicht ausgeführt.

Beenden Debug

Mit diesem Befehl wird der Debug-Modus beendet, und zwar sowohl in der APOSS-IDE als auch in der Steuerung

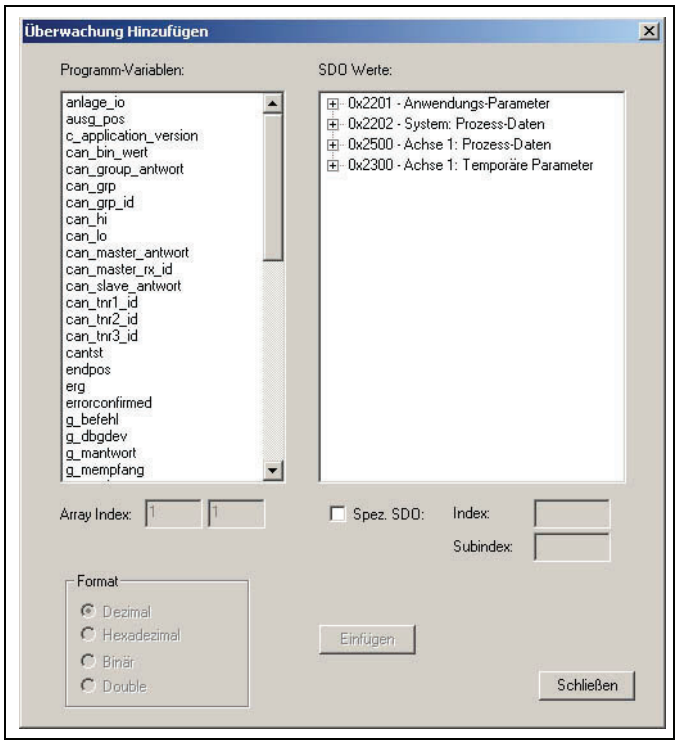
☐ Überwachung hinzufügen / starten / stoppen

Diese Funktion aktiviert die Online-Überwachung der Variablen, Arrays, System- und Achsprozessdaten und der Achsparameter.

Alle gerade überwachten Werte werden im Überwachungsfenster angezeigt. Neue Werte können Sie zu dieser Liste mit *Entwicklung → Überwachung* hinzufügen. Zum Löschen vorhandener Werte klicken Sie auf den Wert und drücken die [Entf]-Taste.

Werte können durch Rechtsklicken auf einen Listeneintrag zur Steuerung geschrieben werden (wenn eine Steuerung angeschlossen ist). Die Überwachung muss dafür nicht aktiv sein.

Auf der linken Seite werden alle Programmvariablen und Arrays dargestellt, auf der rechten Seite alle Systemwerte (in einem Verzeichnisbaum). Um einen Wert zum Überwachungsfenster hinzuzufügen, markieren Sie den Wert, wählen dazu das Format, das für die Darstellung benutzt werden soll und klicken auf → *Einfügen*. Sie können Werte auch einfach durch Doppelklicken hinzufügen. Sie können mehrere Werte einfügen, bevor Sie das Dialogfeld schließen.



Wenn ein Array-Wert zum Überwachungsfenster hinzugefügt wird, dann müssen die Array-Indices spezifiziert werden. Diese Felder sind deaktiviert, wenn der ausgewählte Wert nicht zu einem Array gehört. Beachten Sie, es können nur die ersten 250 Elemente eines Arrays überwacht werden.

Aktivieren und stoppen Sie das Monitoring mit → *Überwachung starten, Überwachung stoppen* oder klicken Sie auf .

Wenn die Überwachung aktiviert ist, wird das Überwachungsfenster ständig aktualisiert. Dies belegt sowohl Ressourcen der Steuerung als auch des Netzwerks. Daher sollte in die Anzahl der überwachten Werte auf ein vernünftiges Maß begrenzt werden (im Allgemeinen nicht mehr als 10 bis 15 Werte, abhängig von der Netzwerk-Geschwindigkeit). Wenn es notwendig ist, mehr Werte zu überwachen, sollte das Oszilloskop benutzt werden.

☐ Syntaxprüfung [F4]

Entwicklung → *Syntaxprüfung* erzeugt eine „Test-Kompilierung" des Programms, das gerade editiert wird. Das kann während einer neuen Programmierung oder Änderung eines existierenden Programms sehr hilfreich sein. Es ist ein schneller Weg, Syntax-Fehler im Programm zu finden ohne das Programm in die Steuerung downloaden und ausführen zu müssen. Wenn ein Syntaxfehler gefunden wird, wird die Zeilennummer und eine Fehlerbeschreibung im Kommunikationsfenster ausgegeben und der Cursor automatisch an die Position mit dem Syntaxfehler gestellt.

Die Syntaxprüfung erzeugt eine zusätzliche Debug-Datei zur Prüfung der Syntax. Diese Datei erhält den gleichen Namen wie das Programm, jedoch mit der Extension „.ad$".

☐ In Datei kompilieren

Der Befehl *Ausführen[F5]* kompiliert ein Programm in eine „maschinenlesbare" binäre Version, die in die Steuerung geladen und ausgeführt wird. Der Befehl *In Datei kompilieren* ist ähnlich, außer dass die kompilierte binäre Version nicht heruntergeladen und ausgeführt wird. Stattdessen wird die binäre Version als „.bin"-Datei auf der PC-Festplatte gespeichert. Binäre „.bin"-Dateien werden mit *Steuerung* → *Programme* verwaltet.

Beim → *In Datei kompilieren* bietet ein Dialogfeld die Möglichkeit, genau die Steuerung zu bestimmen, für die das Programm kompiliert werden soll. Beachten Sie, dass die kompilierten binären Versionen hardwareabhängig sind und für die Hardware kompiliert werden müssen, in welcher sie ausgeführt werden. Danach kann in einem *Sichern als* Dialog der Dateiname zum Speichern der Datei ausgewählt werden. Als Default wird der Programmname mit der Erweiterung „.bin" benutzt. Erst danach wird das Programm kompiliert und auf der Festplatte gesichert.

ACHTUNG!:
Diese Funktion ist nur verfügbar, wenn *Binär-Datei Unterstützung* 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".

```
DIM receive [4]
// LINKGPAR muss wie folgt geändert werden ...
LINKGPAR 133 710 "DATA WORD 1" 0 255 0

// LINKGPAR muss wie folgt geändert werden ...
LINKGPAR 134 711 "DATA WORD 2" 0 255 0
```

☐ Abbruch alle

Falls das Programm in mehreren Steuerungen läuft, die alle im gleichen Netzwerk hängen, klicken Sie auf *Entwicklung* → *Abbruch alle*, um alle laufenden Programme abzubrechen.

Das heißt, diese Funktion bricht nur die Programmausführung ab, die in Steuerungen laufen, die das gleiche Anschluss-Interface haben wie die Steuerung, die mit diesem APOSS Fenster verbunden ist (z.B. mehrere Steuerungen an einem Feldbus). Steuerungen, die andere Anschluss-Interfaces benutzen, werden nicht abgebrochen.

Beachten Sie, dass die Programme auch in den Steuerungen abgebrochen werden, die aktuell nicht mit einem APOSS-Fenster verbunden sind, so lange die Steuerung das gleiche Anschluss-Interface im ID-Scan-bereich nutzt. Wären zum Beispiel die Steuerungen 1 und 2 beide in einem Feldbus-Netzwerk, aber APOSS ist gerade nur mit Steuerung 1 verbunden, dann würde *Abbruch alle* auch das Programm in Steuerung 2 abbrechen.

ACHTUNG!:
Falls eine Programmausführung abgebrochen wird, während sich der Antrieb dreht, wird der Antrieb mit der maximal zulässigen Verzögerung abgebremst.

□ Start alle

Die Funktion → *Start alle* sendet einen 'Ausführen' Befehl an alle angeschlossenen Steuerungen, die das gleiche Anschluss-Interface nutzen, wie die Steuerung, die mit dem APOSS-Fenster verbunden ist. Wenn Sie Programme testen, die in mehreren Steuerungen in einem Netzwerk laufen, ist dies eine schnelle Möglich-keit, alle Steuerungen zur gleichen Zeit zu starten.

□ Download alle

Entwicklung → Download alle öffnet den *APOSS Download-Modus* Dialog, mit dem die gerade editierte .m-Datei in mehrere Steuerungen heruntergeladen werden kann.

Mehr Details und welche Optionen möglich sind, lesen Sie bitte im Abschnitt *Download → Programme.*

□ Alle Programme löschen

Entwicklung → Alle Programme löschen löscht alle Programme in allen Steuerungen, die die gleiche Schnitt-stelle benutzten, wie die Steuerung, die mit diesem APOSS-Fenster verbunden ist (d.h. mehrere Steuerun-gen an einem Feldbus). Verbundene Steuerungen, die andere Schnittstellen benutzen, sind nicht betroffen.

Beachten Sie, dass die Programme auch in den Steuerungen gelöscht werden, die gerade nicht mit einem APOSS-Fenster verbunden sind. Solange wie die Steuerung die gleiche Schnittstelle benutzt und innerhalb des ID-Scanbereichs ist, werden die Programme gelöscht. Wenn zum Beispiel die beiden Steuerungen 1 und 2 in einem Feldbus-Netzwerk sind, aber APOSS gerade nur mit der Steuerung 1 verbunden ist, wird → *Alle Programme löschen* die Programme in beiden Steuerung 1 und 2 löschen.

Dieser Befehl wird in Verbindung mit *Entwicklung → Download alle* benutzt.

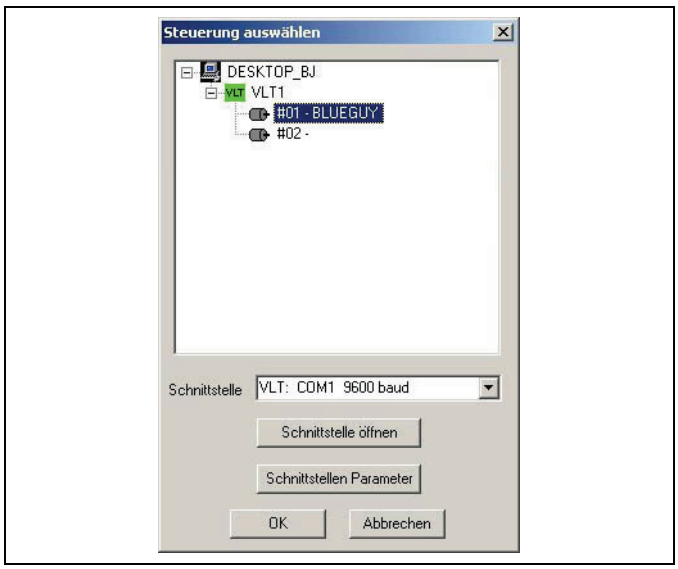
□ Steuerung auswählen

Wenn Sie mehr als einen FC300 konfiguriert haben, wählen Sie mit → *Steuerung auswählen* den FC 300 aus, in den Sie Programme laden und den Sie star-ten 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*. Dann können Sie das gewünschte Interface auswählen.

Beachten Sie, wenn Sie hier eine Schnittstelle aus-wählen, ändert das nicht die Default-Schnittstelle, die mit *Einstellungen → Schnittstelle* festgelegt wurde.



MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

Wenn eine Steuerung schon mit einem APOSS-Fenster verbunden ist, kann mit *Entwicklung* → *Steuerung auswählen* das APOSS-Fenster mit einer anderen Steuerung verbunden werden. Benutzen Sie dazu einfach den Dialog *Steuerung auswählen*. Die vorhandene Verbindung wird dann geschlossen und eine Verbindung zur neuen Steuerung geöffnet.

Weil jede spezifische Steuerung zur gleichen Zeit mit nur einem APOSS-Fenster verbunden sein kann, führt das Auswählen einer Steuerung, die schon mit einem anderen APOSS-Fenster verbunden ist, dazu, dass die Steuerung vom anderen APOSS-Fenster zu diesem APOSS-Fenster wechselt.

□ Schnittstelle schließen / Alle Schnittstellen schließen

Wenn eine Steuerung mit einem APOSS-Fenster verbunden ist, können Sie mit dieser Funktion die Verbindungen zu allen Steuerungen schließen, die das gleiche Anschluss-Interface nutzen (z.B. alle Steuerungen an einem CAN-Bus). Eine Meldung zeigt an, welche Verbindungen beendet werden.

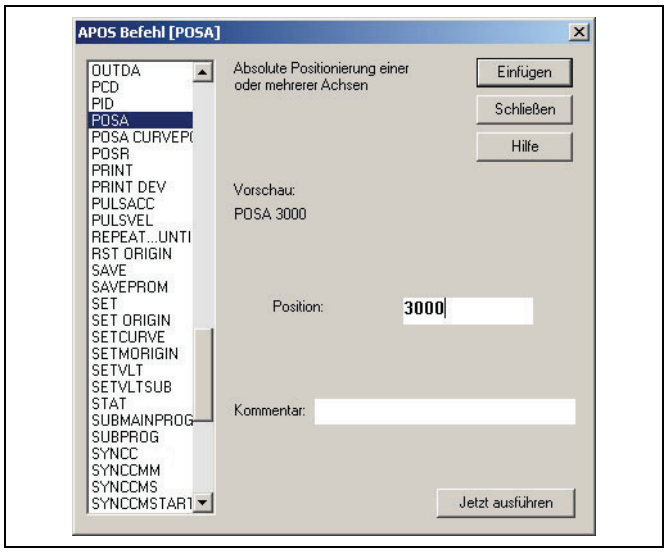
Mit *Alle Schnittstellen schließen* werden alle Steuerungen an allen Anschluss-Interfaces geschlossen.

□ Befehlshilfe [F12]

Die *Befehlshilfe* zeigt alle Befehle mit der entsprechenden Syntax, die Sie ganz einfach in Ihr Programm → *Einfügen* oder direkt ausführen können. Besonders für selten benutzte Befehle kann dies ganz hilfreich sein. Sie erhalten ausführliche Informationen zu einem markierten Befehl, wenn Sie auf → *Hilfe* klicken oder [F1] drücken.

Zum Beispiel POSA: Geben Sie die Position in das Feld für die Achse ein. Die Vorschau zeigt Ihnen die genaue Syntax des Befehls.

Im Betriebsmodus Offline ist das direkte Ausführen eines Befehls mit der Funktion → *jetzt ausführen* nicht möglich.



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

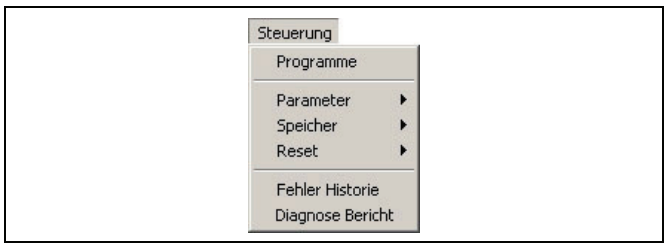
Stellen Sie den Cursor im Editierfenster an die Stelle, wo Sie ein oder mehrere Befehle einfügen wollen, wählen Sie *Entwicklung* → *Befehlshilfe* und hier den Befehl aus, zum Beispiel POSA, ergänzen den Wert und ggf. einen Kommentar und klicken Sie auf → *Einfügen*.

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

ACHTUNG!:
Abhängig vom Befehl können freigegebene Antriebe anlaufen, stoppen etc.

□ Menü Steuerung

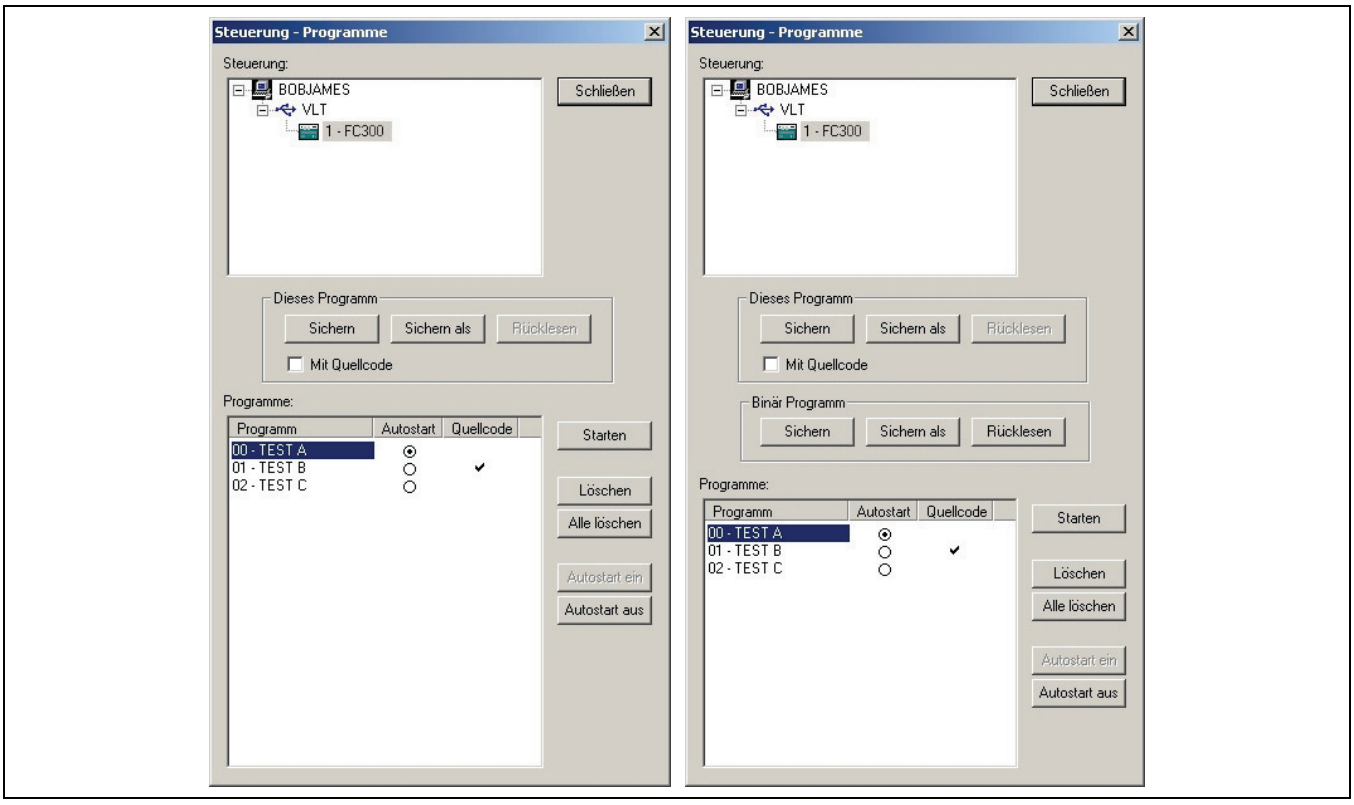
Dieses Menü bietet Funktionen um die Konfiguration der Steuerung anzusehen und einzustellen sowie den Speicher in der Steuerung zu organisieren. Dazu gehört: Verwalten der gespeicherten Programme (und markieren der Programme für den *Autostart)*, prüfen und setzen der globalen und Achsparameter sowie sichern und zurückzusetzen des EPROM-Speichers der Steuerung.



□ Programme

Klicken Sie auf *Steuerung* → *Programme* und das Dialogfeld zeigt alle angeschlossenen Steuerungen. Sie können natürlich auch einen anderen FC 300 markieren und bearbeiten.

ACHTUNG!:
Wenn *Binärdatei-Unterstützung* in *Einstellungen* → *Optionen* aktiviert ist, sieht das Dialogfeld wie rechts gezeigt aus.



Unter *Programme:* steht die Liste aller Programme, die derzeit in der Steuerung gesichert sind. Jedes Programm erhält eine Programmnummer sowie einen Programmnamen. Abhängig vom verfügbaren Speicher und der Größe der gesicherten Programme können bis zu 91 Programme zur gleichen Zeit gesichert werden. Das Symbol ⊙ in der Autostart-Spalte zeigt, dass das Programm für den *Autostart* gekennzeichnet wurde. Das Häkchen ✓ zeigt, dass das Programm *mit Quellcode* in der Steuerung gesichert ist.

Dieses Programm sichern

Immer wenn Sie ein Programm mit *Entwicklung* → *Ausführen [F5]* wird es kompiliert und in einen temporären Speicherbereich der Steuerung heruntergeladen. Dieser wird mit jedem folgenden *Ausführen* überschrieben und geht verloren, sobald die Steuerung ausgeschaltet wird. Mit *Steuerung* → *Programme* können Sie jedoch das aktuelle Programm aus dem APOSS-Fenster permanent → *Sichern*. Das Programm wird kompiliert und in einen permanenten Speicherbereich der Steuerung heruntergeladen. Das neue Programm wird an das Ende der Liste der vorhandenen Programme gehängt.

ACHTUNG!:
Beachten Sie, dass *Dieses Programm* → *Sichern* nicht automatisch die originale Quelldatei des Programms sichert. Sie sollten daher immer mit dem normalen Befehl die *Datei* auch auf der Festplatte des PCs → *Speichern*.

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

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

Sichern mit Quellcode

Wenn die Checkbox aktiviert ist, wird zusätzlich zur kompilierten und direkt ausführbaren Programmdatei der Quellcode im FC 300 gesichert. Sie können diesen dann später wieder aus der Steuerung auslesen (upload) und auf dem PC erneut bearbeiten.

Wenn ein Programm-Quellcode heruntergeladen wird, werden eingebundene Dateien (Include-Dateien), die der Quellcode benutzt, auch mit dem Quellcode heruntergeladen. Dies ermöglicht es, statt Programmteile das komplette Programm in der Steuerung zu speichern.

Alle mit Quellcode gesicherten Programme erhalten in der Spalte Quellcode ein ✓ Häkchen.

ACHTUNG!:
Sollte in der Steuerung nicht genügend Speicher zur Verfügung stehen, um den Quellcode zu speichern, erfolgt eine Meldung und Sie müssen erst andere Programme löschen, bevor das neue Programm gespeichert werden kann.

Dieses Programm Rücklesen

Für alle mit ✓ gekennzeichneten Programme ist der dazugehörende Quellcode in der Steuerung gespeichert. Dieser Quellcode kann zur weiteren Verwendung aus der Steuerung zurück auf den PC gelesen werden.

Wählen Sie das gewünschte Programm aus und klicken Sie auf *Dieses Programm* → *Rücklesen*. Es wird ein Default-Programmname generiert, der Datum und Uhrzeit enthält, zu der der Original-Quellcode in der Steuerung gespeichert wurde. Damit werden Probleme vermieden, die durch das Überschreiben von vorhandenen Dateien auf dem PC entstehen könnten.

Binär Programm Sichern oder Rücklesen

Die Funktion wird nur dargestellt, wenn *Binär Datei Unterstützung* im Dialogfeld Einstellungen → *Optionen* ausgewählt ist.

Wählen Sie das gewünschte Programm aus und klicken Sie auf *Sichern* oder *Sichern als* und wählen Sie im folgenden Dialogfeld die kompilierte .bin Datei aus. Geben Sie einen Programmnamen und eine Nummer ein und downloaden Sie dann die binäre Datei in den FC 300.

Klicken Sie auf *Rücklesen* und das gerade ausgewählte binäre Programm wird aus dem FC 300 ausgelesen.

ACHTUNG!:
Steuerungen mit Versionen älter als MCO 5.00 unterstützen das Rücklesen der binären Dateien nicht; in diesem Fall ist die *Rücklesen*-Schaltfläche nicht anwählbar.

Programm Starten

Sie können im Programm-Dialogfenster ein beliebiges gespeichertes Programm auswählen und direkt → *Starten.*

Autostart

Um den Einsatz der Steuerung als Standalone oder schlüsselfertige Anwendung zu unterstützen (d.h. ohne Eingaben des Anwenders), kann die Steuerung so konfiguriert werden, dass ein APOSS-Programm automatisch startet, wenn die Steuerung eingeschaltet wird. Typischerweise darf die Steuerung bei diesen Anwendungen auch nie gestoppt werden. Dann kann die Steuerung so konfiguriert werden, dass ein „Restart"-APOSS-Programm startet, wenn das ausgeführte Programm abbricht.

Der erste Teil der Autostart-Prozedur ist das *Autostart-Programm* (sofern eines definiert ist). Es wird normalerweise immer automatisch gestartet wenn die Steuerung eingeschaltet wird. Ist kein Autostart-Programm definiert, dann wird beim Einschalten kein Programm gestartet.

Sobald die Steuerung eingeschaltet wird, laufen verschiedene Selbsttest-Routinen ab. Wenn einer dieser Tests fehlschlägt, wird das Autostart-Programm nicht gestartet.

Der zweite Teil der Autostart-Prozedur ist das *Restart-Programm*. Es wird mit einigen Ausnahmen automatisch gestartet, wenn das derzeit ausgeführte Programm abbricht; dies beinhaltet insbesondere auch den Abbruch des Autostart-Programms.

ACHTUNG!:
Wenn das *Autostart-Programm*, das *Restart-Programm* oder irgend ein anderes Programm durch den Anwender abgebrochen wird, dann wird der Autostart-Mechanismus deaktiviert. Das Autostart-Programm oder ein Restart-Programm wird nicht automatisch wieder gestartet bis die Steuerung aus- und wieder eingeschaltet wird.

ACHTUNG!:
Ein Programm, das als Autostart-Programm gekennzeichnet ist, kann immer auch manuell durch den Anwender gestartet werden. In diesem Fall wird das Programm jedoch als normales Programm ausgeführt und nicht als ein Autostart-Programm, und es wird auch kein Restart-Programm gestartet, wenn das Programm abbricht

Autostart Programm

Die Steuerung behandelt das Autostart-Programm wie folgt:

1. Wenn ein Programm explizit als „Autostart-Programm" gekennzeichnet wurde (siehe oben), dann wird dieses Programm gestartet, sobald die Steuerung eingeschaltet wird.
2. Wenn 33-5* *I_FUNCTION_n_13 oder 14* auf einen digitalen Eingang gesetzt ist, bestimmt 33-5* *I_FUNCTION_n_15* die Programmnummer des Autostart-Programms. Die Steuerung wartet bis der Eingang aktiviert wird und startet dann das Autostart-Programm.
   Beachten Sie, dass der Eingang schon aktiviert sein kann, wenn die Steuerung eingeschaltet wird. Diese Parameter sind dafür vorgesehen, um das Autostart-Programm von externen Quellen (z.B. eine SPS) auswählen zu können.
3. Oder es gibt kein Autostart-Programm.

Jedes gespeicherte Programm kann durch Auswahl und klicken auf → *Autostart ein* als Autostart-Programm bestimmt werden. Es wird dann in der Spalte mit einem ⊙ gekennzeichnet. Diese Zuweisung kann einfach durch erneute Auswahl und klicken auf → *Autostart aus* entfernt werden. Zur gleichen Zeit kann nur ein Autostart-Programm bestimmt sein. Wird also ein zweites Programm als Autostart-Programm ausgewählt, wird die Markierung für das vorhergehende entfernt.

Eine der Hauptaufgabe von Autostart-Programmen ist in Anwendungen, die verschiedene Programme in verschiedenen Situationen erfordern, auszuwählen, welches von mehreren Programmen ausgeführt werden soll. Das Autostart-Programm identifiziert (gewöhnlich entweder durch Konfigurationsparameter oder durch Setzen von digitalen Eingängen) welches Programm ausgeführt werden soll. Es bestimmt dann dieses Programm als das Restart-Programm und bricht ab. Danach startet die Steuerung automatisch das ausgewählte Programm.

ACHTUNG!:
Die Programmstart Prozedur (33-5* *I_FUNCTION_n_13 oder 14*) funktioniert in einigen Steuerungen vor MCO 5.00 nicht korrekt. Wenn dies die Anwendung tangiert, kontaktieren Sie bitte Ihren Lieferanten wegen eines Updates der Firmware.

Restart Programm

Wenn die Steuerung beim Einschalten ein Autostart-Programm ausführt, wird die Autostart-Prozedur aktiviert. Ist diese aktiv, startet die Steuerung automatisch das gewünschte Restart-Programm, wann immer das gerade ausgeführte Programm abbricht. Dies wird solange fortgeführt, bis die Autostart-Prozedur deaktiviert wird.

Die Autostart-Prozedur wird deaktiviert, wenn eines der folgenden Ereignisse eintritt:

1. Der Anwender hat einen „Abbruch"-Befehl ausgeführt (d.h. ein [Esc]).
2. Ein Programm wurde mit einem Fehler beendet (einem anderen als einen der unten aufgeführten).

Wenn die Autostart-Prozedur deaktiviert wurde kann sie nicht wieder durch Aus- und Wiedereinschalten der Steuerung reaktiviert werden.

Die Autostart-Prozedur wird nicht deaktiviert, wenn einer der folgenden Fehler auftritt:

1. Schleppfehler (Fehler-Nr. 108)
2. Endschalter erreicht (Fehler-Nr. 125)
3. Software Endschalter überschritten (Fehler-Nr. 111)

Die Steuerung bestimmt, welches Programm das Restart-Programm ist wie folgt:

1. Wenn 33-80 *Aktivierte Programmnummer* gesetzt ist (d.h. nicht -1), ist dies die Programmnummer des Restart-Programms.
2. Oder wenn 33-5* *I_FUNCTION_n_13 oder 14* auf einen digitalen Eingang gesetzt ist: dann wartet die Steuerung bis dieser Eingang aktiviert wird. Sobald aktiviert, bestimmt 33-5* *I_FUNCTION_n_15* die Programmnummer des Restart-Programms.
   Beachten Sie, dass der Eingang nicht schon aktiviert sein darf, wenn das gerade laufende Programm abbricht; die Aktivierung wird ausschließlich durch eine steigende Flanke getriggert.
3. Oder wenn das Autostart-Programm das einzige Programm ist, das ausgeführt werden soll; dann ist das Autostart-Programm auch das Restart-Programm (d.h. das Autostart-Programm wird wiederholend ausgeführt).
4. Oder es gibt kein Restart-Programm und die Steuerung startet überhaupt kein Programm.

Der Parameter 33-80 *Aktivierte Programmnummer* ist für die „Verkettung" von Programmen vorgesehen. Dazu setzt das gerade ausgeführte Programm einfach nur den Parameter, um festzulegen welches Programm als nächstes ausgeführt werden soll.
Beachten Sie, dass falls die *Aktivierte Programmnummer* nicht zurückgesetzt wird, das laufende Programm wiederholt ausgeführt wird.

Die Parameter 33-5* *I_FUNCTION Programmstart* und *Programmwahl* sind dafür vorgesehen, das nächste Programm das ausgeführt werden soll von externen Quellen (wie z.B. eine SPS) auswählen zu können.

ACHTUNG!:
Das ursprünglich bestimmte Autostart-Programm wird nur dann wieder gestartet, wenn 33-80 *Aktivierte Programmnummer* und 33-5* *I_FUNCTION Programmstart* nicht benutzt wurden.
Wurden entweder *Aktivierte Programmnummer* oder *Programmstart* benutzt, wird das Autostart-Programm nur einmal ausgeführt, wenn die Steuerung eingeschaltet wird. Diese „einmalige" Ausführung kann für ausführende Funktionen wie HOME nützlich sein
Beachten Sie, dass der Parameter 33-80 *Aktivierte Programmnummer* benutzt werden kann, um explizit das Autostart-Programm wieder zu starten.

ACHTUNG!:
Die I_PRGSTART Prozedur funktioniert in einigen Steuerungen vor MCO 5.00 nicht korrekt. Wenn dies die Anwendung tangiert, kontaktieren Sie bitte Ihren Lieferanten wegen eines Updates der Firmware.

Programme löschen

Wählen Sie ein Programm aus und klicken sie auf *Löschen***,** wenn Sie ein bestimmtes Programm in der Steuerung löschen wollen. Damit wird das Programm in der Steuerung vollständig gelöscht; Sie sollten also sicher sein, dass Sie es vorher auf dem PC gespeichert haben.

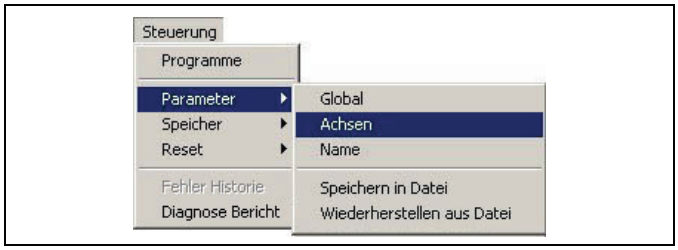
Klicken Sie auf *Alle Löschen*, wenn Sie alle Programme in der Steuerung löschen wollen.

ACHTUNG!:
Vergewissern Sie sich zuvor, dass Sie die Programme noch im PC zur Sicherheit oder für das Archiv gespeichert haben, denn sie werden in der Steuerung vollständig gelöscht.

□ Parameter

Die Parameter im Menü *Steuerung* sind in zwei Gruppen unterteilt: Globale Parameter, die für die gesamte Steuerung gelten und Achsparameter.

Detaillierte Informationen, Parameterwerte und Werkseinstellungen finden Sie in der Parameter-Referenz. Oder drücken Sie [F1]**,** wenn der Cursor in einem der Eingabefelder steht und die Informationen zu diesem Parameter werden eingeblendet.



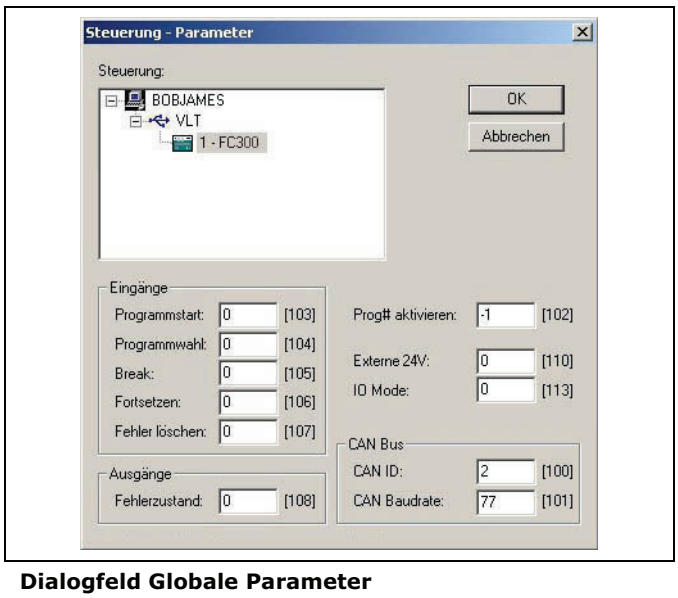
Parameter > Global

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.

Markieren Sie die Steuerung die Sie bearbeiten wollen. Sie können jeden voreingestellten Wert einzeln verändern. Klicken Sie auf *OK* um die Änderungen als Benutzerparameter in die Steuerung zu laden.

Wählen Sie die Steuerung aus, die Sie bearbeiten wollen und ändern Sie die gewünschten Parameterwerte. Klicken Sie auf *OK* um die Änderungen in die Steuerung zu laden.

ACHTUNG!:
Beachten Sie: Sie können eine andere Steuerung auswählen, nachdem Sie Werte geändert haben und bevor Sie *OK* drücken. Falls Sie das tun, werden die geänderten Werte zur vorherigen Steuerung geschrieben und die neuen Steuerungswerte werden gelesen und in den Feldern dargestellt.



Dialogfeld Globale Parameter

MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

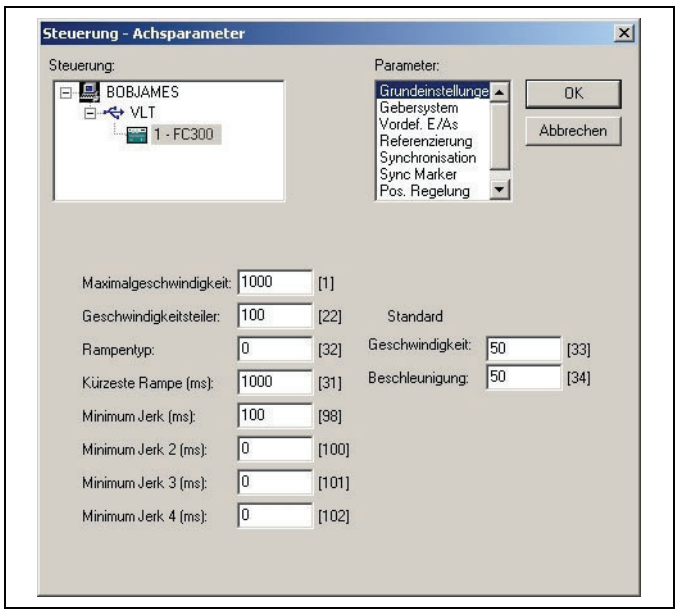
Parameter > Achsen

Klicken Sie auf *Steuerung* → *Parameter* → *Achsen* und wählen Sie den FC 300 aus, den Sie bearbeiten wollen.

Wählen Sie außerdem im Feld *Parameter* den Typ aus:

Grundeinstellungen	– Par. Gruppe 32-8*
Gebersystem	– Par. Gruppe 32-0*, 32-3*
Vordef. Ein-/Ausgänge	– Par. Gruppe 33-4*, 33-5*
Referenzierung (Homefahrt)	– Par. Gruppe 33-0*
Synchronisation	– Par. Gruppe 33-1*
Sync Marker	– Par. Gruppe 33-1*
Positions-Regelung	– Par. Gruppe 32-6*

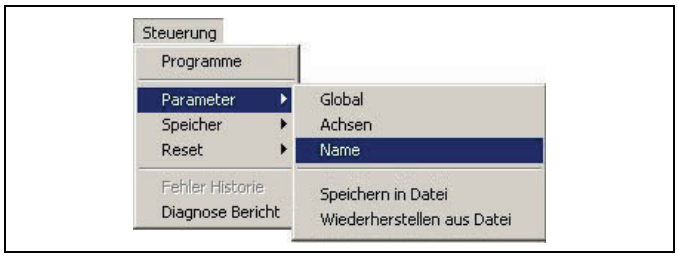
Ändern Sie die gewünschten Parameter. Sie können eine andere Parametergruppe auswählen und auch diese Parameter ändern. Klicken Sie dann auf *OK*, um die Änderungen in den FC 300 zu laden.



ACHTUNG!:
Beachten Sie, dass Sie nach der Änderung und bevor Sie auf *OK* klicken eine andere Steuerung auswählen können: Die geänderten Werte werden in die Steuerung geschrieben und die Werte der neuen Steuerung gelesen und dargestellt.

Parameter > Name

Sie können zusätzlich zur Nummer für jede Steuerung einen Namen eingeben oder einen vorhandenen mit dieser Funktion ändern. Dieser Name wird in den diversen Dialogen in der APOSS Benutzeroberfläche dargestellt und erleichtert die Unterscheidung, wenn mehrere Steuerungen benutzt werden.



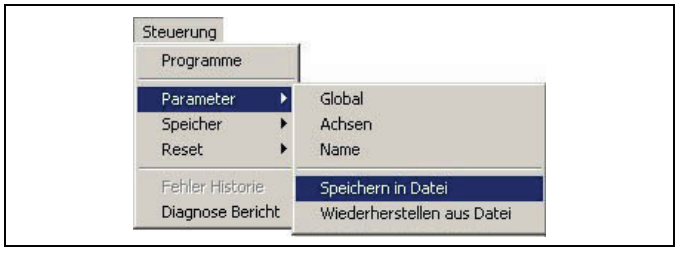
Klicken Sie auf *Steuerung* → *Parameter* → *Name* und wählen Sie im Dialogfeld die Steuerung aus, die Sie bezeichnen wollen; geben Sie einen maximal 8-stelligen Namen in Großbuchstaben ein und klicken auf *OK*.

ACHTUNG!:
Sie können eine andere Steuerung auswählen, nachdem Sie den Namen geändert haben, jedoch bevor Sie auf *OK* klicken. Wenn Sie das tun, wird der geänderte Name zur vorherigen Steuerung geschrieben und die neu ausgewählte Steuerung wird eingelesen und dargestellt.

Parameter > Speichern in Datei

Für Backup-Zwecke kann die gesamte Steuerungskonfiguration (d.h. die globalen Parameter, Achsenparameter, Benutzerparameter und Arrays) als Datei in den PC geladen und gespeichert werden.

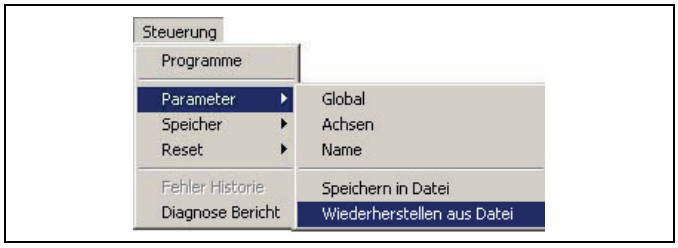
Diese Datei kann später wieder in den FC 300 heruntergeladen werden, falls die Konfiguration wiederhergestellt werden muss.



Wählen Sie die Steuerung aus, klicken Sie dann auf → *Sichern* und geben Sie einen Namen in das folgende Dialogfeld ein. Falls Parameter von mehreren FC 300 gesichert werden sollen, dann wählen Sie einfach nacheinander die Steuerungen aus und → *Sichern* erneut.

Parameter > Wiederherstellen aus Datei

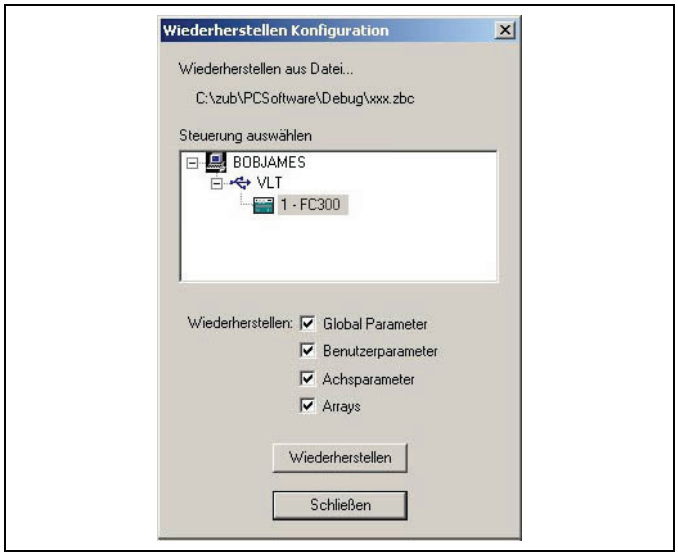
Wenn eine Steuerungskonfiguration als Backup-Datei im PC gespeichert wurde, kann die Konfiguration durch Downloaden dieser Konfigurationsdatei wiederhergestellt werden.



Klicken Sie auf *Parameter* → *Wiederherstellen aus Datei* und wählen Sie im folgenden Dialogfeld die Datei aus, die geladen werden soll.

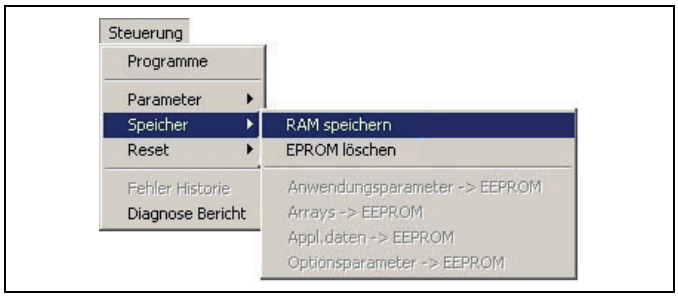
Im nächsten Dialogfeld wählen Sie den FC 300 aus, in den die Daten geladen werden sollen. Markieren Sie im Dialogfeld, welche Parameter geladen werden sollen. Dann klicken Sie auf → *Wiederherstellen*. Die ausgewählten Parameter werden in den FC 300 geladen. Alle früheren Parameter werden überschrieben.

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



□ Speicher

Je nachdem welche Steuerung angeschlossen ist, sind die Funktionen im ersten oder zweiten Block aktiviert. Ist zum Beispiel eine ältere Version eines VLT 5000 angeschlossen, werden die ersten beiden Menü-Befehle benutzt. Ist dagegen eine MCO 305 Option angeschlossen, werden die vier Befehle des zweiten Blocks benutzt.



RAM speichern

Mit *RAM speichern* werden alle Programme, Parameter und Arrays vom RAM in das EPROM gespeichert. Dies entspricht dem Befehl SAVEPROM. Daher wird die Funktion normalerweise nur zum Sichern von Arrays benötigt, denn Programme und Parameter werden automatisch gesichert. Alle Daten, die nicht vom RAM in das EPROM gespeichert wurden, gehen verloren, wenn die Steuerung ausgeschaltet wird.

ACHTUNG!:
Einige sehr alte Versionen von Steuerungen speichern nicht automatisch Programme in das EPROM. Benutzen Sie in diesem Fall die Funktion, um Programme in das EPROM zu speichern.

EPROM löschen

Mit *EPROM löschen* werden alle Parameter auf die Init-Werte zurückgesetzt und alle Arrays sowie Programme gelöscht. Die Steuerung wird auf die Werkseinstellungen zurückgesetzt (Reset). Allerdings erst nach dem Aus- und Wiedereinschalten des FC 300.

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

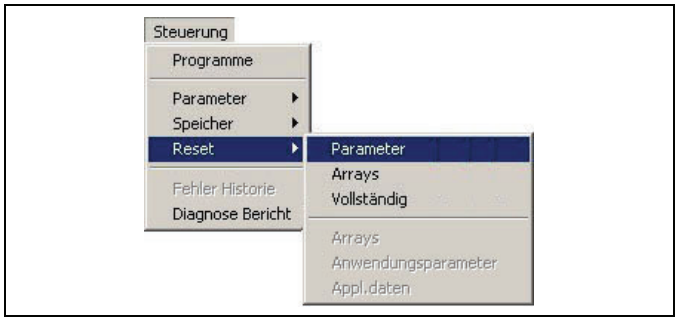
1. Prüfen Sie, ob Sie alle noch benötigten APOSS-Programme auf dem PC gesichert haben. Diese werden benötigt, um sie nach dem Wiedereinschalten wieder in den FC 300 zu laden.
2. Prüfen Sie, ob Sie die Steuerungskonfiguration in eine Backup-Datei auf dem PC gesichert haben (mit *Parameter* → *Speichern in Datei*). Mit dieser Datei können Sie die vorherigen Parameter und Arrays, falls erforderlich, wieder laden.
3. Klicken Sie auf Speicher → *EPROM löschen*.
4. Laden Sie die benötigten Konfigurationsparameter und APOSS-Programme wieder in die Steuerung.

Anwendungsparameter, Arrays, Appl.daten, Optionsparameter -> EPROM

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-EPROM zu speichern.

□ Reset

Je nachdem welche Steuerung angeschlossen ist, sind die Funktionen im ersten oder zweiten Block aktiviert. Ist zum Beispiel eine ältere Version eines VLT 5000 angeschlossen, werden die ersten drei Menü-Befehle benutzt. Ist dagegen eine MCO 305 Option angeschlossen, werden die drei Befehle des zweiten Blocks benutzt.



Parameter

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

Arrays

Mit *Reset* → *Arrays* können Sie alle Arrays im RAM löschen. Diese Funktion bewirkt das Gleiche, wie der Befehl DELETE ARRAYS.

ACHTUNG!:
Wenn Sie anschließend *Speicher* → *RAM speichern* oder ein APOSS-Programm führt einen SAVE ARRAYS Befehl aus, werden auch die Arrays im EPROM überschrieben!

Vollständig

Mit *Reset* → *Vollständig* werden alle Parameter zurückgesetzt und alle Programme und Arrays gelöscht. Die MCO 305 Option wird auf Werkseinstellungen zurückgesetzt.

ACHTUNG!:
Reset → *Vollständig* wird sofort ausgeführt; nicht erst nach dem Aus- und Wiedereinschalten der Steuerung wie bei *Speicher* → *EPROM löschen*.

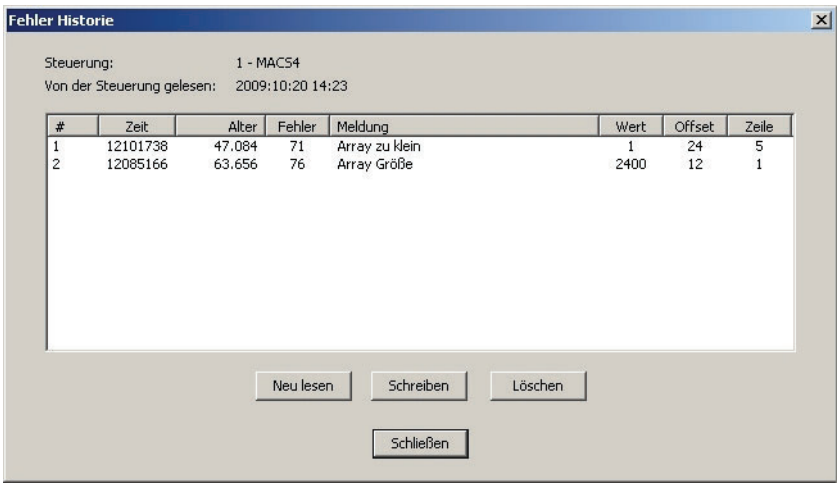
Arrays, Anwendungsparameter, Appl.daten

Benutzen Sie die entsprechende Funktion, um → *Arrays,* → *Anwendungsparameter* oder → *Alle Applikations-Daten* (diese enthalten zusätzlich zu den Anwendungsparametern und Arrays auch das Applikations-Programm) individuell im LCP-EEPROM zurückzusetzen. Der Reset wird sofort ausgeführt!

□ Fehler Historie

Klicken Sie auf *Steuerung* → *Fehler Historie*, um alle Fehler zu sehen, die in der Steuerung auftraten.

ACHTUNG!:
Einige ältere Versionen von Steuerungen unterstützen diese Funktion nicht. Dann ist sie auch nicht aktiviert.



Die letzten Fehler werden am Anfang, die ältesten am Ende der Liste aufgelistet. Die Liste wird jedes Mal gelöscht, wenn die Steuerung ausgeschaltet wird. Sie enthält maximal 50 Fehler; dann wird der älteste Fehler entfernt, sobald ein neuer zur Liste hinzukommt.

Folgende Informationen werden aufgelistet:

Zeit Systemzeit der Steuerung, als der Fehler auftrat. Siehe „sys_clock" in den Systemprozessdaten.

Alter Alter des Fehlers (in Sekunden), wenn das Fehler Historie Dialogfeld geöffnet wird (d.h. wie lange es her ist, als der Fehler auftrat). Es wird jedes Mal aktualisiert, wenn das Dialogfeld aktualisiert wird.

Fehler Fehlernummer. Siehe Kapitel „Fehlersuche und -behebung".

Meldung Text der Fehlermeldung.

Wert Optionaler Wert, der mit dem Fehler gespeichert wird. Die Bedeutung dieses Wertes hängt vom Fehler ab.

Offset Binärer Offset innerhalb des kompilierten Programms, in dem der Fehler auftrat.

Line Zeilennummer des Programms, das bearbeitet wurde, als der Fehler auftrat. Doppelklick auf den Fehler in der Liste schließt den Fehler Historie Dialog und markiert diese Zeile im Editierfenster.

Achtung!:
Die APOSS-IDE kann nicht erkennen, ob das Programm das gerade bearbeitet wird, zu dem Programm gehört, das gerade in der Steuerung ausgeführt wird, als der Fehler auftrat. Das liegt in der Verantwortung des Anwenders. Wenn das Programm nicht dazugehört, wird die falsche Zeile im Editierfenster markiert.

Folgende Funktionen stehen zur Verfügung:

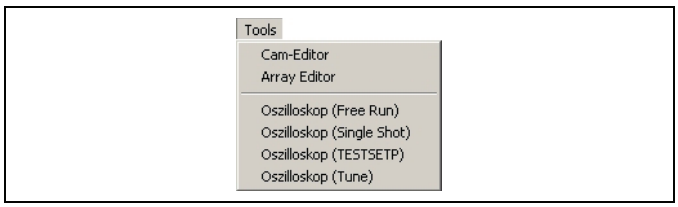
Neu lesen	Aktualisiert die Liste mit der aktuellen Fehler Historie der Steuerung. Beachten Sie, dies aktualisiert das „age" der vorhandenen Fehler in der History.
Schreiben	Schreibt die Fehler Historie in eine ASCII Textdatei, damit die Fehler analysiert werden können. Diese Datei kann später mit einem beliebigen Texteditor bearbeitet oder in ein Tabellenkalkulationsprogramm importiert werden
Löschen	Löscht die Fehler Historie.

□ Diagnose Bericht

Mit *Steuerung* → *Diagnose Bericht* können Sie eine Textdatei erzeugen, die den kompletten Status der Steuerung enthält. Diese Datei wird oft vom Support benötigt, um bei Diagnose Problemen helfen zu können.

□ Menü Tools

Das Menü *Tools* bietet umfangreiche Werkzeuge, die jeweils in eigenen Abschnitten im Kapitel „APOSS Tools" beschrieben werden:
CAM-Editor, Array Editor und *Oszilloskop.*

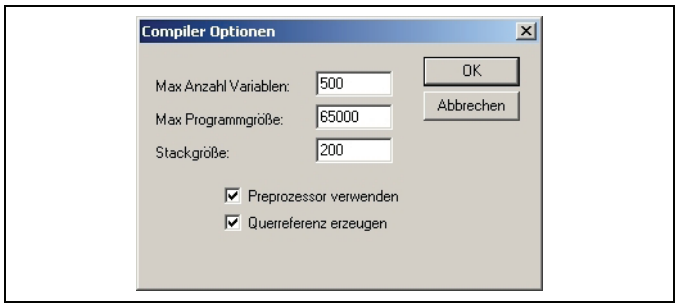


□ Menü Einstellungen

In diesem Menü können Sie die Compiler Optionen auswählen, die Einstellungen der Schnittstelle ändern, die Farben des Editors nach Ihren Wünschen einstellen und verschiedene andere Optionen sowie die Sprache auswählen.

□ Compiler

Die Default-Werte für die Compiler Optionen sind für die meisten Anwendungen passend gesetzt.



Max Anzahl Variablen

Die *Maximale 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.

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 > 65.000 Bytes nicht unterstützen.

Stackgröße

Die *Stackgröße* legt fest wie viel Speicher für die internen Prozeduren (Stack genannt) reserviert wird. Wenn nicht sehr verschachtelte Prozeduraufrufe benutzt werden, sollte dieser Wert nicht geändert werden.

Preprozessor verwenden

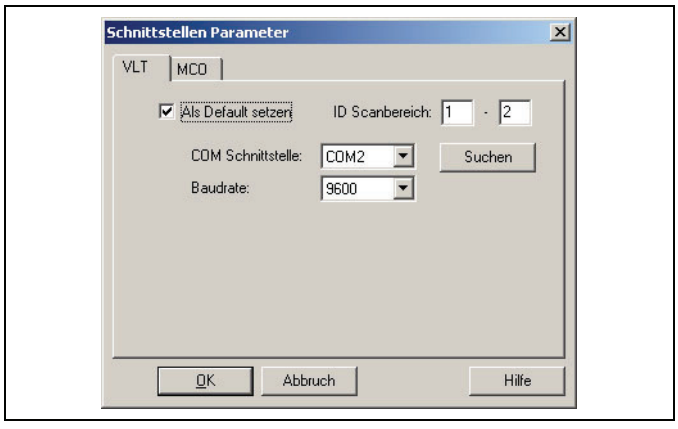
Es können Preprozessor-Statements wie #define innerhalb der Programme verwendet werden; auch verschachtelte Dateien können dann genutzt werden. Komplette Beschreibung siehe Preprozessor in Kapitel „Programmieren mit APOSS".

Querreferenz erzeugen

Querreferenzen sollten immer eingeschaltet sein. Sie sind notwendig für das Funktionieren des Überwachungsfensters und für das Debuggen in APOSS.

□ Schnittstellen Parameter

Normalerweise wurden die Schnittstellen Parameter schon bei der Inbetriebnahme der Steuerung festgelegt (siehe „MCO Parameter für die Grundeinstellungen setzen" im „MCO 305 Produkthandbuch"). Falls jedoch aus irgendeinem Grund die Verbindung geändert werden muss (z.B. wenn sich die Baudrate oder der COM Port geändert hat, wenn der ID Scanbereich der Steuerung erweitert werden muss, wenn eine neue Schnittstelle hinzugefügt wurde, usw.) können mit *Einstellungen → Schnittstelle* die Parameter aktualisiert werden. Ein Dialogfenster ähnlich diesem wird dargestellt:



Wählen Sie die passende Schnittstelle aus und ändern Sie die notwendigen Parameter.

MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

ACHTUNG!:
Beachten Sie, dass nur eine Schnittstelle als Default gewählt werden kann. Diese Schnittstelle wird benutzt um die Verbindungen zu den Steuerungen nach einem [Esc] wiederherzustellen.

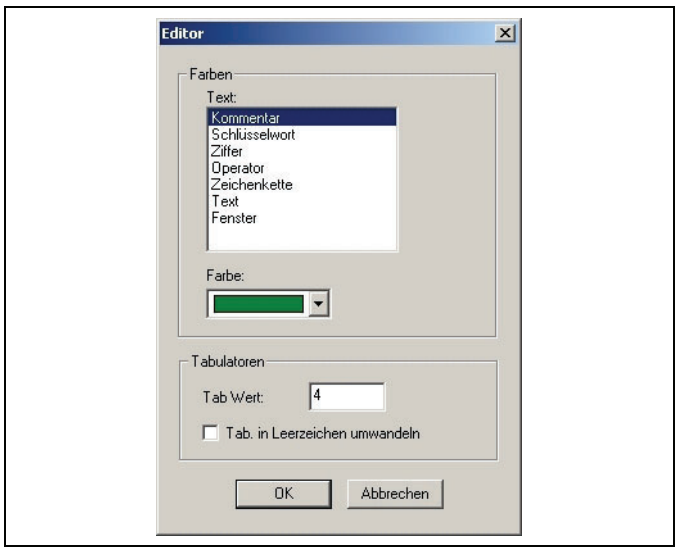
ACHTUNG!:
Die Einstellung der Schnittstelle wirkt sich nur auf künftige Verbindungen aus. Verbindungen, die schon vorhanden sind, bleiben erhalten und werden nicht geändert. Wenn eine bereits vorhandene Verbindung zu einer Steuerung geändert werden muss, dann müssen Sie diese → *Schnittstelle schließen,* die Schnittstellen-Einstellungen ändern und danach erneut die → *Steuerung auswählen.*

□ Sprache

Wenn Sie eine andere Sprache wünschen, klicken Sie auf *Einstellungen* → *Sprache* und wählen im darauf folgenden Dialogfeld zwischen Englisch und Deutsch. Danach müssen Sie APOSS schließen und neu starten.

□ Editor Einstellungen

Zur besseren Übersicht können den verschiedenen Programmelementen wie Kommentar, Schlüsselwort, Ziffer usw. unterschiedliche Farben zugeordnet werden. Auch Tabulatoren können für eine bessere Lesbarkeit genutzt werden.



Farbe

Markieren Sie das Programmelement, z. B. Kommentar und wählen Sie dazu die gewünschte Farbe.

Tabulatoren

Wählen Sie den gewünschten Tabulator-Wert ab zwei Zeichen aufwärts.

Tabulatoren konvertieren

Klicken Sie auf → *Tab in Leerzeichen umwandeln*, um alle Tab-Zeichen während des Schreibens in Leerzeichen umzuwandeln.

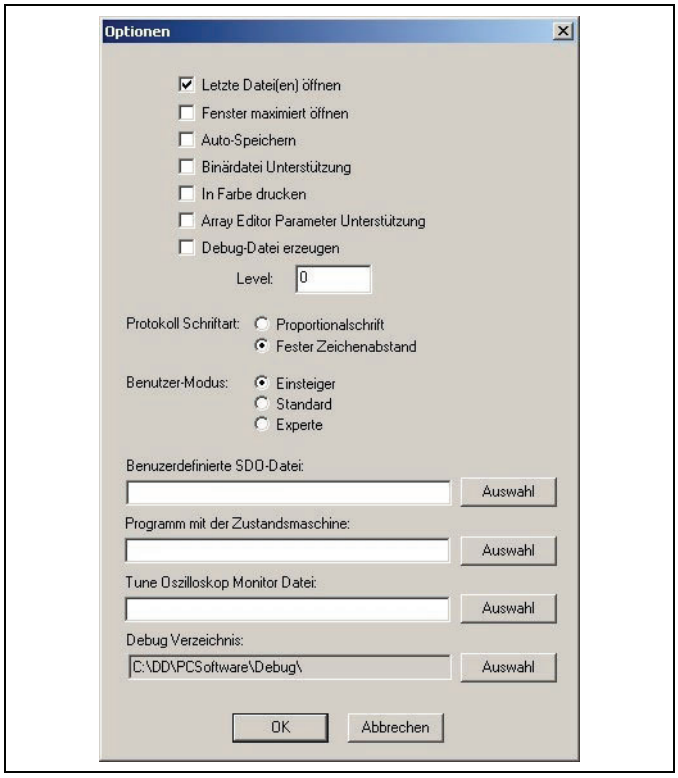
Benutzen Sie *Bearbeiten* → *Tab. Umwandeln*, wenn Sie alle vorhandenen Tabs eines Programms in Leerzeichen umwandeln wollen.

Einstellungen speichern

Klicken Sie auf OK um die neuen Einstellungen zu speichern.

□ Optionen

Das Dialogfenster bietet verschiedene Optionen um das Verhalten von APOSS zu steuern. Geänderte Einstellungen werden sofort beim nächsten Einsatz genutzt. Es ist nicht notwendig APOSS zu schließen und wieder zu öffnen.



Letzte Datei(en) öffnen

Wenn diese Option aktiviert ist, wird APOSS beim Starten versuchen, die meisten der kürzlich verwendeten Dateien wieder zu öffnen.

Fenster maximiert öffnen

Wenn diese Funktion aktiviert ist, werden alle Fenster in APOSS in Bildschirmgröße geöffnet. Sie können danach falls gewünscht verkleinert werden; sie werden immer in Bildschirmgröße geöffnet.

Ist diese Funktion nicht aktiviert, wird nur das erste Fenster in maximaler Größe geöffnet und alle weiteren immer ein wenig kleiner, so dass alle Fenster gleichzeitig zu sehen sind.

Auto-Speichern

Wenn *Auto-Speichern* aktiviert ist, speichert APOSS die aktuelle Datei, die gerade editiert wurde, automatisch auf die Festplatte, bevor diese kompiliert und ausgeführt wird. Ist *Auto-Speichern* nicht aktiviert, benutzt APOSS zum Kompilieren eine temporäre Datei.

Binär Datei Unterstützung

Wenn aktiviert, erlaubt APOSS das direkte Handling der binären Daten (Image) eines kompilierten Programms im Fenster *Steuerung* → *Programme*. Dies beinhaltet den Upload kompilierter Programme von der Steuerung, so dass diese im PC gespeichert werden können und beinhaltet das Downloaden früherer gespeicherter Images zurück in die Steuerung. Beachten Sie, dass diese binären Images nicht bearbeitet werden können; sie können nur gesichert und zurück geladen werden. Beachten Sie auch, dass einige ältere Steuerungen diese Funktion nicht unterstützen.

Wenn die Option aktiviert ist, erscheint die Funktion → *In Datei kompilieren* auch im Menü *Entwicklung* (sobald APOSS das nächste Mal wieder gestartet wird). Dann können Sie die aktuelle Datei manuell kompilieren und in eine Binär-Datei sichern.

Siehe auch BinFileMap in „Abbildungen" im Kapitel „Technische Referenz" im Handbuch „MCO 305 Befehlsreferenz".

In Farbe drucken

Wenn aktiviert, werden die Programme aus dem Editierfenster mit den Farben gedruckt, wie sie im Editierfenster benutzt wurden. Ansonsten werden sie Schwarzweiß gedruckt.

Array Editor Parameter Unterstützung

Normalerweise liest und schreibt der Array Editor alle Benutzerparameter und Arrays. Wenn jedoch diese Einstellung aktiviert ist, liest und schreibt der Array Editor auch die globalen Parameter, die temporären und permanenten Achsenparameter, die Achsprozessdaten und Systemprozessparameter.

Debug-Datei erzeugen

Aktivieren Sie diese Funktion, wenn Sie eine Debug-Datei mitschreiben lassen wollen. Normalerweise wird dies nur für den Support benötigt; daher sollte die Option in der Regel nicht aktiviert sein.

Protokoll Font

Zeigt die Meldungen im Kommunikations-Fenster als → *Proportionalschrift* oder mit → *festem Zeichenabstand*. Eine Änderung der Auswahl wirkt sich erst in neu geöffneten Fenstern aus.

Benutzer-Modus

Mit der Einstellung des Benutzer-Modus kann APOSS auf den Erfahrungsgrad des Anwenders zugeschnitten werden: Einsteiger, Standard oder Experte. Zurzeit gilt diese Einstellung hauptsächlich für die *Oszilloskop* Funktionen.

Anwenderspezifische Bereiche

Benutzerdefinierte SDO Dateien erlauben dem erfahrenen Anwender, die Inhalte der SDO Auswahllisten, die in APOSS benutzt werden, individuell anzupassen. Zum Beispiel jene, die im Überwachungsfenster und im *Oszilloskop* benutzt werden.

Mittels der Einstellung *Programm mit der Zustandsmaschine* kann der erfahrene Anwender ein kundenspezifisches Programm mit der Zustandsmaschine im *Tune Oszilloskop* benutzen.

In *Tune Oszilloskop Monitor Datei* kann der erfahrene Anwender ein kundenspezifisches Testfenster auswählen und im *Tune Oszilloskop* benutzen.

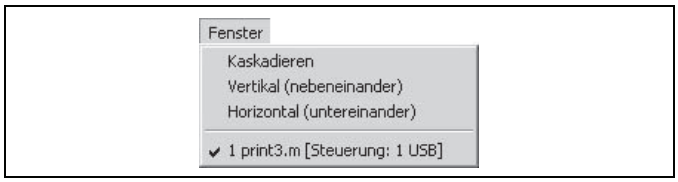
Im *Debug Verzeichnis* kann der Anwender festlegen, wo die Debug Logdateien gespeichert werden sollen.

□ Oszilloskop

Dieser Dialog ermöglicht die Auswahl von verschiedenen Optionen, um das Verhalten des APOSS Oszilloskop zu steuern. Siehe „Oszilloskop Einstellungen” in Kapitel „APOSS Tools”, APOSS Oszilloskop.

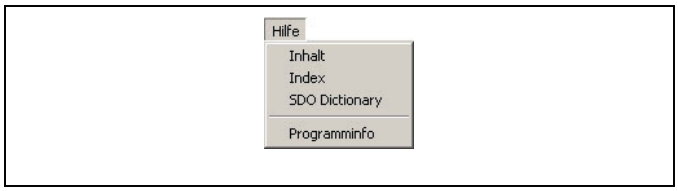
□ Menü Fenster

Die Funktionen des Menüs *Fenster* verhalten sich Windows-konform (d.h. *Kaskadieren* für sich überlappende Fenster nach unten und nach rechts versetzt, *Vertikal* für jedes neben dem anderen oder *Horizontal* für jedes unter dem anderen).



□ Menü Hilfe

Die *Befehlshilfe* und alle Parameter-Dialogfelder im Menü *Steuerung* sowie im CAM-Editor bieten einen direkten Zugang zur Online-Hilfe. Wählen Sie einen Befehl in der *Befehlshilfe* oder ein Parameter-Eingabefeld aus und drücken Sie [F1]. Sie erhalten dann den entsprechenden Abschnitt der Hilfe direkt angezeigt.



Die Funktion → *Inhalt* startet die Online-Hilfe mit der Registerkarte „Inhalt”,

die Funktion → *Index* startet sie mit der Registerkarte „Index”.

Wählen Sie → *SDO Dictionary***,** dann wird sofort dieses Kapitel der Online-Hilfe aufgerufen.

Die Funktion → *Programminfo* zeigt die Versionsnummern des APOSS-Programms, des Interface-Treibers und des Compilers.

Das Aussehen der Online-Hilfe hängt vom eingesetzten Betriebssystem ab.

□ Menü Download

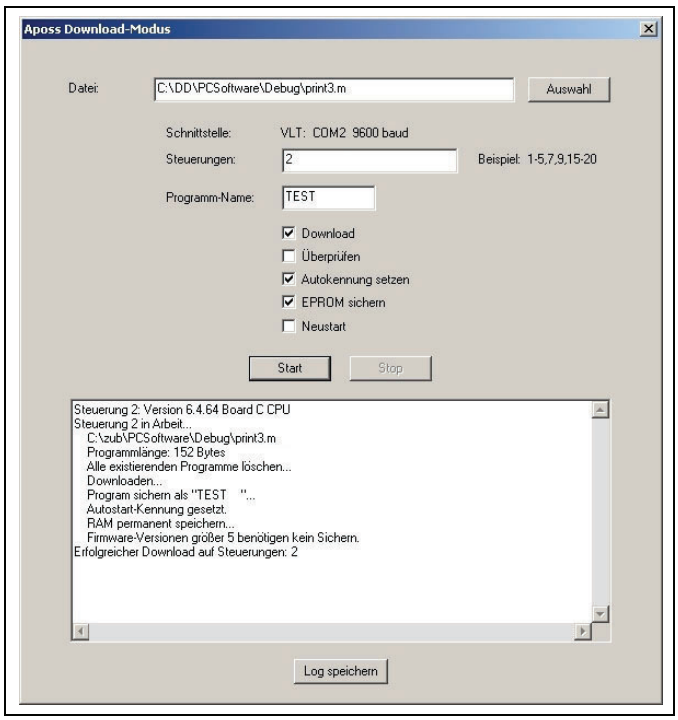
Das Menü *Download* bietet eine einfache Schnittstelle um das Downloaden von Firmware und Programmen in mehrere Steuerungen zu unterstützen.

Beachten Sie, dass alle APOSS Fenster, mit Ausnahme des APOSS-Fensters selbst, geschlossen sein müssen, bevor dieses Menü zu sehen ist.

Wählen Sie → *Programme* und im folgenden Dialogfenster das APOSS Programm das heruntergeladen werden soll. Falls die Schnittstelle mehrere Steuerungen unterstützt, dann geben Sie den ID-Bereich der Steuerungen ein, in die das Programm geladen werden soll.

Geben Sie einen Programmnamen (bis zu 8 Zeichen) ein, der zum Identifizieren des Programms in der Steuerung benutzt wird.

Wählen Sie die gewünschten Aufgaben in den Checkboxen aus: *Download, Überprüfen, Autokennung setzen, EPROM sichern* und/oder *Neustart*. Wenn Sie zum Beispiel nur *Überprüfen* aber nicht *Downloaden* wollen, wird das Download-Programm nur überprüfen, ob die ausgewählte Datei diejenige ist, die schon in der Steuerung ist.



Start Download Programme

Klicken Sie auf *Start*. Es wird mit der Default-Schnittstelle eine Verbindung zu den Steuerungen hergestellt. Alle Steuerungen, die gerade ein Programm ausführen, werden angehalten. Die folgenden Schritte werden für jede erreichbare Steuerung nacheinander ausgeführt.

Sie könnten zum Beispiel auch nur *Überprüfen* auswählen. Dann wird nur geprüft, ob die ausgewählte Datei diejenige ist, die schon in der Steuerung ist.

- Das Programm wird mit den zur Steuerung gehörenden Compiler-Einstellungen kompiliert.
- Alle vorhandenen Programme der Steuerung werden gelöscht.
- Das Programm wird heruntergeladen.
- Das Programm wird als Programm Nummer 0 mit dem festgelegten Programmnamen gespeichert. Ist kein Programmname festgelegt, werden die ersten 8 Zeichen des Dateinamens benutzt.
- Wenn *Überprüfen* ausgewählt ist, wird das heruntergeladene Programm wieder hochgeladen und die Download- und Upload-Versionen miteinander verglichen. Falls sie voneinander abweichen, wird der Download für diese Steuerung angehalten.
  Beachten Sie, dass nicht alle älteren Steuerungen eine Firmware enthalten, die ein Upload unterstützen. In diesem Fall wird die Auswahl der Funktion *Überprüfen* ignoriert.
- Die *Autokennung* wird gesetzt, falls ausgewählt.
- Alle Daten werden *im EPROM* gespeichert, falls ausgewählt. Bei neueren Steuerungen wird dies automatisch durchgeführt.
- Die Steuerung wird mit dem neuen Programm *neu gestartet*, falls ausgewählt.

Die Verbindung zu den Steuerungen wird geschlossen.

Treten während des Ausführens in einer Steuerung Fehler auf, wird die Ausführung für diese Steuerung angehalten, aber für alle anderen Steuerungen des ausgewählten ID-Bereichs fortgeführt. Wenn der gesamte Download fertig ist, wird eine Zusammenfassung im Dialogfeld ausgegeben. Diese zeigt die IDs von den Steuerungen, die ohne Fehler und jene bei denen Fehler auftraten. Sie können durch die Zusammenfassung scrollen, um festzustellen welche Probleme auftraten.

Log speichern

Der Fortgang des Downloads wird in dem unteren Teil des Dialogfelds gezeigt. Falls gewünscht, können diese Informationen mit → *Log speichern* in eine Textdatei gespeichert werden.

Falls der Download für eine oder mehrere Steuerungen fehlschlägt, erhalten Sie eine entsprechende Fehlermeldung. Wenn Sie die *Log*-Datei *speichern*, können Sie später die Fehler suchen und bearbeiten.

□ Programme debuggen

Die APOSS-IDE enthält einen mächtigen integrierten Debugger. Dieser bietet allgemeine Debug-Funktionen wie Einzelschritt, Breakpoints und die Möglichkeit, Programmvariablen zu lesen und zu setzen.

ACHTUNG!:
Der Debugger kann nicht in allen Fällen benutzt werden. Zum Beispiel ist es nicht möglich, den Debugger mit Programmen zu benutzen, die aktiv einen Motor steuern. Das Stoppen der Programmausführung an einem Breakpoint ist gleichbedeutend wie das Drücken der [Esc]-Taste, um die Programmausführung abzubrechen. Dies bremst und stoppt den Motor mit der maximal erlaubten Verzögerung. In vielen Fällen macht das Anhalten des Motors auf diese Art die Testprozedur ungültig und die Debug-Ergebnisse wertlos. Ebenso ist es unwahrscheinlich, dass wenn die Programmausführung nach einem Breakpoint fortgesetzt wird, dass der Motor korrekt wieder gestartet werden kann um das System in den Status zu bringen, in dem es vor dem Breakpoint war.

ACHTUNG!:
Der Debugger kann auch nicht mit Programmen benutzt werden, die auf ON PERIOD Funktionen beruhen. Der interne Timer, der ON PERIOD Funktionen triggert, stoppt nicht wenn die Pogramm-ausführung an einem Breakpoint anhält. So könnte ein anstehender Interrupt dann einen ON PERIOD Aufruf auslösen, wenn die Programmausführung fortgesetzt wird.

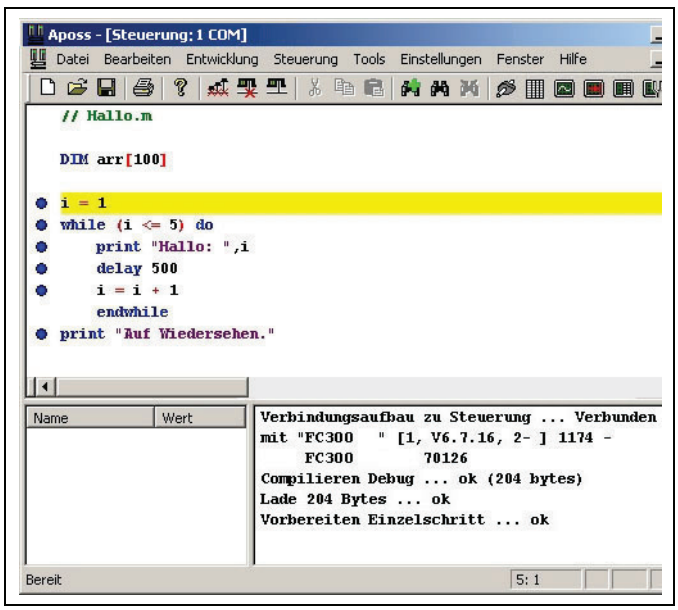
Für Situationen, in denen der Debugger wie die oben nicht benutzt werden kann, bietet das Oszilloskop ausgezeichnete Debug-Einsatzmöglichkeiten. Es kann Programmvariablen und den Systemstatus überwachen und aufzeichnen, ohne dass die Programmausführung unterbrochen werden muss. Diese können dann später geprüft werden, um Probleme zu identifizieren. Für mehr Information siehe APOSS Oszilloskop.

□ Debugger starten

Zum Starten des Debuggers bearbeiten Sie das Programm ganz normal, so dass es im Editierfenster dargestellt wird. Dann klicken Sie auf *Entwicklung → Vorbereiten Einzelschritt.* Das bewirkt folgende Aktionen:

1. Das Programm wird im Debug-Modus kompiliert und in die Steuerung heruntergeladen. Die Programmausführung wird jedoch zu diesem Zeitpunkt nicht gestartet.
2. Vor jeder ausführbaren Anweisung im Programm wird ein blauer Punkt gesetzt. Dies sind die Positionen, an denen der Anwender Breakpoints einfügen kann.
3. Die nächste Anweisung die ausgeführt wird (d.h. wenn die Ausführung gestartet ist oder fortgesetzt wird) wird gelb markiert.

So könnte das Editierfenster dann aussehen:



ACHTUNG!:
Das Programm sollte nicht verändert werden, während der Debugger aktiv ist. Falls doch, stimmt die APOSS-IDE nicht mehr mit der Programmversion übereinstimmen, die in der Steuerung ausgeführt wird und der Debugger ist nicht mehr in der Lage, der Programmausführung korrekt zu folgen. Falls das Programm geändert werden muss, sollte der Debugger angehalten und der Test erneut von Anfang an gestartet werden.

☐ Debugger anhalten

Um den Debugger anzuhalten und die Debug-Sitzung zu beenden klicken Sie auf *Entwicklung* → *Beenden Debug.* Dies entfernt die blauen Punkte, die die ausführbaren Anweisungen markieren. Beachten Sie, dies entfernt keine (rot markierten) Breakpoints. Breakpoints bleiben erhalten und können benutzt werden, wenn der Debugger das nächste Mal gestartet wird. Siehe auch Breakpoints anwenden.

☐ Einzelschritt

Um mit einzelnen Schritten durch das Programm zu gehen, benutzen Sie *Entwicklung* → *Einzelschritt* oder drücken [F9]. Dies führt die nächste Anweisung aus (d.h. die Anweisung, die gerade gelb markiert ist) und stoppt automatisch bevor die nächste Anweisung ausgeführt wird. Dann wird diese nächste Anweisung gelb markiert.

Während die Ausführung angehalten ist, steht es dem Anwender frei den Wert einer beliebigen Programmvariablen zu prüfen und zu verändern, Breakpoints zu setzen oder zu entfernen, etc.

Zu jeder Zeit kann die Programmausführung ohne Einzelschritt mit *Entwicklung* → *Start* fortgeführt werden.

☐ Breakpoints anwenden

Zum Setzen von Breakpoints doppelklicken Sie irgendwo in die Anweisung im Programm (außer auf den blauen Punkt), vor der der Breakpoint stehen soll. Der blaue Punkt ändert sich Rot und zeigt so den gesetzten Breakpoint an.

Erneutes Doppelklicken in eine Anweisung, die schon einen Breakpoint enthält, entfernt diesen wieder und der Punkt ist wieder blau markiert.

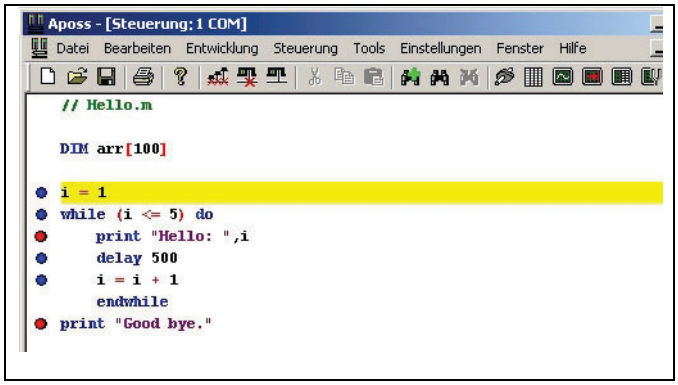
An jeder ausführbaren Anweisung im Programm (mit blauen Punkten markiert) kann ein Breakpoint gesetzt werden. Wenn die Programmausführung auf einen Breakpoint trifft, wird die Ausführung sofort vor der Anweisung mit dem Breakpoint angehalten. Die Anweisung wird dann gelb markiert, da es die nächste auszuführende Anweisung ist.

Während die Ausführung angehalten ist, steht es dem Anwender frei den Wert einer beliebigen Programmvariablen zu prüfen und zu verändern, Breakpoints zu setzen oder zu entfernen, etc.

Wenn der Anwender damit fertig ist, kann er mit *Entwicklung* → *Start* die Programmausführung fortzusetzen. Das Programm wird dann so lange ausgeführt, bis es auf einen anderen Breakpoint trifft. Es ist auch möglich, nach dem Anhalten bei einem Breakpoint mit *Entwicklung* → *Einzelschritt* [F9] fortzufahren.

Zu jeder Zeit kann der Anwender die Programmausführung durch Drücken der [Esc]-Taste anhalten. Die Ausführung stoppt sofort, nicht erst beim nächsten Breakpoint. Die nächste auszuführende Anweisung wird gelb markiert. Auch in diesem Fall kann der Anwender entweder mit *Entwicklung* → *Start* oder → *Einzelschritt* [F9] fortfahren.

Dieses Beispiel zeigt Breakpoints vor den PRINT Anweisungen:



ACHTUNG!:
Maximal 10 Breakpoints sind erlaubt.

ACHTUNG!:
Die Breakpoints haben die Befehle #DEBUG on/off ersetzt. Vorhandene #DEBUG Befehle müssen nicht unbedingt aus dem Programm entfernt werden, da sie ignoriert werden.

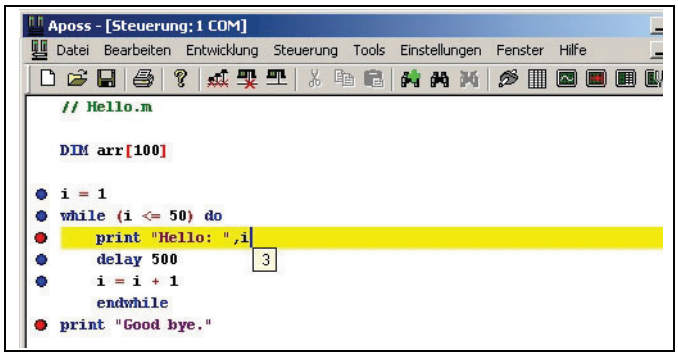
MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

☐ Darstellung und Änderung von Variablen

Zu jeder Zeit, während der Debugger aktiv ist (d.h. auch wenn das Programm ausgeführt oder angehalten ist), kann der aktuelle Wert einer Programmvariablen dargestellt werden.

Klicken Sie dazu mit der linken Maustaste auf (oder direkt hinter) eine Variable. Der aktuelle Wert wird in einer gelben Popupbox so lange dargestellt, bis die Maus davon wegbewegt wird.

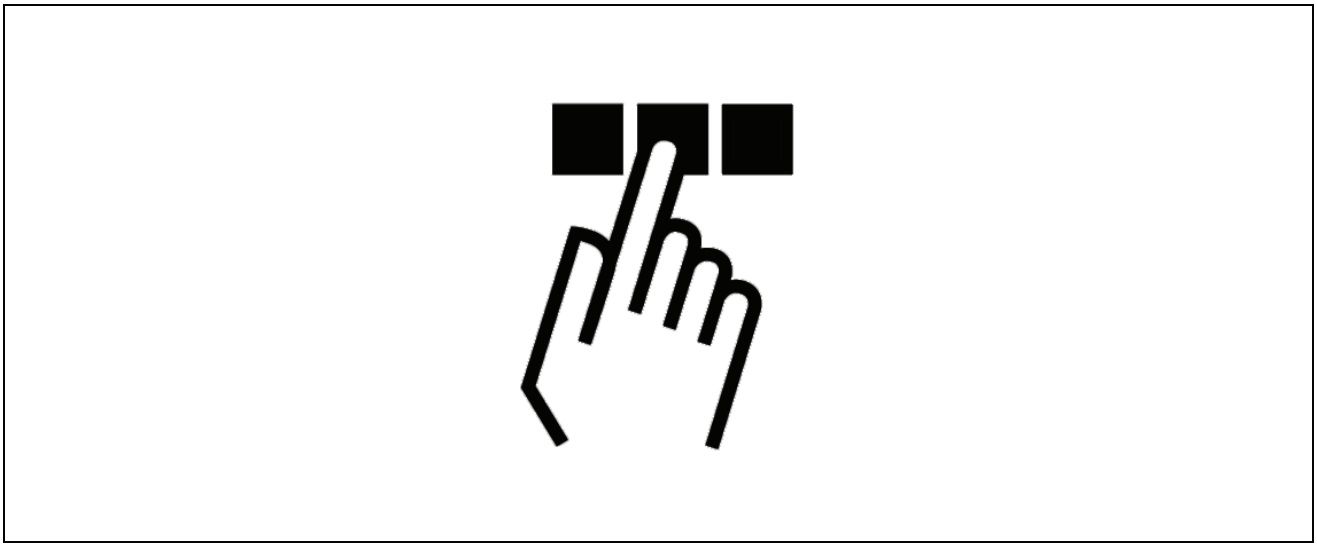
Im folgenden Beispiel wurde nahe der Variablen „i" in der PRINT Anweisung geklickt.



Zu jeder Zeit, während der Debugger aktiv ist, kann der aktuelle Wert einer Programmvariablen auch geändert werden. Klicken Sie dazu mit der rechten Maustaste auf (oder direkt hinter) die Variable.

Der Dialog „Debug Wert setzen" ermöglicht die Änderung des Wertes der Variablen:

APOSS Tools



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. Die Kurven und Parameter werden im Kurvenprofil grafisch dargestellt und können sowohl grafisch als auch per Eingabe bearbeitet werden.

Im Kapitel „Funktionen und Beispiele" finden Sie Anwendungsbeispiele von Kurvenscheibensteuerungen und Nockenschaltwerk.

Jedes Kurvenprofil wird in einem eigenen Globalen Array in der Steuerung gespeichert.

Der CAM-Editor speichert die Kurvenprofile und alle Parameterwerte in einer Konfigurationsdatei mit der Erweiterung „zbc", die im Windows-Ordner das links gezeigte Symbol erhält. Wie andere Dateitypen können Konfigurationsdateien im CAM-Editor später geöffnet, bearbeitet und gespeichert werden.

ACHTUNG!:
Frühere Versionen der APOSS-IDE benutzten die Dateierweiterung „.cnf" für die Konfigurationsdateien. Um Konflikte mit der „cnf" Erweiterung von Microsoft Windows zu vermeiden, wurde diese in „.zbc" geändert. Vorhandene cnf-Dateien werden jedoch unterstützt und können vom CAM-Editor einfach durch normales Öffnen gelesen werden. Sobald eine cnf-Datei einmal vom CAM-Editor geöffnet wurde, wird sie als zbc-Datei gespeichert.

Der CAM-Editor ist so ausgelegt, dass er exakt das gleiche Format der Konfigurationsdatei „.zbc" benutzt, wie es auch von den Befehlen *Steuerung → Parameter → speichern in Datei* und *Wiederherstellen aus Datei* benutzt wird. Daher speichern die Konfigurationsdateien, die mit dem CAM-Editor erzeugt wurden, alle globalen, Benutzer- und Achsparameter zusätzlich zu den Kurvenprofilen. Demzufolge sollten Sie den CAM-Editor immer mit den Parametern von der Steuerung starten, für die die Kurvenprofile vorgesehen sind. Andernfalls könnten die falschen globalen, Benutzer- und Achsparameter in die Steuerung geladen werden, wenn Sie die Kurvenprofile downloaden.

Sobald Kurvenprofile erzeugt und in einer „.zbc" Datei gespeichert wurden, können Sie diese in die Steuerung mit *Steuerung → Parameter → Wiederherstellen aus Datei* downloaden.

Falls Sie keine Parameter laden, werden die Werkseinstellungen der Steuerung eingetragen.

□ CAM-Editor starten

MCT 10 Online- und Offline-Modus

Sie können Konfigurations-Dateien (*.zbc und *.cnf) mit MCT 10 öffnen. Dies startet auch den APOSS CAM-Editor. Je nach Modus sind einige der Menü-Funktionen disabled. Benutzen Sie die Funktionen des MCT 10 Motion Control Tools für *Neu*, *Öffnen* und *Sichern als*.

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

Wenn Sie mit einem neuen Kurvenprofil beginnen und noch keine Konfigurationsdatei existiert, empfiehlt es sich immer mit den Parametern von der Steuerung zu beginnen, für die die Kurvenprofile vorgesehen sind.

ACHTUNG!:
Wenn Sie keine Parameter von der Steuerung einlesen, werden die Werkseinstellungen des FC 300 benutzt. Dann werden die vorhandenen Parameter der Steuerung mit den Werkseinstellungen überschrieben, sobald Sie diese Konfigurationsdatei in die Steuerung downloaden

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

Klicken Sie auf → *Zum Antrieb schreiben,* um die neuen zbc-Werte (insbesondere die CAM-Arrays) in den Antrieb zu downloaden. Die neuen Werte werden auch im MCT 10 Datenbestand gespeichert und überschreiben dabei eine frühere Version der Datei (d.h. speichert diese zurück in den MCT 10 Datenbestand).

APOSS stand-alone

In APOSS stand-alone klicken Sie auf *Tools* → *CAM-Editor* oder auf das Symbol zum Starten des Editors. Ein „Datei öffnen" Dialog bietet die Auswahl der Konfigurationsdatei; dann wird das CAM-Editor-Fenster geöffnet.

Der CAM-Editor kann auch durch *Datei* → *Öffnen* und Auswahl der gewünschten Konfigurationsdatei gestartet werden.

Doppelklicken auf eine Konfigurationsdatei (zbc) in Windows startet ebenfalls den CAM-Editor. Beachten Sie, doppelklicken auf cnf-Dateien startet nicht den CAM-Editor.

Wenn Sie mit einem neuen Kurvenprofil beginnen und noch keine Konfigurationsdatei existiert, empfiehlt es sich immer mit den Parametern von der Steuerung zu beginnen, für die die Kurvenprofile vorgesehen sind. Zwei Vorgehensweisen sind möglich:

1. Verbinden Sie zuerst die Steuerung und erzeugen Sie dann eine Konfigurationsdatei mit *Steuerung* → *Parameter* → *speichern in Datei.* Diese neue Konfigurationsdatei kann dann mit dem CAM-Editor geöffnet werden.
2. Öffnen Sie entweder mit *Datei* → *Neu* oder durch Starten des CAM-Editors eine „neue" Konfigurationsdatei und klicken Sie im folgenden Dialogfeld auf die Schaltfläche → *Leere Datei anlegen.* Klicken Sie auf das Symbol *Von der Steuerung lesen* um die aktuellen Parameter der Steuerung einzulesen.

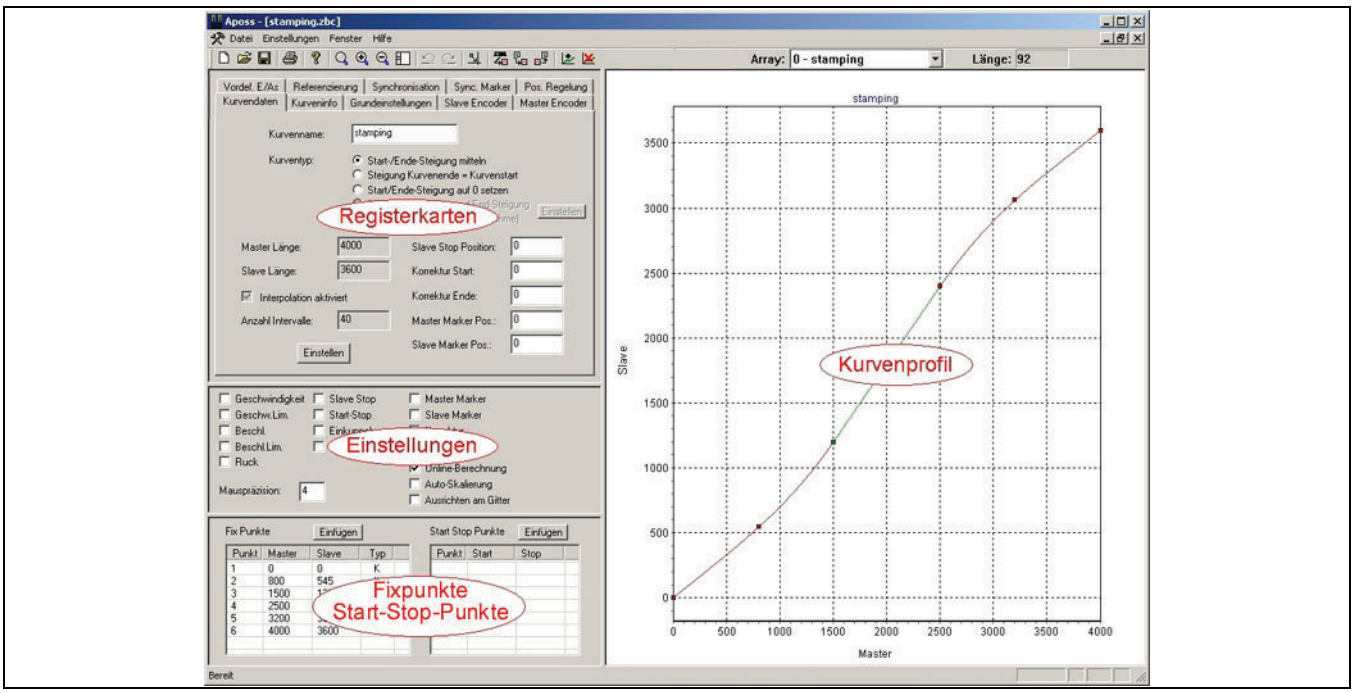
ACHTUNG!:
Wenn Sie keine Parameter von der Steuerung einlesen, werden die Werkseinstellungen des FC 300 benutzt. Dann werden die vorhandenen Parameter der Steuerung mit den Werkseinstellungen überschrieben, sobald Sie diese Konfigurationsdatei in die Steuerung downloaden.

Neue Konfigurationsdateien enthalten keine Kurvenprofile. Benutzen Sie die → *CAM Array hinzufügen***,** um ein neues Kurvenprofil hinzuzufügen.

Sie finden einige Programmbeispiele in *Datei* → *Beispiel.*

MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

□ CAM-Editor-Fenster



Die vier Bereiche nach der Titel-, Menü- und Symbolleiste im *CAM-Editor*-Fenster sind:

- Kurvenprofil-Diagramm
- Registerkarten: *Kurvendaten, Kurveninfo* und alle Registerkarten der Achsparameter, die normalerweise bei *Steuerung* → *Parameter* → *Achsen* zur Verfügung stehen.
- Einstellungen zum Ein- und Ausschalten der Darstellung der verschiedenen Eigenschaften des Kurvenprofil-Diagramms.
- Tabelle der *Fixpunkte* und der *Start-Stop-Punkte.*

Das CAM-Editor-Fenster kann wie üblich beliebig vergrößert oder verkleinert werden. Ebenso können die Trennleisten der vier Fenster mit der Maus verändert werden.
Mit → *Teilung zurücksetzen* kehren Sie zur Standarddarstellung zurück.

CAM Editor Symbolleiste

Die CAM-Editor Symbolleiste enthält folgende vier besondere Felder:

Array – In dieser Liste aller vorhanden definierten Arrays wird das Array ausgewählt, das bearbeitet werden soll. Jeder Listeneintrag besteht aus einer Array-Nummer, gefolgt vom Namen des Kurvenprofils. Wenn ein Array nicht dem Kurvenprofil entspricht, wird als Name „Array" dargestellt. Arrays, die keine Kurven sind, können zwar ausgewählt, aber nicht als Kurven bearbeitet werden.

Länge – Dies zeigt die Mindestlänge, die für das Array erforderlich ist. Dieser Mindestwert wird für die DIM Anweisung im APOSS-Programm benötigt.

Die Symbole, die speziell für den CAM-Editor benötigt werden, sind:

Zoom Zurücksetzen – Stellt das Diagramm so dar, dass die Kurve vollständig zu sehen ist.

Zoom 2-fach verstärken – Jeder Klick vergrößert das Diagramm um Faktor 2 und stellt es immer von der Mitte heraus dar.

Zoomstufe reduzieren – Mit Faktor 2 verkleinern.

Trennleisten zurücksetzen – Setzt die Trennleisten (Splitter) des Fensters auf die Default-Einstellungen zurück.

Rückgängig – Macht die letzte Änderung eines Fixpunktes oder Start-Stopp-Punktes rückgängig. Wenn mehrere Änderungen zur gleichen Zeit durchgeführt wurden (zum Beispiel mehrere Punkte hinzugefügt oder durch Aktivieren der Funktion → *Ausrichten am Raster*) werden diese als Ganzes zurückgenommen.

Beachten Sie – Wenn mehrere Kurven-Arrays existieren, gibt es auch separate Rückgängig-Listen für jede Kurve.

Wiederherstellen – Stellt die letzte rückgängig gemachte Änderung wieder her.

Fixpunkte am Raster ausrichten – Richtet alle Fixpunkte und Start-Stop-Punkte am Interpolations-Raster aus.

Von der Steuerung lesen – Liest die vorhandenen Konfigurationsparameter und alle Arrays und Kurven von einer Steuerung. Dies ersetzt alle gerade im CAM-Editor vorhandenen Parameter und Arrays.

Array importieren – Importiert eine ACSII Textdatei, die zuvor mit → *Array exportieren* ausgelesen wurde. Die importierte Kurve ersetzt die aktuelle ausgewählte Kurve. Das Dateiformat muss exakt der Export-Datei entsprechen; andernfalls schlägt das Einlesen fehl.

Array exportieren – Exportiert die Kurvendaten der aktuell ausgewählten Kurve in eine ASCII Textdatei mit der Dateierweiterung „.dat". Diese eignet sich für die Weiterverarbeitung ohne APOSS (zum Beispiel mit einer Tabellenkalkulation). Für die Beschreibung des genauen Array-Formats schlagen Sie bitte in „Array Structure of CAM Profiles" im Kapitel „Technische Referenz" im Handbuch „MCO 305 Befehlsreferenz" nach.

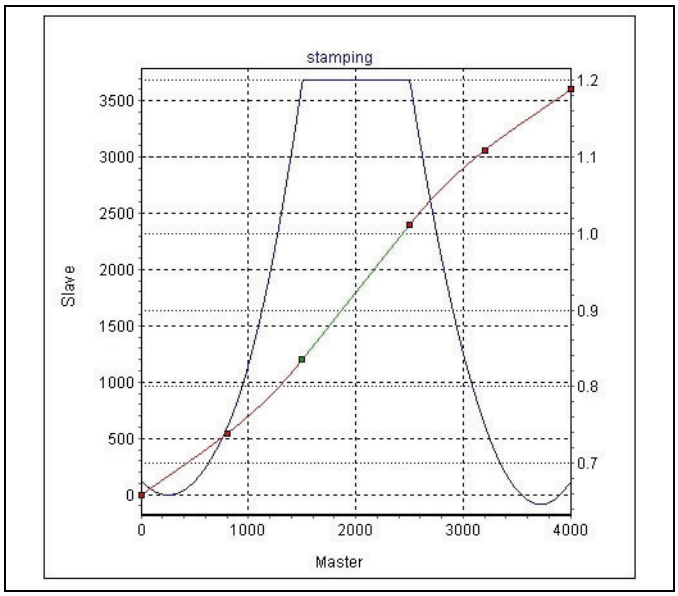
CAM Array hinzufügen – Eine neues Kurvenprofil-Array wird nach dem letzten vorhandenen Array eingefügt. Ein Dialog ermöglicht dem Anwender die ersten Kurveneigenschaften zu setzen. Die neue Kurve wird dann automatisch ausgewählt und dargestellt. Die Anfangskurve enthält nur zwei Fixpunkte und ist eine Gerade. Weitere Fixpunkte sollten dann hinzugefügt werden um die eigentliche Form der Kurve zu definieren.

Array löschen – Das gerade ausgewählte Array wird gelöscht. Beachten Sie, dass jedes Array gelöscht werden kann; das Array muss nicht ein Kurvenprofil-Array sein.

□ Kurvenprofil-Diagramm

Das Kurvenprofil-Diagramm zeigt die Kurve mit weiteren Informationen der Kurvensynchronisation. Hier können Sie Fixpunkte hinzufügen, löschen und mit der Maus verschieben.

- Der Name der Kurve, falls vorhanden, steht über der Kurve.
- Die Masterlänge ist an der horizontalen Achse unten aufgetragen, die Slave-Positionen stehen vertikal und auf der linken Seite. Verschiedene andere Skalierungen (z.B. Geschwindigkeit) werden auf der rechten Seite des Diagramms aufgetragen, falls notwendig.



- Fixpunkte werden als kleine rote oder grüne Quadrate dargestellt.
  „Kurven"-Punkte (d.h. Fixpunkte am Anfang eines gebogenes Kurvensegments im Profil) werden rot und „Tangenten"-Punkte (d.h. Fixpunkte am Anfang eines Tangentensegments) werden grün dargestellt.
- Kurvensegmente sind rot, Tangentensegmente grün gezeichnet.
  Beachten Sie, dass Segmente die Geraden sind, grün dargestellt werden, obwohl sie „Kurvensegmente" sein können (d.h. Segmente, die mit einem „Kurven"-Punkt beginnen).

MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

Zoomen und Verschieben

Die Maus kann im Kurvenprofil-Diagramm für folgende Funktionen genutzt werden:

Zoom

Vergrößern durch Ziehen eines Rechtecks über den gewünschten Bereich (linke Maustaste drücken, ziehen, loslassen).

ACHTUNG!:
Das ausgewählte Rechteck wird so skaliert, dass es exakt in das Fenster passt. Falls diese Skalierung nicht gewünscht ist, drücken und halten Sie die [Strg]-Taste, bevor Sie die linke Maustaste drücken.

Benutzen Sie *Zoomstufe reduzieren* oder *Zoom zurücksetzen* zum Verkleinern bzw. auf Standard zurückzusetzen.

Bei beim Zoomen wird die ☑ *Auto-Skalierung* automatisch deaktiviert.

Verschieben

Mit gedrückter rechter Maustaste können Sie das Diagramm innerhalb des Fensters bewegen. Bei beim Verschieben wird die ☑ *Auto-Skalierung* automatisch deaktiviert.

Mit *Zoom zurücksetzen* wird die Darstellung wieder auf Standard zurückgesetzt. Eine evtl. Vergrößerung wird dabei auch zurückgesetzt, so dass wieder das ganze Diagramm zu sehen ist.

Fixpunkte mit der Maus bearbeiten

Fixpunkte können im Diagramm interaktiv mit der Maus bearbeitet werden. Aber auch die direkte Eingabe in der Tabelle der Fixpunkte ist möglich, siehe „Fixpunkte".

Fixpunkt identifizieren

Zeigen Sie mit dem Mauscursor auf einen Fixpunkt im Diagramm. Sobald sich der Cursor in das Handsymbol ändert, wird der entsprechende Fixpunkt in der Tabelle grau hinterlegt. Die erforderliche „Nähe" der Maus zum Fixpunkt kann in → *Mauspräzision* im Fenster „Einstellungen" bestimmt werden.

Fixpunkt verschieben

Stellen Sie den Mauszeiger auf einen Fixpunkt bis das Handsymbol erscheint. Klicken und halten Sie die linke Maustaste. Nun können Sie den Punkt auf die gewünschte neue Position ziehen. Die Positionen werden automatisch in der Tabelle aktualisiert.

Wenn → ☑ *Online-Berechnung* aktiviert ist, wird die Kurve simultan nachgezeichnet und die Registerkarte *Kurveninfo* aktualisiert, sobald ein Punkt bewegt wird.

Wenn → ☑ *Ausrichten am Raster* aktiviert ist, wird der Punkt immer an der nächsten Rasterlinie einschnappen und nicht exakt der Maus folgen.

Mit *Rückgängig* können Sie die letzte Änderung rückgängig machen.

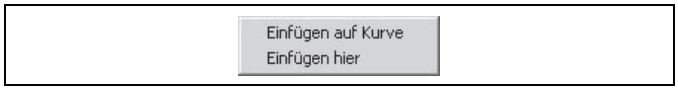
Fixpunkt hinzufügen

Doppelklicken Sie mit der linken Maustaste auf die Position wo ein neuer Fixpunkt eingefügt werden sollt. Mit *Rückgängig* kann der Punkt wieder gelöscht werden.

Im Abschnitt „Fixpunkte" finden Sie weitere Informationen, z.B. Fixpunkte bearbeiten und Fixpunkte Typ.

CAM Kontextmenüs einsetzen

Klicken Sie mit der rechten Maustaste in einen leeren Bereich des Diagramms, dann öffnet sich das Kontextmenü zum Festlegen der Position dieses Fixpunktes:



Einfügen auf Kurve

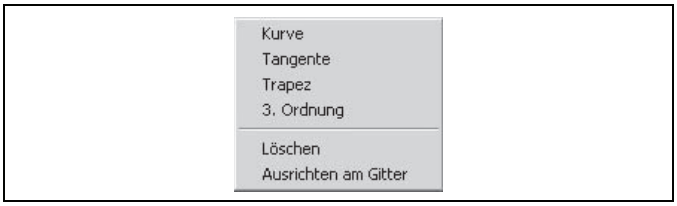
Ein neuer Fixpunkt wird an der nächstmöglichen Masterposition von der Stelle eingefügt, an der mit der Maustaste geklickt hatten.

Einfügen hier

Ein neuen Fixpunkt wird genau an der Stelle eingefügt, an der mit der rechten Maustaste geklickt hatten. Es ist das gleiche, als würden Sie mit der linken Maustaste doppelklicken. Das Diagramm wird aktualisiert.

Fixpunkt Typ ändern

Klicken mit der rechten Maustaste auf einen Fixpunkt öffnet das Kontextmenü zum Ändern dieses Fixpunktes:



Wählen Sie den Typ aus, in den dieser Fixpunkt geändert werden soll: *Kurve, Tangente, Trapez,* oder *3. Ordnung.*

Löschen

→ *Löschen* entfernt den ausgewählten Fixpunkt.

Ausrichten am Raster

Richtet diesen Fixpunkt an der nächstmöglichen Rasterlinie aus.

□ CAM Registerkarten

Die CAM-Editor Registerkarten ermöglichen es, verschiedene Parameter bezogen auf die Kurvenprofile zu ändern. Dazu werden die Registerkarten *Kurvendaten* und *Kurveninfo* verwendet. Mit den anderen Registerkarten können zusätzlich alle Achsenparameter geändert werden. Diese Achsenparameter werden gemeinsam mit den Kurvenprofilen in die Steuerung heruntergeladen, wenn Sie die Konfigurationsdatei downloaden.

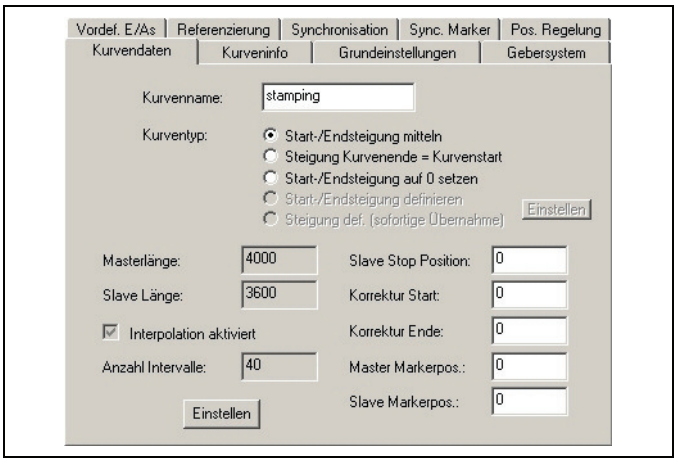
Die Achsenparameter *Synchronisation* sind besonders wichtig, wenn mit Kurvenprofilen gearbeitet wird. Diese werden im Folgenden beschrieben.

In der „Parameter Referenz" finden Sie eine detaillierte Beschreibung aller Achsenparameter. Oder Sie drücken [F1], wenn der Textcursor im Parameterfeld steht (z.B. durch Klicken ins das Feld).

Registerkarte Kurvendaten

Kurvenname

Wenn Sie mehrere Kurven editieren, können Sie hier aussagefähige → *Kurvennamen* eingeben. Diese können bis zu 16 Zeichen lang sein und werden im „Array"-Feld in der Symbolleiste des CAM-Editors dargestellt. Diese Namen dienen nur Ihrer Information, die Steuerung benutzt sie nicht.



Kurventyp

Typischerweise sind Kurvenprofile zyklisch; wenn die Steuerung das Ende einer Kurve erreicht, startet sie wieder von vorn. Daher ist es äußerst wichtig, dass die Steigung (d.h. die Geschwindigkeit) des Kurvenprofils zum Start und Ende der Kurve passt. Dies erlaubt dem Slave-Motor sanft vom Ende der Kurve zurück zum Start der Kurve zu laufen, so wie die Masterzyklen im Kurvenprofil. Eine Unstetigkeit der Steigung würde einen harten Geschwindigkeitssprung verursachen, der den Motor oder die angeschlossene Anlage beschädigen könnte.

Um Drehzahlsprünge bei mehrmaligem Kurvendurchlauf zu verhindern, können Sie zwischen verschiedenen Kurventypen wählen:

- *Start-/Endsteigung mitteln* – Die tatsächliche Slave-Geschwindigkeit ist nicht wichtig, weder am Beginn noch am Ende des Profils. Der CAM-Editor wählt eine mittlere Steigung um lediglich eine Laufruhe zu gewährleisten.
- *Steigung Kurvenende = Kurvenstart* – Die tatsächliche Slave-Geschwindigkeit am Kurvenstart ist wichtig und sollte nicht geändert werden. Der CAM-Editor gleicht nur die Steigung am Ende an, so dass sie zur Steigung am Start passt. Die Steigung am Start wird nicht geändert.
- *Start-/Endsteigung auf 0 setzen* – Der Slave-Motor muss beim Lastwechsel des Masters gestoppt werden. Der CAM-Editor setzt die Steigung am Start und Ende der Kurve auf 0.
- *Start-/Endsteigung definieren* – Die tatsächliche Slave-Geschwindigkeit ist bei beiden wichtig, sowohl beim Start als auch am Ende des Profils. Der Anwender muss explizit die Start- und Endgeschwindigkeiten → *Einstellen*. Der CAM-Editor wird den Anwender zwar warnen, wenn die Start- und Endsteigungen nicht passen, aber es liegt in seiner Verantwortung, zu vermeiden, dass Unstetigkeiten zu Problemen führen könnten.
- *Steigung definieren (sofortige Übernahme)* – Der Anwender muss explizit die Start- und Endgeschwindigkeiten → *Einstellen*. Die Startsteigung wird jedoch durch die tatsächliche aktuelle Geschwindigkeit ersetzt, sobald die Kurve in der Steuerung aktiviert wird. Dieser Kurventyp ist dazu vorgesehen, eine vorhandene Kurve durch eine andere Kurve zu ersetzen, bevor die erste Kurve ihren Zyklus beendet hat.

ACHTUNG!:
Nicht alle Steuerungsversionen unterstützen alle Kurventypen.

Masterlänge und Slavelänge

Die Masterlänge ist der Abstand, den der Master-Drehgeber fahren muss, um ein Intervall zu beenden; es ist die horizontale Länge eines Kurvenprofils. Wenn der Master ein Intervall beendet, dann beendet auch der Slave-Motor ein Intervall, indem er dem Kurvenprofil vom Startpunkt bis zum Endpunkt folgt. Die Slave-Länge ist die Differenz zwischen der Slave-Position vom Anfang bis zum Ende der Kurve.

ACHTUNG!:
Das ist aber nicht das Gleiche wie die vertikale Höhe der Kurve. Falls der Slave zu seiner Start-Position zurückgekehrt ist, wenn der Master ein Intervall beendet hat, ist die Slave-Länge 0, auch wenn sich der Slave während des Intervalls bewegt hat.

Die Master- und die Slavelänge 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- und Slave-Intervall-Länge direkt. → *Einstellen* in der Registerkarte *Kurvendaten* öffnet ein Dialogfenster, in dem sowohl die *Masterlänge* als auch die Anzahl der Interpolations-Intervalle (siehe unten) gleichzeitig geändert werden kann. Die Slave-Länge kann in diesem Dialog nicht geändert werden.

Wählen Sie nicht zu kleine Längen für das Masterintervall. 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-Markerposition liegt.

Falls es notwendig ist, den letzten Fixpunkt außerhalb dieser Grenzen zu schieben, müssen zuerst die begrenzenden Punkte bzw. Positionen geändert werden. Danach kann der letzte Fixpunkt verschoben werden. Wenn zum Beispiel der letzte Fixpunkt durch die Master-Markerposition blockiert ist, bewegen Sie die Master-Markerposition zuerst und dann den letzten Fixpunkt.

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.

Ein horizontaler grauer Balken zeigt diese Position im Kurvenprofil. Aktivieren Sie dazu → ☑ *Slave-Stop-Position*.

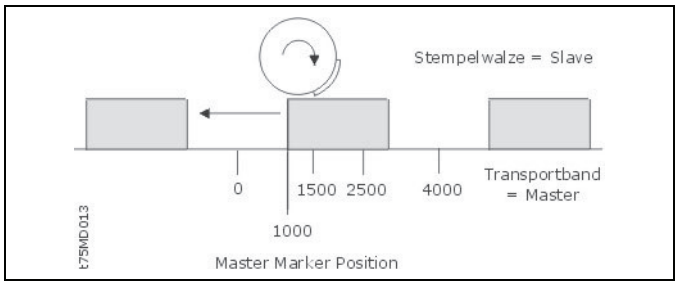
Korrektur Start / Ende

Geben Sie die Masterpositionen 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 als blaugrauer Balken entlang der x-Achse dargestellt. Aktivieren Sie dazu → ☑ *Korrektur*.

Master-Marker und Slave-Marker Position

Tragen Sie die Masterposition (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. Dies ermöglicht die Festlegung des Korrekturbereiches.

Wenn → ☑ *Master Marker* aktiviert ist, zeigt eine vertikale grüne Linie die Master-Markerposition im Kurvenprofil-Diagramm. Wenn → ☑ *Slave Marker* aktiviert ist, zeigt eine horizontale grüne Linie die Slave-Markerposition.

Interpolation und Anzahl Intervalle

Steuerungen mit einer älteren Firmware als MCO 5.00 benötigen „Interpolationspunkte" beim Ausführen von Kurvenprofilen. Für neuere Steuerung ist dies nicht mehr notwendig und dieser Abschnitt kann übersprungen werden. Die Firmware-Version der Steuerung wird im APOSS Kommunikationsfenster gezeigt, wenn die Steuerung zum ersten Mal mit der APOSS-IDE verbunden wird.

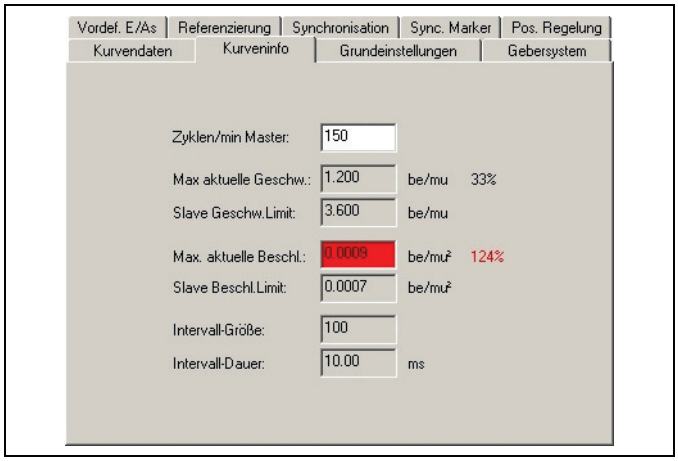
Wenn also das Kurvenprofil für eine ältere Steuerung vorgesehen ist, sollte die Checkbox → ☑ *Interpolation aktiviert* werden, entweder wenn ein neues Kurvenprofil hinzugefügt wird oder mittels → *Einstellen*. Die Checkbox kann nicht direkt in der Registerkarte Kurvendaten aktiviert werden.

Ältere Steuerungen erfordern wesentlich mehr Fixpunkte als neuere Steuerungen. Wenn Interpolation aktiviert ist, fügt der CAM-Editor automatisch Interpolationspunkte (d.h. „virtuelle" Fixpunkte) in ein gleichmäßig aufgeteiltes „Interpolationsraster" hinzu. Die Größe des Interpolationsintervalls (d.h. der Abstand zwischen den Interpolationspunkten) bestimmt die Genauigkeit, mit der die Steuerung dem Kurvenprofil folgen kann und muss vom Anwender gemäß der Genauigkeitsanforderung der Anwendung gewählt werden. Kleinere Interpolationsintervalle bringen zwar eine höhere Genauigkeit, erzeugen aber mehr Interpolationspunkte und größere Arrays in der Steuerung. Sehr kleine Intervalle können als Folge einer hohen Anzahl von kurzen Segmenten zu Performance-Problemen der Steuerung führen. Die *Intervall-Dauer* (siehe Registerkarte *Kurveninfo*) sollte nicht kleiner als 20-30 ms sein.

Um Interpolationspunkte mit Nachkommastellen zu vermeiden, sollte die Masterlänge ein ganzzahliges Vielfaches des Interpolationsintervalls sein. Um dies sicherzustellen, zieht der CAM-Editor immer den letzten Fixpunkt auf das Interpolationsraster. Wenn → *Einstellen* benutzt wurde, dann stellt der CAM-Editor auch sicher, dass die neue *Masterlänge* ein Vielfaches des Interpolationsintervalls ist. Wenn die *Masterlänge* mit → *Einstellen* gewechselt wurde, wird auch die Anzahl der Interpolationspunkte automatisch geändert, so dass das Interpolationsintervall (d.h. die Genauigkeit der Kurve) unverändert bleibt. Für ein Intervall wird eine *Masterlänge* von mindestens 1000 empfohlen, um eine ausreichende Auflösung des Interpolationsrasters zu erhalten.

Obwohl nicht unbedingt notwendig, wird empfohlen, auch die Benutzer-Fixpunkte exakt auf das Interpolationsraster zu setzen. Das Schaltsymbol *Fixpunkte am Raster ausrichten* und die Checkbox-Funktion ☑ *Ausrichten am Raster* können dazu benutzt werden. Wenn Fixpunkte nicht auf dem Interpolationsraster gesetzt sind, wird das tatsächliche Kurvenprofil, dem die Steuerung folgt, leicht außerhalb des Rasters liegen, aber nicht exakt durch die Fixpunkte gehen.

Registerkarte Kurveninfo



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.

Maximale aktuelle Geschwindigkeit

Der Wert *Max. aktuelle Geschw.* gibt die maximale Geschwindigkeit des Slaves in diesem Kurvenprofil an, und zwar in Benutzereinheiten pro Master Unit [BE/MU] und in Prozent des *Slave Geschwindigkeits-Limits.* Falls die Geschwindigkeit das Limit erreicht, wird das Feld rot markiert.

Zur grafischen Darstellung der Geschwindigkeitskurve aktivieren Sie → ☑ *Geschwindigkeit*. Die Geschwindigkeitskurve des Slaves ist blau und die Achse auf der rechten Diagrammseite (solange nicht auch Beschleunigung und/oder Ruckbegrenzung dargestellt werden). Beachten Sie, es kann notwendig sein, dass Sie für die Geschwindigkeitskurve den *Zoom zurücksetzen*, damit sie mit einer innerhalb der vorhandenen Diagrammgrenzen geeigneten Skala dargestellt wird.

Slave Geschwindigkeits-Limit

Das *Slave Geschw.-Limit* ist die maximale Geschwindigkeit, die der Slave erreichen kann – bestimmt durch den aktuellen Wert von *Zyklen/min Master* und der maximalen Nenndrehzahl des Slave-Motors. Wenn *Zyklen/min Master* zunimmt, verringert sich das *Slave-Geschwindigkeits-Limit*, so dass die tatsächliche Geschwindigkeit des Slaves nicht die maximale Nenndrehzahl des Slave-Motors erreicht. Wenn die Geschwindigkeit, die durch diese Kurve gefordert ist (*Max. aktuelle Geschwindigkeit*) das Limit erreicht, kann der Slave dem Master nicht mehr folgen.

Wenn → ☑ *Geschw.-Limit* aktiviert ist, wird das Geschwindigkeits-Limit als horizontale dunkelblaue Linie dargestellt. Beachten Sie, dass sowohl positive als auch negative Geschwindigkeits-Limits dargestellt werden.

Maximale aktuelle Beschleunigung

Der Wert *Max. aktuelle Beschleunigung* gibt die maximale Beschleunigung des Slaves in diesem Kurvenprofil an, und zwar Benutzereinheiten pro Master-Units zum Quadrat (BE/MU²) und in Prozent des *Slave Beschleunigung-Limits*. Falls die Beschleunigung das Limit erreicht, wird das Feld rot markiert.

Zur grafischen Darstellung der Beschleunigungskurve aktivieren Sie → ☑ *Beschleunigung*. Die Kurve des Slaves ist magentarot und die ist Achse auf der rechten Seite des Diagramms (solange nicht auch Beschleunigung und/oder Ruckbegrenzung dargestellt werden). Beachten Sie, dass es notwendig sein kann, dass Sie für die Beschleunigungskurve den *Zoom zurücksetzen*, damit diese mit einer innerhalb der vorhandenen Diagrammgrenzen geeigneten Skala dargestellt wird.

Slave Beschleunigungs-Limit

Das *Slave Beschleunigungs-Limit* ist die maximale Beschleunigung, die der Slave erreichen kann – bestimmt durch den aktuellen Wert von *Zyklen/min Master* und der maximalen Nennbeschleunigung des Slave-Motors. Wenn *Zyklen/min Master* zunimmt, verringert sich das *Slave-Beschleunigungs-Limit*, so dass die tatsächliche Beschleunigung des Slaves nicht die maximale Nennbeschleunigung des Slave-Motors erreicht. Wenn die Beschleunigung, die durch diese Kurve gefordert ist (*Max. aktuelle Beschleunigung*) das Limit erreicht, kann der Slave dem Master nicht mehr folgen.

Wenn → ☑ *Beschl.-Limit* aktiviert ist, wird das Beschleunigungs-Limit als horizontale dunkle magentarote Linie im Kurvenprofil-Diagramm dargestellt. Beachten Sie, dass sowohl positive als auch negative Beschleunigungs-Limits dargestellt werden.

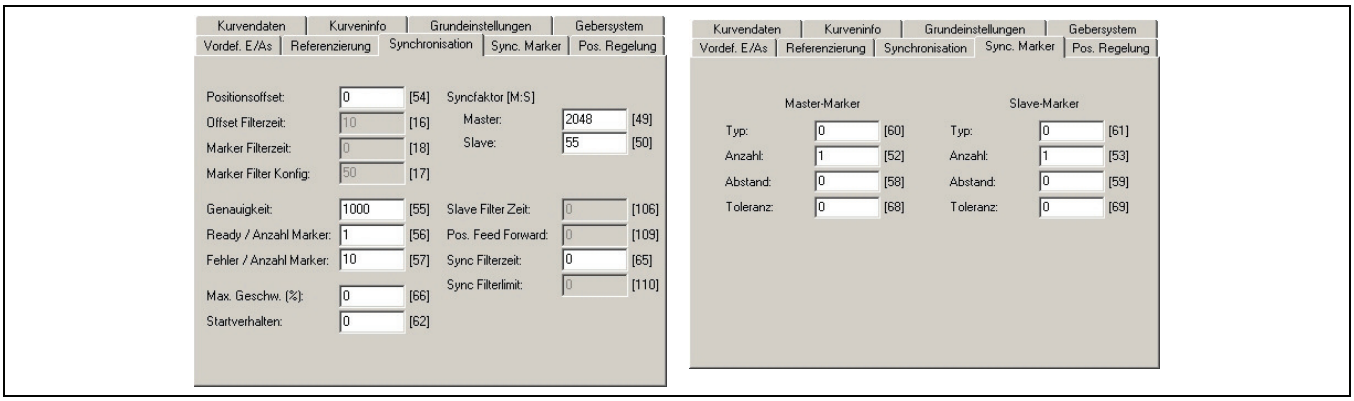
Intervall-Größe und Intervall-Dauer (ms)

Diese Werte sind nur wichtig, wenn Interpolation aktiviert ist. Die Intervall-Größe ist die Länge eines einzelnen Interpolationsintervalls in Master-Units. Sie wird abgeleitet von der Anzahl der Intervalle und der Länge des Masterzyklus.

Die Intervall-Dauer (in Millisekunden) wird abgeleitet von der Intervall-Größe und der Anzahl der Masterzyklen pro Minute. Sie sollte nicht kleiner als 30 ms sein. 30 bis 100 ms sind geeignete Werte. Verändern Sie also in *Kurvendaten* die → *Anzahl Intervalle*, damit Sie eine vernünftige Größe erhalten.

Registerkarte Synchronisation und Sync Marker

Die Achsparameter *Synchronisation* sind besonders wichtig, wenn mit Kurvenprofilen gearbeitet wird, da sie die Synchronisation des Slaves zum Master beeinflussen. Die relevanten Parameter sind nachfolgend erläutert:



Syncfaktor Master und Slave

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

Marker Abstand

Tragen Sie hier den Abstand des Sensors zum Bearbeitungspunkt ein: Parameter 33-17 *Markerabstand Master* und 33-18 *Markerabstand Slave*. Aus der Master-Markerposition und dem Markerabstand wird die Position der Kurve errechnet, bei der der Marker erkannt wird.

Wenn → ☑ *Master Marker* bzw. *Slave Marker* aktiviert ist, werden diese Positionen im Kurvenprofil als grüne Linie dargestellt.

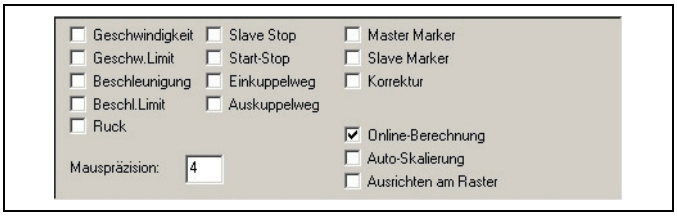
Toleranz

Toleranzfenster für das Auftreten der Master-Marker Par. 33-21 bzw. der Slave-Marker Par. 33-22.

Das Toleranzfenster wird im Kurvenprofil als grüner Bereich dargestellt. Aktivieren Sie dazu → ☑ *Master Marker* bzw. *Slave Marker*.

☐ Einstellungen Kurvenprofil

Dieser Bereich des CAM-Editors enthält Checkboxen zum Ein- und Ausschalten der Darstellung der verschiedenen Eigenschaften des Kurvenprofils.



Geschwindigkeit

Zeigt die Geschwindigkeitskurve entsprechend dem Kurvenprofil als blaue Linie. Die dazu gehörende Skala ist auf der rechten Seite des Diagramms (so lange dies die einzige Skala ist, die auf der rechten Seite des Diagramms dargestellt wird).

Geschw.-Limit

Zeigt die *Geschwindigkeits-Limits* als zwei horizontale blaue Linien.

Beschleunigung

Zeigt die Beschleunigungskurve entsprechend dem Kurvenprofil (magentarote Linie). Die dazu gehörende Skala ist auf der rechten Seite des Diagramms (so lange dies die einzige Skala ist, die auf der rechten Seite des Diagramms dargestellt wird).

Beschl.- Limit

Zeigt das *Bechleunigungs-Limit* als zwei horizontale dunkle magentarote Linien.

Ruck

Zeigt den Ruck entsprechend dem Kurvenprofil als hellblaue Linie. „Ruck" ist der Impuls des Antriebs bei Änderungen der Beschleunigung. Die dazu gehörende Skala ist auf der rechten Seite des Diagramms (so lange dies die einzige Skala ist, die auf der rechten Seite des Diagramms dargestellt wird).

Slave Stop

Zeigt die Slave-Stop-Position, die in der Registerkarte *Kurvendaten* definiert ist. Sie wird als graue horizontale Linie dargestellt.

Start-Stop

Zeigt die Positionen der Start-Stop-Punkte als gelbe oder rote Flags im Kurvenprofil.

Einkuppelweg

Zeigt den Einkuppelweg für das aktuell ausgewählte Start-Stop-Punktepaar. Dieser wird als dunkelblaue Kurve ausgehend von der Slave-Stop-Position zum Kurvenprofil hin zwischen dem Start-Stop-Punktepaar dargestellt.

Beachten Sie, dass ein Start-Stopp-Punktepaar in der Tabelle der Start-Stop-Punkte ausgewählt sein muss, bevor etwas dargestellt werden kann.

Auskuppelweg

Zeigt den Auskuppelweg für das aktuell ausgewählte Start-Stop-Punktepaar als dunkelblaue Kurve.

Master Marker

Zeigt die Master-Markerposition (abgeleitet von den Daten aus den Registerkarten Kurvendaten, Synchronisation und Sync Marker) als vertikale grüne Linie. Falls eine Toleranz angeben ist, wird diese als hellgrünes Rechteck unter der Master-Markerposition dargestellt.

Slave Marker

Zeigt die Slave-Markerposition (abgeleitet von den Daten aus den Registerkarten Kurvendaten, Synchronisation und Sync Marker) als horizontale grüne Linie. Falls eine Toleranz angeben ist, wird diese als hellgrünes Rechteck unter der Slave-Markerposition dargestellt.

Korrektur

Zeigt die Positionen, wo die Markerkorrektur voraussichtlich beginnt und endet (definiert in der Registerkarte *Kurvendaten*). Dies wird als blaugrauer Balken entlang der x-Achse des Kurvenprofils dargestellt.

Online-Berechnung

Wenn Online-Berechnung aktiviert ist, wird das Kurvenprofil kontinuierlich berechnet und neu dargestellt, und die Fixpunkte werden im Kurvenprofil nachgezogen. Dieses Neuzeichnen der Kurve kann hohe Prozessleistung erfordern. Falls sich das Neuzeichnen unberechenbar verhält oder Störungen auf Ihrem Computer verursacht, sollten Sie Online-Berechnung ausschalten; die Kurve wird dann neu berechnet und dargestellt, sobald Sie die Maustaste loslassen.

Auto-Skalierung

Wenn Auto-Skalierung aktiviert ist wird das Diagramm automatisch so skaliert, dass immer die gesamte Kurve zu sehen ist.

Wenn das Diagramm manuell gezoomt oder gescrollt wurde, wird die Funktion automatisch deaktiviert, damit der so auswählte Bereich sichtbar bleibt.

Ausrichten am Raster

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

ACHTUNG!:
Der letzte Fixpunkt rastet immer auf dem Interpolationsraster ein, unbeachtet ob die Funktion aktiviert ist oder nicht. So kann eine exakte Beziehung zwischen der Länge des Masterintervalls und der Anzahl der Interpolationspunkte sichergestellt werden.

Mauspräzision

Wenn die rechte Maustaste gedrückt wird, während die Mauszeiger auf einen Fixpunkt zeigt, wird das Fixpunkte-Kontextmenü, andernfalls das normale Kontextmenü dargestellt. Dieser Wert legt fest, wie nah der Mauscursor an einem Fixpunkt sein muss, bevor der CAM-Editor erkennt, dass der Mauszeiger auf einen Fixpunkt zeigt. Dann wird auch der Pfeil in das Handsymbol geändert. Gleichzeitig wird dieser Fixpunkt in der Fixpunkte-Tabelle gekennzeichnet, so dass der Anwender auch hier den Punkt identifizieren kann.

□ Fixpunkte

Fixpunkte werden benutzt, um die grundlegende Form der Kurve über die Länge eines Masterintervalls. Der *CAM-Editor* errechnet daraus eine mathematische Spline-Funktion, die durch diese Fixpunkte führt. Sie sollten nicht mehr Fixpunkte definieren als notwendig sind, um das gewünschte Kurvenprofil angemessen zu erzeugen. Mehr Punkte als notwendig verbrauchen nur unnötig Leistung der Steuerung ohne entsprechend Nutzen.

Änderungen in der Fixpunkte in der Fixpunkte-Tabelle können mit ⮌ *Rückgängig* gemacht werden.

Fixpunkte bearbeiten

Sie können Fixpunkte in der Tabelle eingeben, ändern und löschen, oder mit der Maus direkt im Kurvenprofil mit der Maus setzen und verschieben oder mittels der CAM Kontextmenüs, das mit der rechten Maustaste geöffnet wird, bearbeiten.

Fixpunkte einfügen in der Tabelle

Klicken Sie in der Fixpunkte 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 zur Kurve hinzugefügt werden.

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

Wenn ein neuer Fixpunkt nach dem letzten Fixpunkt eingefügt wurde, ändert dies die Länge des Masterzyklus. Wenn dies geschieht, wird auch die Anzahl der Interpolationspunkte automatisch geändert, so dass das Interpolationsintervall (d.h. der Abstand zwischen den Interpolationspunkten) unverändert bleibt. Der letzte Fixpunkt rastet immer auf dem Interpolationsraster ein, um Interpolationspunkte mit Nachkommastellen zu vermeiden.

Fixpunkte löschen

Klicken Sie auf den Fixpunkt in der Tabelle Fixpunkte und drücken Sie [Entf].

Fixpunkte ändern

Um einen Fixpunkt zu ändern, doppelklicken Sie auf den Fixpunkt in der Tabelle Fixpunkte. Zum Ändern der Position klicken Sie die Spalte Master oder Slave, zum Ändern des Punktetyps in die Spalte Typ.

Beachten Sie, dass ein Fixpunkt nicht außerhalb der beiden benachbarten Fixpunkte bewegt werden kann. Nur der letzte Fixpunkt kann so weit wie gewünscht nach rechts bewegt werden. In diesem Fall wird die Länge des Masterzyklus automatisch geändert. Die Anzahl der Interpolationspunkte wird auch automatisch geändert, so dass das Interpolationsintervall (d.h. der Abstand zwischen den Interpolationspunkten) unverändert bleibt. Der letzte Fixpunkt rastet immer auf dem Interpolations-Raster ein, um Interpolationspunkte mit Nachkommastellen zu vermeiden.

Fixpunkte Typ

Der Fixpunkt Typ bestimmt die Form des einzelnen Kurvenprofilsegments, das dem Fixpunkt folgt. Segmente können entweder gerade (d.h. eine Tangente) oder gebogen sein. Der *CAM-Editor* wird immer versuchen, einen fließenden Übergang von einem Segment zum nächsten zu erzeugen. Daher ist mindestens ein Kurvensegment zwischen zwei benachbarten Tangentensegmenten notwendig.

Im Kurvenprofil-Diagramm werden Tangentenpunkte als kleine grüne Quadrate, alle anderen Punktetypen als kleine rote Quadrate dargestellt.

Folgende Fixpunkt Typen werden unterstützt:

Kurve (C)

Das folgende Segment ist gebogen. Wenn es zwei benachbarte Tangentensegmente verbindet, wird es als einzelne Kurve 5. Ordnung berechnet. Dies bietet den bestmöglichen fließenden Übergang zwischen den Segmenten, da sowohl die Geschwindigkeit als auch die Beschleunigung exakt auf beiden Enden des Segments übereinstimmen., Es kann jedoch höhere Geschwindigkeiten und Beschleunigungen als andere Kurventypen erfordern. Wenn es mehr als ein folgendes Kurvensegment gibt, wird die Serie von Kurvensegmenten gemeinsam berechnet, wobei ein einzelner kubischer Spline benutzt wird.

Tangente (T)

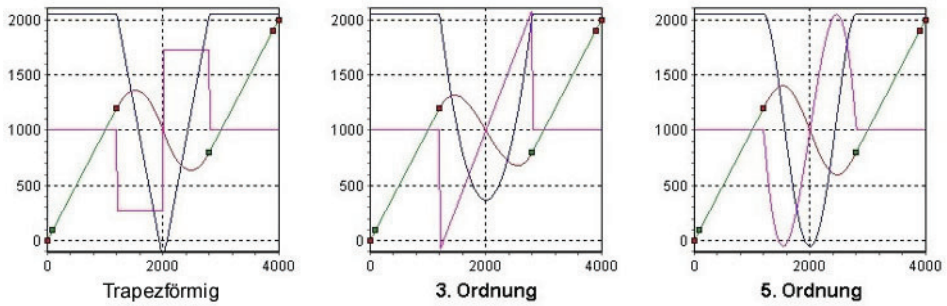
Das folgende Segment ist gerade. Die Geschwindigkeit ist konstant und die Beschleunigung ist 0.

Trapez (TZ)

Zur Berechnung des folgenden Segments wird ein trapezförmiges Geschwindigkeitsprofil benutzt (d.h. zwei Segmente einer konstanten Beschleunigung). Dieser Kurventyp kann nur zwischen zwei benachbarten Tangentensegmenten verwendet werden.

3. Ordnung (K3)

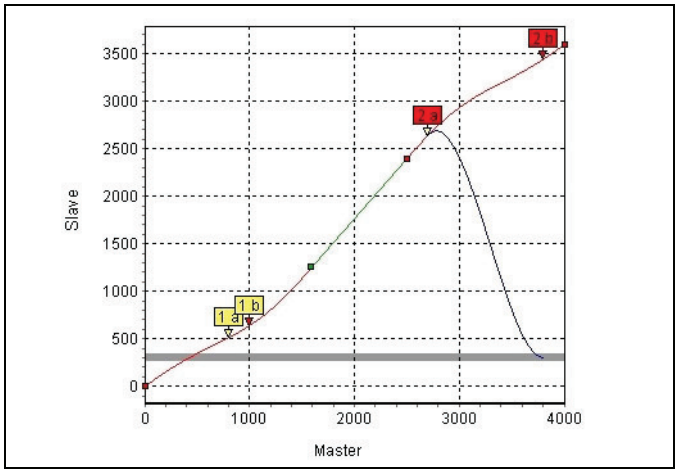
Zur Berechnung des folgenden Segments wird eine einzelne Kurve 3. Ordnung benutzt. Dieser Kurventyp kann nur zwischen zwei benachbarten Tangentensegmenten verwendet werden.



In der trapezförmigen Kurve kann man die zwei Segmente der konstanten Beschleunigung sehen (magenta-rote Linie). Dies erzeugt in der Mitte der Kurve einen abrupten Wechsel der Geschwindigkeit (blaue Linie). In der Kurve 3. Ordnung wechselt die Geschwindigkeit sanfter, aber es gibt immer noch einen abrupten Wechsel der Beschleunigung. In der Kurve 5. Ordnung sind die Wechsel sowohl der Geschwindigkeit wie auch der Beschleunig deutlich sanfter.

☐ Start-Stop-Punkte

Mit den Start-Stop-Punkten werden Punktepaare für das Ein- und Auskuppeln des Slaves bei der Synchronisation definiert. Ein Punktepaar benötigen Sie um zu bestimmen, an welcher Masterposition 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.



Wenn → ☑ *Start-Stop* aktiviert ist, werden die Start-Stop-Punkte mit gelben Flags dargestellt. Der Start-Punkt wird mit „a" gekennzeichnet und hat einen gelben Pfeil; der Stop-Punkt mit „b" und rotem Pfeil. Wenn Sie ein Punktepaar in der Tabelle auswählen, wird es rot markiert.

Zum Darstellen der Ein- und Auskuppelkurven aktivieren Sie → ☑ *Einkuppelweg* und/oder → ☑ *Auskuppelweg* und klicken dann auf das gewünschte Start-Stopp-Punktepaar in der Tabelle *Start-Stop-Punkte.* Zur gleichen Zeit kann immer nur ein Paar der Einkuppel- und Auskuppelkurven dargestellt werden.

Zum Darstellen der Slave-Stop-Position aktivieren Sie → ☑ *Slave Stop*. Dies ist die Position, an der das Einkuppeln beginnt und das Auskuppeln endet. Sie wird als grauer horizontaler Balken dargestellt.

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!

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.

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-Punkte* = *Stop-Punkte*, wird der Slave mit der eingestellten Maximalgeschwindigkeit – also ohne Kurve – eingekuppelt, sobald der Master diesen Punkt erreicht hat.

Wenn *Start-Punkt* > *Stop-Punkt,* werden die Wege beim Einkuppeln und Auskuppeln „auf das nächste Masterintervall" überlaufen.

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

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

Start-Stop-Punkte bearbeiten

Start-Stop-Punkte werden in der Tabelle Start-Stop-Punkte bearbeitet; die Maus kann hierfür nicht benutzt werden.

Start-Stop-Punkte hinzufügen

Klicken Sie auf → *Einfügen* in der Tabelle Start-Stop-Punkte. Geben Sie im folgenden Dialogfeld die Werte für die Start-Stop-Punkte paarweise ein. Dabei können Sie mehrere Paare gleichzeitig eingeben. Anders als bei den Fixpunkten (diese müssen der Reihe nach sortiert sein) können Start-Stop-Punkte in beliebiger Reihenfolge eingegeben werden.

Start-Stop-Punkte löschen

Klicken Sie auf das Punktepaar in der Tabelle Start-Stop-Punkte drücken Sie die [Entf]**-**Taste.

Start-Stop-Punkte ändern

Zum Ändern von Start-Stopp-Punkten doppelklicken Sie in der Tabelle auf das entsprechende Punktepaar und geben im folgenden Dialogfeld die Änderungen ein.

MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

□ Array-Editor

Der *Array-Editor* bietet ein listenorientiertes Interface, mit dem man alle Arrays und Parameter der Steuerung betrachten und bearbeiten kann. Dies beinhaltet Benutzerparameter, globale und Achsenparameter sowie die Status-Variablen der Achse und des Systems.

Zusätzlich kann eine *„Array-Definitionsdatei"* („.zba") erzeugt werden, die als Vorlage für die Festlegung von Element-Namen, Minimum und Maximum Werten, Default-Werten usw. dient. Durch den Einsatz dieser Array-Definitionsdatei ist sehr benutzerfreundliches Bearbeiten von Arrays möglich. Der Array Editor verhält sich ein wenig anders, je nachdem ob eine Array-Definitionsdatei benutzt wird oder nicht (siehe Benutzer- und Expertenmodus MIT und OHNE Array-Definitionsdatei).

Der Array Editor unterstützt zwei Modi: Den Expertenmodus und den Benutzermodus. Der Expertenmodus erlaubt den vollständigen Zugang zu allen Arrays und Parametern sowie zu allen Bearbeitungsfunktionen. Mit dem Expertenmodus werden die Array-Definitionsdateien erzeugt, die danach im Benutzermodus verwendet werden. Im Benutzermodus ist der Zugang zu den Arrays und Parametern auf die in der Array-Definitionsdatei vordefinierten Elemente beschränkt. Eine Array-Definitionsdatei kann zwar benutzt, aber nicht bearbeitet werden. Der Hauptzweck des Benutzermodus ist, ein „Konfigurations-Tool" für das Personal vor Ort bereitzustellen. Die Array-Definitionsdatei wird dann dazu benutzt, den Zugang auf jene vordefinierten Einträge zu begrenzen, die die Konfiguration erfordern.

In *Einstellungen* → *Optionen* legen Sie fest, welche Daten in den Array-Editor geladen werden:
Wenn ☑ *Array Editor Parameter-Unterstützung* aktiviert ist, werden alle Arrays und Parameter (d.h. globale, Achsen- und Benutzerparameter, usw.) in den Array Editor geladen.
Wenn ☐ *Array Editor Parameter-Unterstützung* nicht aktiviert ist, werden nur Arrays und Benutzerparameter geladen. Andere Parameter sind nicht zugänglich, selbst wenn sie in einer Array-Definitionsdatei definiert sind.

□ Array-Editor starten

Der Array-Editor kann entweder innerhalb der APOSS IDE oder als eine stand-alone Anwendung gestartet werden:

Wenn er innerhalb der APOSS IDE gestartet wird, dann startet der Array Editor immer im Expertenmodus und erlaubt den vollständigen Zugang zu allen Arrays und Parametern. Das Array-Editor-Fenster verhält sich ähnlich wie die anderen APOSS-Fenster (d.h. Editierfenster, CAM-Editor, Oszilloskop, etc.).

Wenn der Array-Editor als stand-alone Anwendung gestartet wird, kann er entweder im Expertenmodus oder im Benutzermodus gestartet werden.

APOSS kann immer nur einmal gestartet werden. Falls also APOSS bereits läuft, erfolgt eine Meldung und der Array-Editor wird geschlossen.

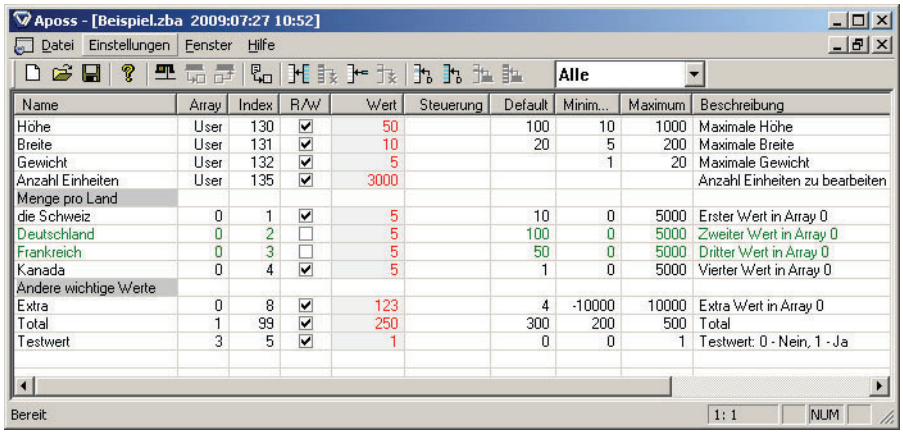
In APOSS starten

Der Array-Editor kann innerhalb der APOSS-IDE mit folgenden Methoden gestartet werden:

1. Mit *Tools* → *Array Editor* oder dem Array-Editor.
2. Mit *Datei* → *Neu* oder dem 🗎 Symbol und dann eine → *Array-Datei* (*.zba) anlegen.
3. Mit *Datei* → *Öffnen* und dann eine vorhandene Array-Datei (.zba) auswählen.
4. Oder mit irgendeiner anderen der Windows Datei-Öffnen-Methoden wie Doppelklick auf eine .zba-Datei oder Ziehen einer .zba-Datei in ein geöffnetes APOSS-Fenster.

In all diesen Fällen werden die Arrays und Parameter automatisch von der Steuerung geladen wenn eine Steuerung bereits mit dem gerade aktiven Fenster verbunden ist.

Die Titelleiste zeigt die ID und den Namen der Steuerung (fall eine verbunden ist) und das Datum, wann die Daten von der Steuerung gelesen wurden.

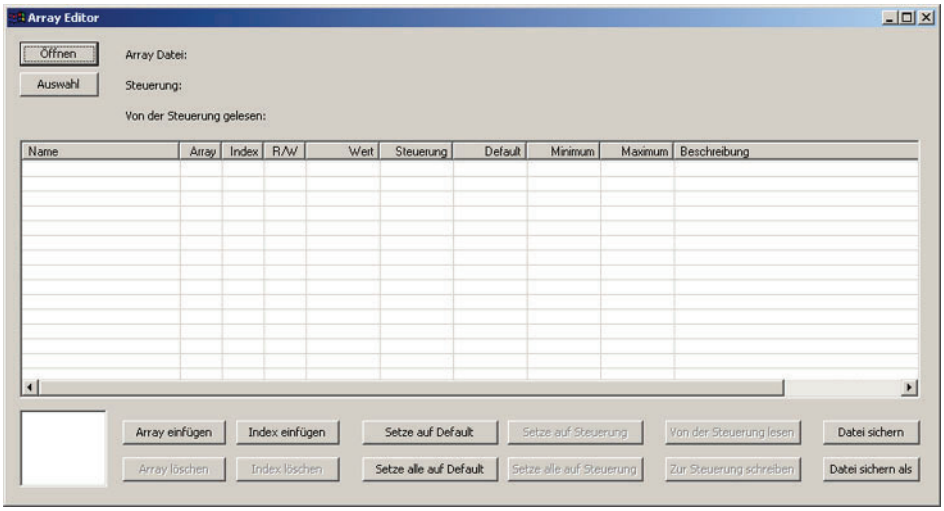


Stand-alone starten

Als stand-alone Anwendung wird der Array-Editor durch Hinzufügen eines Arguments zur Windows-Befehlszeile gestartet:

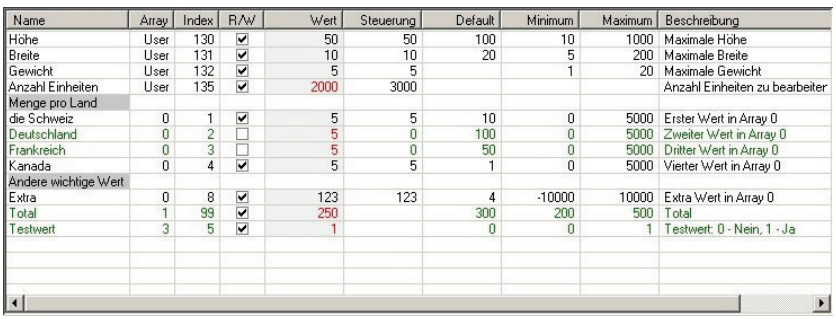
„/a" startet im Benutzermodus und
„/ae" startet im Expertenmodus.

Stand-alone startet der Array Editor immer ohne eine Steuerung zu verbinden und ohne eine Array-Definitionsdatei zu laden. Beides muss der Anwender explizit selbst vornehmen.



□ Array-Editor-Liste

Arrays und Parameter der Steuerung werden in einer Liste dargestellt. Beispiel:



Eine oder mehrere Reihen in der Liste können mit den üblichen Windows-Methoden ausgewählt werden: Klicken, [Umschalt] + Klicken oder [Strg] + Klicken.

Zum Editieren der Werte (sofern möglich) Doppelklicken Sie auf den Wert.

Werte, die vom Anwender geändert wurden, werden fett dargestellt. Werte, die aus irgend einem Grund ungültig sind, erhalten einen roten Hintengrund. Wenn man mit der Maus über einem ungültigen Wert anhält, wird in einem Popup-Fenster die Fehlerbeschreibung eingeblendet.

Name

Wenn der Array-Index oder Parameter einen Namen erhalten hat, wird er in dieser Spalte dargestellt, andernfalls bleibt die Spalte leer. Einträge in der Liste können in „Gruppen" mit einem Titel organisiert werden (durch Auswahl des Typs „Gruppenüberschrift" bei → *Index einfügen*). Diese Titel werden mit einem dunklen Hintergrund dargestellt. Im Expertenmodus kann dieses Feld mittels Doppelklick bearbeitet werden.

Array

Diese Spalte bezeichnet die Array-Nummer oder den Parametertyp.

Index

Diese Spalte bezeichnet den Array-Index oder die Parameternummer.

R/W

Diese Spalte wird geprüft, wenn ein Listeneintrag in die Steuerung geschrieben werden kann und die Reihe als ☑ R/W markiert ist. Bei Read-only wird nicht geprüft und die ganze Reihe grün dargestellt. Die Spalte R/W kann nur im Expertenmodus verändert werden.

Wert

Die editierbaren „Benutzer"-Werte stehen in der Spalte mit hellem Hintergrund. Ein Wert wird rot dargestellt, wenn er sich von dem Wert unterscheidet, der aktuell in der Steuerung gespeichert ist. Das Wertefeld bleibt leer, wenn kein Wert bekannt ist.

Anwender können diese Werte einfach ändern:
→ Auf den Wert klicken und den neuen Wert eingeben oder mit [Strg]+[V] vom Windows-Zwischenspeicher kopieren.
→ [Enter]**-**Taste drücken, um den neuen Wert zu akzeptieren. Mit [Esc] wird der neue Wert verworfen und der alte behalten. Mit der Maus irgendwo außerhalb der kleinen Editierbox klicken wird automatisch so behandelt, als wäre die **[**Enter]-Taste gedrückt worden.

ACHTUNG!:
Beachten sie, dass geänderte Werte nicht in die Steuerung geschrieben werden, bis Sie die Werte explizit → *Zur Steuerung schreiben.*

Steuerung

Wenn der Array-Editor gerade mit einer Steuerung verbunden ist, werden in dieser Spalte die aktuellen Werte der Steuerung dargestellt. Wenn es keine Steuerung gibt, ist diese Spalte leer. Wenn eine Steuerung verbunden ist, aber keine Werte existieren (zum Beispiel könnte ein Array nicht vorhanden sein), dann ist diese Spalte leer und die ganze Reihe wird grün dargestellt, um anzuzeigen, dass der Eintrag „Read-only" ist und nicht in die Steuerung geschrieben werden kann.

Default

Die Default-Werte sind die Werkseinstellungen der globalen und Achsenparameter, die für ein Reset benutzt werden. Im Array Editor können Default-Werte auch für die anderen Parameter (z.B. Benutzerparameter und Arrays) für das Array-Definitionsdatei-Template definiert werden.

Wenn kein Default definiert ist, bleibt das Feld leer.
Im Expertenmodus kann der Wert editiert werden: → Doppelklick in das Feld.

Minimum / Maximum

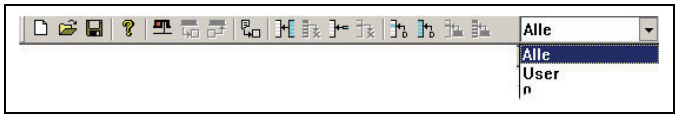
Zeigt den Mindestwert und Maximalwert für den Listeneintrag, falls definiert.
Im Expertenmodus können die beiden Werte editiert werden: → Doppelklick in das Feld.

Beschreibung

Jeder Listeneintrag kann eine optionale Beschreibung erhalten, die dann in dieser Spalte steht.
Im Expertenmodus kann der Beschreibung editiert werden: → Doppelklick in das Feld.

☐ Array-Editor Funktionen

Wird der Array-Editor innerhalb der APOSS IDE gestartet, finden Sie alle Funktionen in der Symbolleiste. Ganz rechts ist das *Array-Auswahlfeld*.



Wird der Array Editor stand-alone gestartet, stehen die Funktionen als normale Buttons zur Verfügung. Das *Array-Auswahlfeld* ist unten links:

Steuerung auswählen

verbindet zu einer Steuerung oder ändert eine vorhandene Verbindung zu einer anderen Steuerung. Damit werden die Arrays und Parameter automatisch von der neuen Steuerung geladen und in der Spalte „Steuerung" dargestellt.

Im stand-alone Array-Editor klicken Sie dazu auf → *Auswahl*.

Von der Steuerung lesen

Alle Arrays und Parameters werden von der Steuerung gelesen und in der Spalte „Steuerung" dargestellt.

Zur Steuerung schreiben

Alle Werte der Spalte „Wert" werden von allen Arrays (auch von den Arrays, die derzeit nicht dargestellt werden) in die Steuerung geschrieben. Einträge ohne „Wert" (d.h. das Feld in „Wert" ist leer) behalten in der Steuerung die vorhandenen Werte. Sobald die Werte in die Steuerung geschrieben wurden, wird die Spalte „Steuerung" aktualisiert, so dass erkennbar ist, dass neue Werte in der Steuerung sind.

Überlagert Beschreibungen

Mit → können Namen, Beschreibungen, Minimum-, Maximum- und Defaultwerte der Liste von einer vorhandenen Array-Definitionsdatei in den Array Editor kopiert werden.

Diese Funktion ist im stand-alone Array-Editor nicht verfügbar.

Array einfügen

Arrays können nur im Expertenmodus → eingefügt werden. Ein Dialogfeld bietet die Eingabe der Array-Nummer und –Größe an.

Array löschen

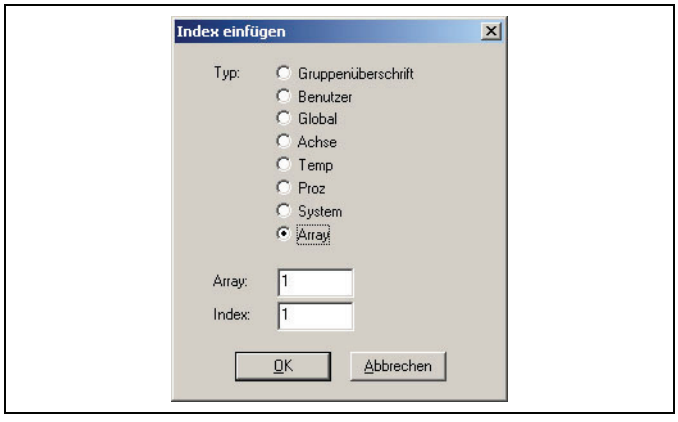
Markieren Sie das Array und klicken Sie auf → oder drücken Sie die [Entf]-Taste,
oder klicken Sie im stand-alone Array-Editor auf → *Array löschen*.

Diese Funktion ist im Benutzermodus nicht verfügbar. Arrays können nur im Expertenmodus gelöscht werden.

Index einfügen

fügt einen neuen Index ein und zwar vor dem gerade ausgewählten Eintrag in der Liste. Ein Dialogfeld bietet die Auswahl des Typs an:

Diese Funktion ist im Benutzermodus nicht verfügbar. Indizes können nur im Expertenmodus eingefügt werden.



MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

Index löschen

Markieren Sie die gewünschte Zeile und klicken Sie auf → oder drücken Sie die [Entf]-Taste, oder klicken Sie im stand-alone Array-Editor auf → *Index löschen.*

Diese Funktion ist im Benutzermodus nicht verfügbar. Indizes können nur im Expertenmodus gelöscht werden.

Setze auf Default

Markieren Sie einen oder mehrere Listeneinträge und klicken Sie auf. . Damit wird der „Wert" von allen ausgewählten Listeneinträgen auf die entsprechenden Default-Werte gesetzt. Alle vorherigen Werte werden überschrieben.

Beachten Sie, Werte von Einträgen mit leeren Default-Feldern werden nicht geändert.

Setze alle auf Default

setzt den „Wert" von allen sichtbaren Listeneinträgen auf die entsprechenden Default-Werte. Alle vorherigen Werte werden überschrieben. Die Werte von Arrays, die nicht im Array Auswahlfeld ausgewählt sind, also in der Liste nicht zu sehen sind, werden auch nicht geändert.

Beachten Sie, Werte von Einträgen mit leeren Default-Feldern werden nicht geändert.

Setze auf Steuerung

setzt den „Wert" von allen ausgewählten Listeneinträgen auf die aktuellen Werte der Steuerung.

Beachten Sie, dass Werte, deren Feld in „Steuerung" leer ist, nicht geändert werden.

Setze alle auf Steuerung

setzt den „Wert" von allen sichtbaren Listeneinträgen auf die aktuellen Werte der Steuerung. Die Werte von Arrays, die nicht im Array Auswahlfeld ausgewählt sind, also in der Liste nicht zu sehen sind, werden nicht geändert.

Beachten Sie, dass Werte deren Feld in „Steuerung" leer ist, nicht geändert werden.

Array Auswahlbox

Die Array-Auswahlbox wird in der Array-Editor Symbolleiste rechts bzw. in der stand-alone Anwendung unten links angezeigt. Hier können entweder alle Arrays oder individuelle Arrays für die Anzeige ausgewählt werden. Denn es mag einfacher sein, mit einzelnen Arrays und somit kleineren Listen zu arbeiten.

Öffnen

Mit *Datei* → *Öffnen* können Sie eine andere neue Array-Definitionsdatei laden.

Im stand-alone Array Editor klicken Sie dazu auf → *Öffnen*.

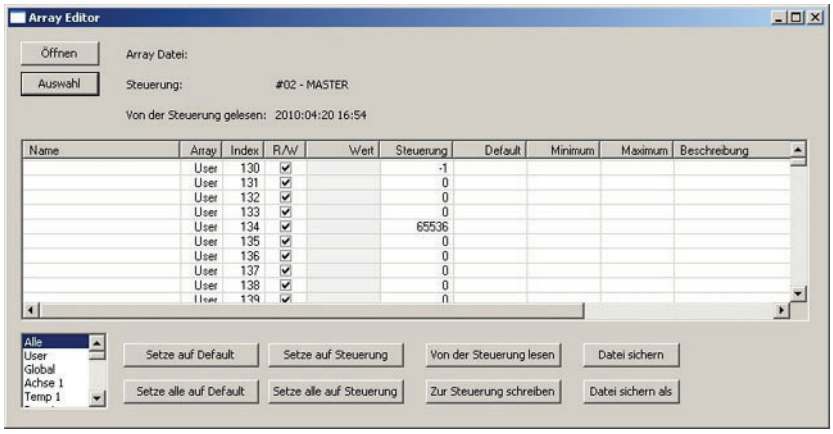
Speichern / Speichern als

Mit *Datei* → *Speichern* Sie die aktuellen Inhalte als Array-Definitionsdatei; mit → *Speichern als* können Sie die Datei unter einem anderen Namen speichern.

Im stand-alone Array Editor klicken Sie dazu auf → *Datei sichern* bzw. → *Datei sichern als*.

□ Benutzer-Modus OHNE Array-Definitionsdatei

Dies ist das Dialogfeld für den stand-alone Array-Editor im Benutzermodus.

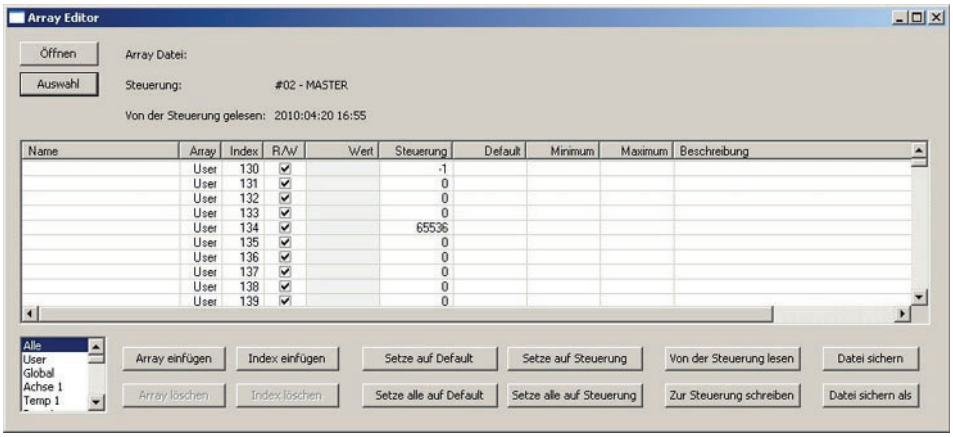


Um es zu benutzen folgen Sie diesen Schritten:

1. Klicken Sie auf → *Auswahl* dann öffnet sich das normale APOSS Dialogfeld *„Steuerung auswählen"* zum Setzen der Schnittstellen-Parameter, Öffnen der Schnittstelle und auswählen einer Steuerung. Sobald die Verbindung zur Steuerung geöffnet ist, erscheint der Name der Steuerung sowie Datum und Uhrzeit, wann die Arrays „Von der Steuerung gelesen" wurden.
   Alle vorhandenen Arrays in der Steuerung (inklusive der Benutzer-Arrays, globalen Parameter-Arrays, permanente und temporäre Achsparameter-Arrays, Achsenprozessdaten-Arrays, usw.) werden automatisch eingelesen und dargestellt. Die Array-Nummer (oder der Name für Benutzer-, globalen und Achsen-Arrays) und den Index finden Sie in den gleichnamigen Spalten, den Wert in der Spalte „Steuerung".
   Das Array-Auswahlfeld links unten zeigt alle Arrays, die in der Steuerung gefunden werden.
2. Sobald eine Steuerung geöffnet ist, können jederzeit alle Arrays mit → *Von der Steuerung lesen* wieder gelesen und die aktuellen Werte dargestellt werden. Der Eintrag „Von der Steuerung gelesen" wird aktualisiert.
3. Klicken Sie auf ein Array im Array-Auswahlfeld um es zu bearbeiten.
4. Um den Wert eines Array-Elements zu ändern, klicken Sie in der Spalte „Wert" auf das Array-Element, eine kleine Editierbox öffnet sich: Entweder geben Sie den Wert direkt ein oder Sie fügen ihn mit [Strg]+[V] aus der Windows Zwischenablage ein.
   Beachten Sie, dass einige Array-Elemente Read-only sein können. Diese sind grün dargestellt und können vom Anwender nicht geändert werden.
   Mit [Enter] wird die neue Wert akzeptiert. Drücken Sie [Esc], wenn Sie den neuen Wert verwerfen und den vorherigen Wert behalten wollen.
   Wenn Sie mit der Maus auf eine beliebige andere Stelle klicken, wird automatisch der Wert wie mit [Enter] akzeptiert.
   Beachten Sie, dass die Spalte „Wert" nur jene Werte enthält, die geändert wurden. Die Werte werden Rot dargestellt, wenn sie sich von den tatsächlich in der Steuerung gespeicherten unterscheiden.
5. Klicken Sie auf → *Zur Steuerung schreiben*. Damit werden alle Werte der Spalte „Wert" in alle Arrays in die Steuerung geschrieben. Array Elemente ohne Wert (d.h. keinen Eintrag in der Spalte „Wert") behalten ihre vorhandenen Werte. Sobald die Arrays zur Steuerung geschrieben wurden, wird die Spalte „Steuerung" aktualisiert, damit man die neuen Werte der Steuerung sieht.
6. Falls gewünscht, kann man mit → *Auswahl* eine andere Steuerung auswählen und bearbeiten.
7. Sobald das Array-Editor Dialogfenster geschlossen wird und Änderungen gemacht wurden, wird abgefragt, ob die Änderungen in eine Datei gesichert werden sollen. Das Speichern der Änderungen ermöglicht es die geänderten Werte später herauszufinden und zu überprüfen (mittels → *Öffnen*).

☐ Experten-Modus OHNE Array-Definitionsdatei

Dialogfeld für den stand-alone Array-Editor im Expertenmodus. Benutzen Sie die Icons der Symbolleiste, wenn der Array-Editor innerhalb der APOSS-IDE gestartet wurde.



Das Dialogfeld .wird genauso benutzt wie im Benutzermodus. Der einzige Unterschied ist, dass nun auch die Einfügen- und Löschen-Schaltflächen verfügbar sind. Da aber keine Array-Definitionsdatei geladen wurde, erwartet der Array-Editor direkt mit der Steuerung zu arbeiten. Daher verhalten sich die Einfügen- und Löschen-Schaltflächen etwas anders, als wenn eine Datei geladen wäre.

Array einfügen

Ermöglicht das Einfügen eines neuen Arrays nach dem letzten vorhandenen Array. Der Anwender wird aufgefordert die Array-Größe einzugeben.

ACHTUNG!:
Dieses neue Array wird erst mit → *Zur Steuerung schreiben* zur Steuerung hinzugefügt.

Array löschen

Löscht das gerade ausgewählte Array. Beachten Sie, dass nur Benutzer-Array gelöscht werden können, keine Parameter. Da „Löcher" (d.h. fehlende Array-Nummern) nicht erlaubt sind, werden alle folgenden Arrays neu nummeriert, um die entstandene Lücke zu schließen. Tatsächlich wird aber das Array erst mit → *Zur Steuerung schreiben* gelöscht.

Index einfügen

Fügt einen einzelnen Array-Index vor dem gerade ausgewählten ein. Wenn mehr als ein Array-Index ausgewählt ist, wird der neue vor dem ersten ausgewählten Index eingefügt. Wenn kein Index ausgewählt ist, wird der neue Index als letzter dem Array hinzugefügt. Alle Array-Elemente, die dem neuen Index folgen, werden neu nummeriert, um Platz für den neuen Index zu schaffen.

Beachten Sie, neue Indizes können nur zu Benutzer-Arrays hinzugefügt werden.

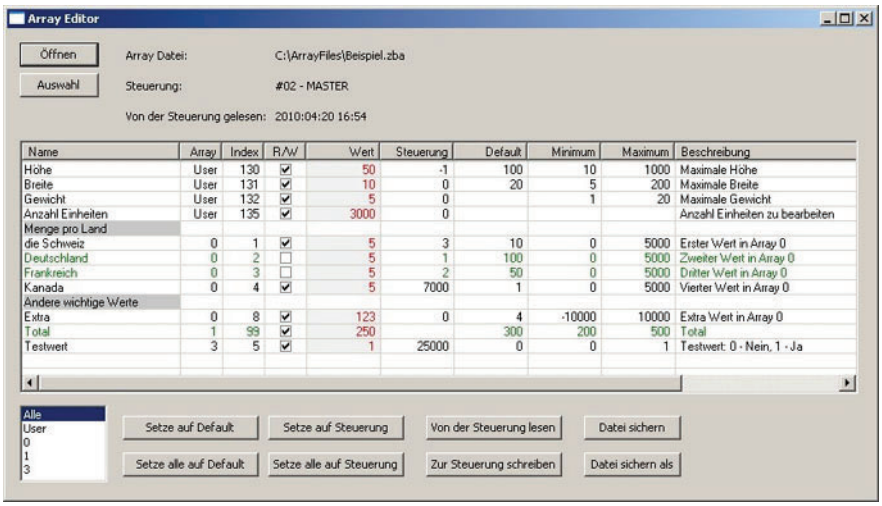
Index löschen

Löscht das gerade ausgewählte Array-Element. Wenn alle Array-Elemente ausgewählt sind, wird das Array selbst gelöscht. Da „Löcher" (d.h. fehlende Nummern) nicht erlaubt sind, werden alle folgenden Array-Elemente neu nummeriert, um die entstandene Lücke zu schließen. Tatsächlich wird aber das Array-Element erst mit → *Zur Steuerung schreiben* gelöscht.

Beachten Sie, dass nur Benutzer-Array-Elemente gelöscht werden können.

□ Benutzer-Modus MIT Array-Definitionsdatei

Dialogfeld für den Benutzermodus im stand-alone Array-Editor, wenn eine Array-Definitionsdatei benutzt wird:



Um den Array-Editor mit einer Array-Definitionsdatei zu benutzen folgen Sie diesen Schritten:

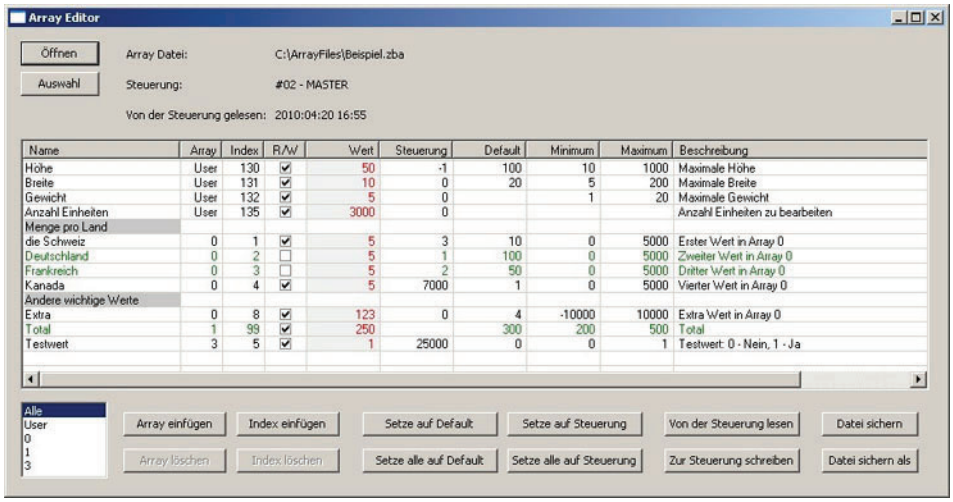
1. Klicken Sie auf → *Öffnen*. Im folgenden normalen APOSS „*Datei öffnen*" Dialogfeld wählen Sie eine Array-Definitionsdatei „.zba" aus. Der Inhalt der Datei wird im Dialogfeld dargestellt. Zu diesem Zeitpunkt ist die Spalte „Steuerung" noch leer (weil keine Arrays von der Steuerung gelesen wurden), aber alle andere Spalten können Werte enthalten.
2. Klicken Sie auf → *Auswahl* und wählen Sie eine Steuerung (wie oben) aus. Alle Arrays werden dann von der Steuerung eingelesen und mit den vorhandenen Werte der Array-Definitionsdatei „vermengt".
3. Sobald eine Steuerung geöffnet ist, können jederzeit alle Arrays mit → *Von der Steuerung lesen* wieder gelesen und die aktuellen Werte dargestellt werden. Der Eintrag „Von der Steuerung gelesen" wird aktualisiert.
4. Wenn das Array-Editor Dialogfenster geschlossen wird und Änderungen gemacht wurden, wird der Anwender gefragt, ob er die Änderungen in eine Datei sichern will. Das Speichern der Änderungen ermöglicht es, die geänderten Werte später herauszufinden und zu überprüfen (mittels → *Öffnen*).

Bitte beachten Sie folgende Punkte beim Benutzen des Array-Editors mit einer Array-Definitionsdatei:

- Ein Array-Element (oder komplettes Array) das nicht in einer Array-Definitionsdatei definiert ist, wird nicht dargestellt und kann auch nicht aufgerufen werden, auch wenn es in der Steuerung vorhanden ist. Die Array-Definitionsdatei ermöglicht nur den Zugang zu den Array-Elementen, die in der Datei definiert sind.
- Ein Array-Element, das in der Array-Definitionsdatei definiert, aber nicht in der Steuerung vorhanden ist, wird zwar in der Liste dargestellt, ist jedoch deaktiviert und grün markiert. Es kann weder ausgewählt noch geändert werden.
- Array-Elemente können als „read-only" in der Array-Definitionsdatei markiert sein. In diesem Fall ist die Checkbox in der Spalte „R/W" (Read/Write) leer und das Array-Element wird grün dargestellt. Read-only Array-Elemente können nicht geändert werden.
- Gruppenüberschriften – sofern in der Datei definiert – werden mit grauem Hintergrund in der Spalte „Name" dargestellt. Diese Reihen können nicht ausgewählt werden. Die Array-Element-Definitionen in einer Gruppe können verschiedenen Arrays entsprechen; alle Definitionen in einer Gruppe müssen nicht vom gleichen Array kommen.

☐ Experten-Modus MIT Array-Definitionsdatei

Dialogfeld für den Expertenmodus im stand-alone Array-Editor, wenn eine Array-Definitionsdatei benutzt wird. Benutzen Sie die Icons der Symbolleiste, wenn der Array-Editor innerhalb der APOSS-IDE gestartet wurde.



Das Dialogfeld wird genau so benutzt wie im Benutzer-Modus: Der Unterschied ist, dass nun auch die Einfügen- und Löschen-Schaltflächen verfügbar sind und dass auch andere Spalten als „Wert" bearbeitet werden können. Außerdem können Gruppenüberschriften und deaktivierte Listenpositionen (d.h. Positionen, die nicht in der Steuerung vorhanden sind) ausgewählt werden. So kann man auch diese Reihen zum Einfügen oder Löschen von Indizes auswählen. Da jedoch eine Array-Definitionsdatei geladen wurde, wird der Array-Editor mit der Datei arbeiten statt mit der Steuerung. Daher verhalten sich die Funktionen Einfügen und Löschen etwas anders, als wenn keine Datei geladen wäre.

Array Einfügen

Zum Einfügen eines neuen Arrays mit Eingabe der Array-Nummer und Array-Größe. Falls keine Arrays von der Steuerung eingelesen wurden, werden alle neuen Array-Elemente aktiviert. Falls bereits Arrays von der Steuerung eingelesen waren, werden die neuen Array-Elemente aktiviert oder deaktiviert je nachdem ob sie in den von der Steuerung eingelesenen Werten vorhanden waren oder nicht. Beachten Sie, dass das Einfügen von Arrays diese nur zur Array-Definitionsdatei hinzufügt; die Arrays werden nicht zur Steuerung hinzugefügt.

Array löschen

Löscht das aktuell ausgewählte Array aus der Array-Definitionsdatei. Das Array wird aber nicht in der Steuerung gelöscht. In diesem Fall ist das Löschen eines Arrays gleichbedeutend mit dem Entfernen des Benutzerzugangs zu diesem Array.

ACHTUNG!:
Beachten Sie aber, wenn eine Array-Definitionsdatei bearbeitet wurde, können Benutzer-, globale , Achsen-Arrays und andere Arrays gelöscht werden!

Index einfügen

Fügt ein einzelnes Array-Element vor dem gerade ausgewählten Array-Element ein. Falls mehr als ein Array-Element ausgewählt ist, wird das neue vor dem ersten markierten Element eingefügt. Wenn kein Element ausgewählt ist, wird das neue Element am Ende der Liste hinzugefügt. Ein Dialog bietet an, genau zu spezifizieren, welche Art von Element eingefügt werden soll.

Index löschen

Löscht alle Array-Elemente, die aktuell markiert sind aus der Array-Definitionsdatei. Die Array-Element in der Steuerung werden nicht verändert. In diesem Fall ist das Löschen von Array-Elementen gleichbedeutend mit dem Entfernen des Benutzerzugangs zu diesen Elementen.

ACHTUNG!:
Beachten Sie aber, wenn eine Array-Definitionsdatei bearbeitet wurde, können Benutzer-, globale und Achsen-Array-Elemente und andere Array-Elemente gelöscht werden!

□ Array-Definitionsdatei erzeugen

Array-Definitionsdatei sind einfache Textdateien mit der Dateierweiterung „.zba" und können direkt im Expertenmodus erzeugt werden. Sie sind jedoch so aufgebaut, dass sie auch mit einem Tabellenkalkulationsprogramms erzeugt werden können. Mit der nötigen Sorgfalt können sie auch mit einem einfachen Texteditor erzeugt und bearbeitet werden.

Um eine Definitionsdatei mit dem Array-Editor zu erzeugen, folgen Sie diesen Schritten:

1. Array-Editor starten.
2. Mit → *Auswahl* wählen Sie eine Steuerung aus und verbinden diese. Alle Arrays werden dann automatisch von der Steuerung eingelesen.
3. Mit → *Sichern als* schreiben Sie eine Array-Definitionsdatei auf die Festplatte. Dies ist wichtig, da damit der Array-Editor in den Modus „MIT Datei" wechselt.
4. Bearbeiten Sie die Array-Elemente soweit notwendig um Namen, Beschreibungen, Minimum- und Maximumwerte usw. zu bestimmen.
5. Klicken Sie auf → *Sichern*, wenn Sie fertig sind.

Im Anschluss sehen Sie ein Beispiel eine Array-Definitionsdatei. Die Datei enthält für jede Definition eine Zeile. Jede Definition besteht aus einem oder mehreren „Objekten" entweder durch ein Komma oder durch ein Semikolon getrennt. Das erste Objekt in jeder Zeile ist ein „Typ" und kann einen der folgenden Werte haben:

Date	Das folgende Objekt ist das Datum, das im Kopf des Dialogfelds in „Von der Steuerung gelesen" dargestellt wird.
Widths	Die folgenden Objekte sind die benutzten Spaltenbreiten.
User	Definiert einen Benutzer-Array-Index.
Global	Definiert einen globalen Parameter.
System	Definiert einen Systemprozessdatenparameter.
Axis	Definiert einen permanenten Achsparameter.
Temp	Definiert einen temporären Achsparameter
Proc	Definiert einen Achsenprozessdatenparameter
Array	Definiert einen normalen Array-Index.
Group	Definiert eine Gruppenüberschrift zur Darstellung in der Liste.

Jede Zeile in dieser Datei, die nicht einen bekannten Objekttyp enthält, wird als Kommentar behandelt und ignoriert.

Im folgenden der Objekttyp, von dem einer in jeder Spalte in der Array-Editor-Liste sein sollte, mit Ausnahme in der Spalte Steuerung. Die Spalte Steuerung wird nicht in der Datei gespeichert. Diese Objekte müssen genau in der folgenden Reihenfolge angeordnet sein:

Array, Index, Name, Wert, Default, Minimum, Maximum, Beschreibung, Read/Write Flag

Die Spalte Array sollte für Benutzer- und globale Objekte leer sein, sie sollte aber die auf 1 bezogene Achsnummer für permanente und temporäre Achsenprozessdaten-Objekte und die auf 0 bezogene Array-Nummer für Array-Objekte enthalten.

Für Gruppenobjekte wird die Darstellung der Überschrift durch die Spalte Array bestimmt; sie sollte eine der folgenden Darstellungen enthalten:

-1	Wird immer dargestellt.
-2	Wird nur dargestellt wenn „Benutzer”-Array (oder „Alle”) ausgewählt ist.
-3	Wird nur dargestellt wenn das „Globale” Array (oder „Alle”) ausgewählt ist.
-100-n	Wird nur dargestellt wenn „Achse n” (oder „Alle”) ausgewählt ist (z.B. -101 für Achse 1).
-200-n	Wird nur dargestellt wenn „Temp n” (oder „Alle”) ausgewählt ist (z.B. -201 für Achse 1).
-300-n	Wird nur dargestellt wenn „Proc n” (oder „Alle”) ausgewählt ist (z.B. -301 für Achse 1).
Auf 0 basierende Array Nummer	Wird nur dargestellt wenn das entsprechende Array (oder „Alle“) ausgewählt ist.

Die Spalte Index sollte für Gruppenobjekte 0 sein.

Die Texte für Name und Beschreibung können Kommata, Semikolon und Leerzeichen enthalten, dann muss aber der Text innerhalb von Anführungszeichen stehen. Diese Texte dürfen keine Anführungszeichen (doppelte Apostroph) enthalten.

Das Read/Write Flag kann entweder 0 (für Read-only) oder 1 (für Read/Write) sein.

Beispiel einer Array-Definitionsdatei

```
Date,2009:09:27 10:52
Widths,150,40,40,40,70,70,70,70,70,365
Dies ist eine Kommentarzeile.
User,,130,Height,50,100,10,1000,"Maximale Höhe.",1
User,,131,Breite,10,20,5,200,"Maximale Breite",1
User,,132,Gewicht,5,,1,20,"Maximale Gewicht",1
User,,135,"Anzahl Einheiten",3000,,,,"Anzahl Einheiten zu bearbeiten",1
***** Weitere Kommentarzeile *****
Group,0,0,"Menge pro Land",,,,,,0
Array,0,1,"die Schweiz",5,10,0,5000,"Erster Wert in Array 0",1
Array,0,2,Deutschland,5,100,0,5000,"Zweiter Wert in Array 0",0
Array,0,3,Frankreich,5,50,0,5000,"Dritter Wert in Array 0",0
Array,0,4,Kanada,5,1,0,5000,"Vierter Wert in Array 0",1
Group,0,0,-1,0,"Andere wichtige Werte",,,,,,0
Array,0,8,Extra,123,4,-10000,10000,"Extra Wert in Array 0",1
Array,1,99,Total,250,300,200,500,Total,1
Array,3,5,Testwert,1,0,0,1,"Testwert: 0 - Nein, 1 - Ja",1
```

Wenn eine Array-Definitionsdatei mit einem Tabellenkalkulationsprogramm erzeugt wird, dann müssen die Tabellenspalten genauso wie oben beschrieben gesetzt werden. Das Tabellenblatt kann dann als „CVS (Comma delimited)” Dateityp gespeichert werden. Das generiert die richtige durch Komma getrennte oder durch Semikolon getrennte Datei. Diese Datei sollte dann unter einem neuen Namen gespeichert werden, um die Dateierweiterung „.zba” zu erhalten.

□ APOSS Oszilloskop

Das APOSS Oszilloskop ist ein Werkzeug für die Aufnahme und grafische Darstellung von Werten einer oder mehrerer Steuerungen während diese in Betrieb sind und Programme ausführen. Es wurde als Hilfe beim Debuggen von Prototyp-Anwendungen, beim Erkennen von Problemen im Betrieb und zum Tuning der internen Steuerungsparameter für eine optimale Systemleistung entwickelt.

Das Arbeiten mit dem Oszilloskop bedingt grundsätzlich folgende Schritte:

1. Entscheiden welche Oszilloskop-Version benutzt wird.
2. Oszilloskop starten (außer bei TESTSETP Oszilloskop, siehe dort.)
3. Festlegen der „Kurven" (d.h. der Werte, Variablen, Status usw.) die von Interesse sind und die bei der Diagnose von Problemen brauchbar sind.
4. Den Test starten (d.h. die Anwendung in der Steuerung ausführen). Die Daten für diese Kurven werden dann automatisch erfasst.
5. Die sich ergebenden Kurven analysieren.
6. Änderungen, welche auch immer angebracht sind, durchführen (d.h. Änderung der Kurvendefinition, der Steuerungsparameter, des Anwendungsprogramms usw.) und den Test wiederholen.

Die genauen Details der Schritte 3 und 4 hängen von der Oszilloskop-Version ab, die eingesetzt wird. Dies wird nachfolgend beschrieben. Für die letzten beiden Schritte ist der Anwender zuständig.

Die folgende allgemeine Symbolleiste hilft die Kurven zu definieren und die Tests zu starten (Schritt 3 und 4):

ALLE Schnittstellen schließen zu allen Steuerungen.

Beachten Sie, dass diese Funktion auch die Verbindungen zu Steuerungen in anderen Fenstern schließt (z.B. von jedem geöffneten APOSS Fenster).

Neu verbinden mit den Standard-Schnittstellen – Verbindet alle im Oszilloskop definierten Kurven mit der derzeit definierten Standard-Steuerung. (Siehe auch *Einstellungen* → *Schnittstelle* im Kapitel „APOSS Benutzeroberfläche".)

Falls Kurven vorher mit verschiedenen Steuerungen verbunden waren, werden diese Verbindungen geändert und mit der neuen Steuerung verbunden.

Alles auf eine neue Steuerung übernehmen – Mit dieser Funktion wird eine neue Steuerung ausgewählt und werden alle Kurven im Oszilloskop zu dieser neuen Steuerung verschoben.

Neue Kurve hinzufügen zum Oszilloskop.

Start – Startet eine Oszilloskop „Testfahrt" und damit die Aufzeichnung der Kurvendaten.

Pause **–** Hält eine Oszilloskop „Testfahrt" an und die Datenaufzeichnung pausiert. Die Testfahrt kann mit ▶ erneut gestartet werden.

Stop **-** Beendet eine Oszilloskop „Testfahrt" und stoppt die Aufzeichnung der Kurvendaten. Wenn die Kurvendaten in der Steuerung gespeichert werden, dann werden sie in diesem Moment zum Oszilloskop hochgeladen..

Aufzeichnungen löschen – löscht alle gespeicherten Daten aller Kurven. Dies beinhaltet auch die Kurven, die gerade nicht dargestellt werden.

□ Oszilloskop-Versionen

Es gibt vier Oszilloskop-Versionen. Jede Version ist für eine andere Anwendung entwickelt, aber alle arbeiten grundsätzlich in der gleichen Art und Weise und haben ein ähnliches Erscheinungsbild. *Free Run, Single Shot* und *TESTSETP Oszilloskop* sind zum Debuggen und für die Diagnose vorgesehen, das *Tune Oszilloskop* zum Optimieren der Systemleistung.

Entscheiden Sie, welche Oszilloskop-Version für die Aufgabe infrage kommt:

Free Run Oszilloskop

Das *Free Run Oszilloskop* ist ein Werkzeug, das speziell für die Diagnose bei betriebsbedingten Problemen in laufenden Systemen entwickelt wurde. Es kann aber auch zum Debuggen von Prototype-Anwendungen benutzt werden. Es wird als „Free Run" (Freilauf) bezeichnet, weil die Anwendung in der Steuerung frei laufen kann, ohne dass man beachten muss wie das Oszilloskop benutzt wird oder ob es überhaupt benutzt wird. Die Auswirkung des Vorhandenseins und Gebrauchs des Oszilloskops in einer laufenden Anwendung wurde so weit wie möglich minimiert. Dies ermöglicht es der Steuerung sich so zu verhalten wie in einer tatsächlichen Betriebssituation.

Beim *Free Run Oszilloskop* werden die Werte während des aktuellen Tests aufgezeichnet und in Echtzeit dargestellt. Die Informationen dafür werden mittels PDOs von der Steuerung abgeholt.

ACHTUNG!:
Beachten Sie, dass PDOs von einigen älteren Steuerungen sowie von einigen Schnittstellen nicht unterstützt werden.

Free Run Oszilloskop im Polling Modus

Unter bestimmten Umständen kann es sein, dass der Anwender PDOs nicht mit dem Oszilloskop verwenden kann oder will. Gründe hierfür können sein:

- Wenn das laufende Programm schon alle verfügbaren Typen von PDOs verwendet, steht für das Oszilloskop kein Typ mehr zum Empfangen der Kurvendaten von der Steuerung zur Verfügung.
- In der Steuerung könnte eine ältere Firmware-Version installiert sein, die die dynamische Konfiguration von PDOs nicht unterstützt. Also kann das Oszilloskop die erforderlichen PDOs nicht konfigurieren.
- Die Steuerung könnte mit einer Schnittstelle verbunden sein, die PDOs nicht unterstützt. Dazu gehört zum Beispiel die EtherCAT-Schnittstelle, die derzeit keine PDOs unterstützt.

In diesen Fällen kann das *Free Run Oszilloskop* entweder nicht benutzt werden oder nur in dem langsameren *Polling-Modus*. Beachten Sie, dass Polling signifikante Auswirkungen auf die Leistung und Antwortzeiten der Steuerung und des Netzwerks haben kann.

Alternativ kann aber das *Single Shot Oszilloskop* eingesetzt werden.

Single Shot Oszilloskop

Das *Single Shot Oszilloskop* ist eher für das Debuggen von Prototyp-Anwendungen als für die Diagnose bei betriebsbedingten Problemen im laufenden System entwickelt worden. Es kann aber für beides verwendet werden. Es eignet sich auch für Situationen, in denen das *Free Run Oszilloskop* nicht eingesetzt werden kann.

Es wird als „Single Shot" bezeichnet, weil die Kurvendaten in der Steuerung aufgezeichnet werden und erst wenn sie vollständig sind, „auf einen Schlag" zum Oszilloskop hochgeladen werden.

Die Werte werden in der Steuerung in ein TESTSETP-Array aufgezeichnet. Die Anweisung für dieses Arrays kann entweder manuell durch den Anwender (d.h. ein Benutzer-Array für diesen Zweck definieren) oder automatisch durch Nutzen des verfügbaren freien Speichers in der Steuerung ausgeführt werden. In jedem Fall kann das *Single Shot Oszilloskop* die Speicherverwaltung des Anwendungsprogramms beeinflussen. Das TESTSETP-Array wird durch das *Single Shot Oszilloskop* automatisch konfiguriert, sobald die Aufzeichnung beginnt.

ACHTUNG!:
Beachten Sie, dass einige ältere Steuerungen die automatische Konfiguration von TESTSETP-Arrays nicht unterstützen und daher das *Single Shot Oszilloskop* nicht benutzt werden kann; statt dessen sollte dann das *TESTSETP Oszilloskop* eingesetzt werden.

TESTSETP Oszilloskop

Das *TESTSETP Oszilloskop* wurde entwickelt um benutzerdefinierte TESTSETP-Arrays, die mit dem Befehl TESTSETP in der Steuerung erzeugt wurden, darzustellen. Wie beim Single Shot Oszilloskop werden die Werte mittels eines TESTSETP-Arrays aufgenommen und zum Oszilloskop hochgeladen, sobald der Test fertig ist. Das TESTSETP Oszilloskop stützt sich darauf, dass die Anwender ihre eigenen TESTSETP-Arrays innerhalb ihrer Anwendungsprogramme konfigurieren. Die Arrays werden nicht automatisch vom Oszilloskop konfiguriert; es liest einfach schon vorhandene TESTSETP-Arrays, die durch die Anwendung des Benutzers erzeugt wurden.

Das *TESTSETP Oszilloskop* ist in folgenden Situationen nützlich:

- Für Anwendungen, bei denen der Anwender innerhalb des Anwendungsprogramms ein kunden-spezifisches TESTSETP-Array erzeugt hat. Dies schließt die Nutzung von mehreren Aufzeichnung innerhalb eines einzelnen TESTSETP-Arrays mit ein.
- Zum Testen von Steuerungen mit einer Firmware, die zu alt ist, um die automatische Konfiguration von TESTSETP-Arrays zu unterstützen. In diesem Fall kann das Single Shot Oszilloskop nicht benutzt werden

Da das *TESTSETP Oszilloskop* einfach die vorhandenen Arrays liest, ist es nicht notwendig, die Aufzeichnung zu starten oder zu stoppen. Demzufolge arbeitet das TESTSETP Oszilloskop geringfügig anders als das Single Shot und Free Run Oszilloskop. Dies sind folgende Schritte:

1. Bevor das TESTSETP Oszilloskop gestartet wird, muss der Anwender APOSS-Programm ausführen, das die gewünschten Daten mittels der verschiedenen TESTSETP Befehle aufzeichnet. Dieses Programm kann ganz normal mit *Steuerung* → *Ausführen [F5]* gestartet werden.
2. Wenn dieser Test beendet ist kann das TESTSETP Oszilloskop gestartet werden. Dieses wird dann automatisch versuchen, das TESTSETP Array von der Steuerung zu lesen.

ACHTUNG!:
In früheren APOSS Versionen war diese Funktion mit *Testfahrt* → *Aufzeichnung anzeigen* verfügbar. Beginnend mit MCO 5.00 wurde sie durch TESTSETP Oszilloskop ersetzt.

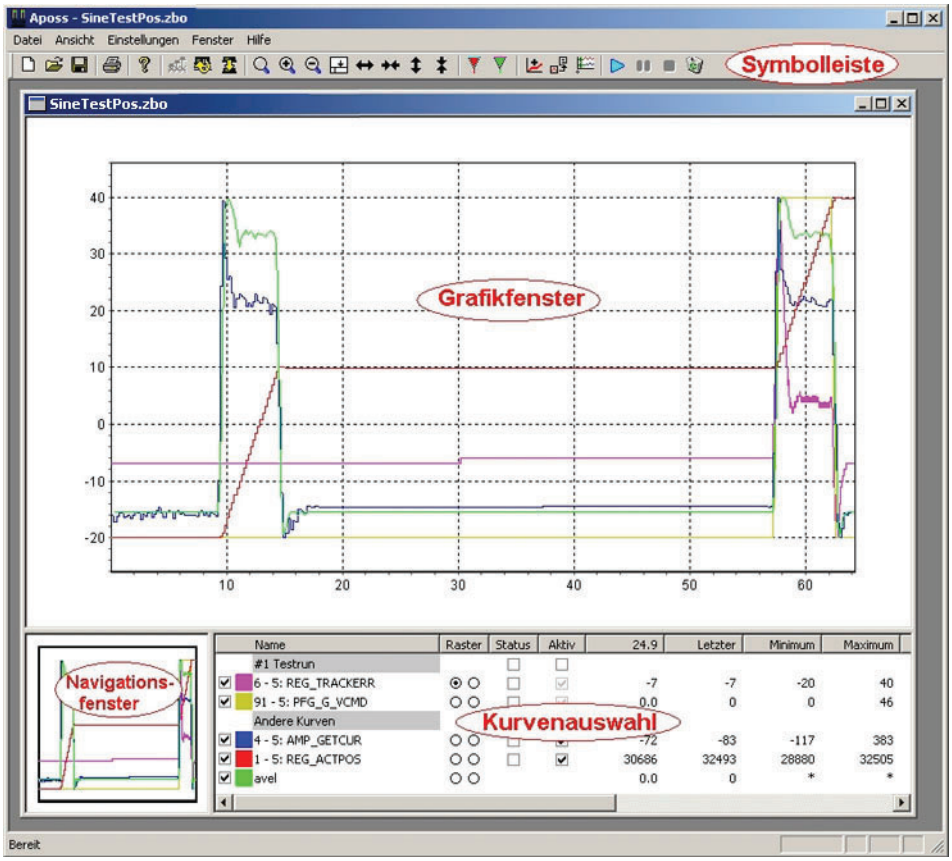
Die *Testfahrt*-Funktion in diesen frühren APOSS Versionen ermöglichte es, die Ergebnisse der von → *Testfahrt ausführen* und → *Aufzeichnung anzeigen* in „.txt" Dateien zu speichern. Diese „alten" .txt-Datein können von den neuen APOSS-Versionen gelesen und dargestellt werden. Siehe Alte Testfahrt-Dateien lesen.

Tune Oszilloskop

Das *Tune Oszilloskop* wurde entwickelt um den Anwender beim „Tuning" und Optimieren der internen Steuerungsparameter zu unterstützen. Dies ist ein wichtiger Schritt um die beste und möglichst laufruhige Leistung der Steuerung in jeder spezifischen Anwendung zu erhalten.

ACHTUNG!:
Beginnend mit MCO 5.00 wurde die frühere Funktion „*Testfahrt* → *Testfahrt ausführen*" durch das *Tune Oszilloskop* ersetzt.

☐ Oszilloskop-Fenster



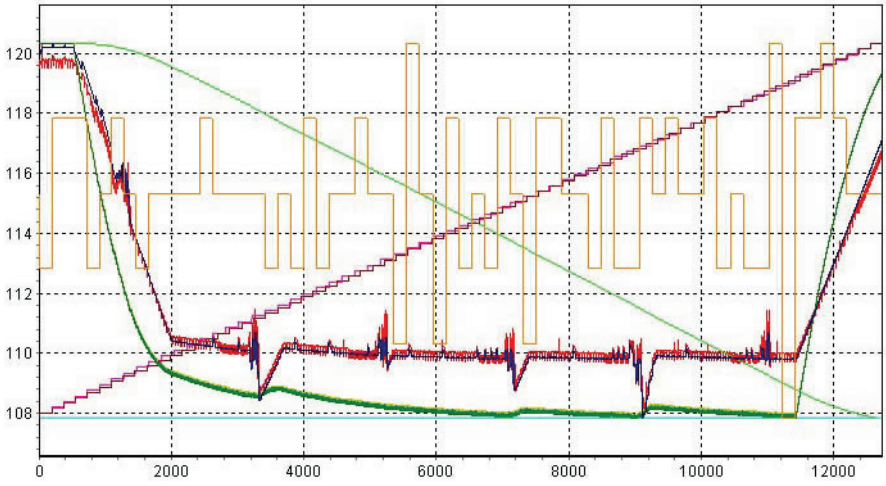
Wenn ein Oszilloskop-Fenster aktiv ist, werden die normalen APOSS-Menüs und die Symbolleiste durch ein Kundenmenü und eine Symbolleiste für das Oszilloskop ersetzt.

Das Oszilloskop-Fensters als Ganzes können Sie mit den Standard-Windows Methoden verändern. Um die einzelnen Fenster zu ändern, stellen Sie den Cursor auf Trennlinien und ziehen diese mit gedrückter linker Maustaste in die gewünschte Richtung.

Beachten Sie, dass das Oszilloskop die Darstellung der Diagramme beendet, sobald das Fenster zu klein wird, um noch brauchbare Diagramme darzustellen.

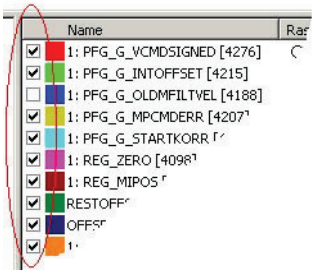
Grafikfenster

In diesem Hauptfenster werden die Kurven dargestellt, die die erfassten Datenwerte repräsentieren. Man kann das Diagramm in diesem Fenster zoomen, verschieben und auf verschiedene Arten modifizieren, um mehr oder weniger Details der Kurven zu sehen.



Jede farbige Kurve in dem oben gezeigten Diagramm repräsentiert einen Wert, der während der Ausführung einer Testfahrt gemessen wurde. Die X-Achse des Diagramms repräsentiert die Zeit, normalerweise in Millisekunden gemessen. Die Zeit ist im Diagramm aufsteigend von links nach rechts eingetragen. Beachten Sie, dass die Zeit von der internen Uhr der Steuerung abgeleitet wird und daher Testfahrten nicht bei 0 starten. Mit → *Kurven auf Zeit 0 verschieben* kann man die Zeitaufnahme bei 0 beginnen lassen; so wird das Diagramm besser lesbar.

Individuelle Kurven können einfach durch Klicken in die Kontrollkästchen der Kurvenauswahl ein- und ausgeblendet werden. Auch die anderen Eigenschaften der Kurve (z.B. Farbe, Skalierung, usw.) können in der Kurvenauswahl gesetzt werden.



<u>Zoomen und Verschieben mit der Maus</u>

Die Maustasten können im Grafikfenster wie folgt benutzt werden:

Linke Maustaste (Zoom) – Vergrößern Sie den gewünschten Bereich, indem Sie mit gedrückter linken Maustaste ein Rechteck über diesen ziehen.

ACHTUNG!:
Das ausgewählte Rechteck wird so gestreckt, dass es genau in das Fenster passt. Ist das nicht gewünscht ist, drücken und halten Sie die Taste [Strg], bevor Sie die linke Maustaste drücken.

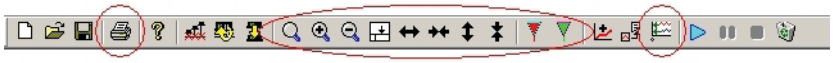
Benutzen Sie die *Zoom reduzieren* oder *Zoom zurücksetzen* um wieder zu verkleinern.

Rechte Maustaste (Verschieben) – Mit gedrückter rechter Maustaste können Sie das Diagramm verschieben (d.h. innerhalb des Fensters bewegen).

MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

Schaltflächen der Symbolleiste

Symbolleiste des Oszilloskop Grafikfensters:



Drucken – Druckt das Diagramm

Zoom Zurücksetzen – Setzt das ganze Diagramm auf Standardgröße zurück.

Zoom 2-fach verstärken – Vergrößern, so dass mehr Details dargestellt werden.

Zoomstufe reduzieren – Verkleinern, so dass mehr vom Diagramm zu sehen ist.

Splitter (Trennleisten) anpassen – Passt die Splitter des Fensters so an, dass die gesamte Liste der Kurven in der Kurvenauswahl zu sehen ist.

Zeitachse verlängern – Streckt die Zeitachse (X-Achse) so dass mehr Zeiten (und damit mehr Details) dargestellt werden.

Zeitachse verkürzen – Staucht die Zeitachse (X-Achse) so dass mehr Zeiten dargestellt werden.

Strecken – Streckt die vertikale Achse (Y-Achse) um mehr Details darzustellen.

Stauchen – Staucht die vertikale Achse (Y-Achse) um weniger Details darzustellen.

Cursor A – Blendet den Cursor „A" ein oder aus.

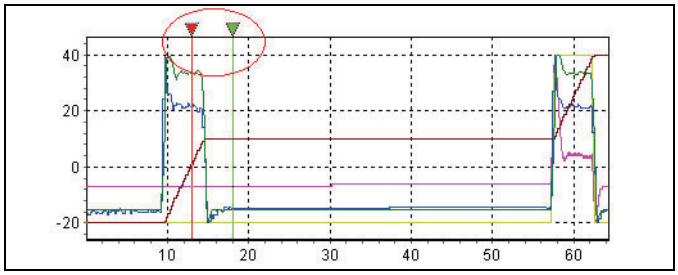
Cursor B - Blendet den Cursor „B" ein oder aus.

Kurven auf Zeit 0 verschieben – Stellt die Zeitachse (X-Achse) auf 0, so dass die Kurven bei Zeit 0 starten. Erneutes Klicken stellt die Startzeit wieder auf den ursprünglichen Wert.

A und B Cursor einsetzen

Mit den beiden „Cursor" A and B des Oszilloskops kann man bequem besondere Zeiten entlang der X-Achse markieren. Sie werden durch *Ansicht* → *Cursor A* und → *Cursor B* oder mit ein- und ausgeblendet.

Die Cursor erscheinen als vertikale Linie mit einem kleinen dreieckigen Griff am oberen Ende (siehe unten). Sie können diese Cursor bewegen, wenn Sie mit der rechten Maustaste in den „Griff" klicken und den Cursor nach links oder rechts ziehen.



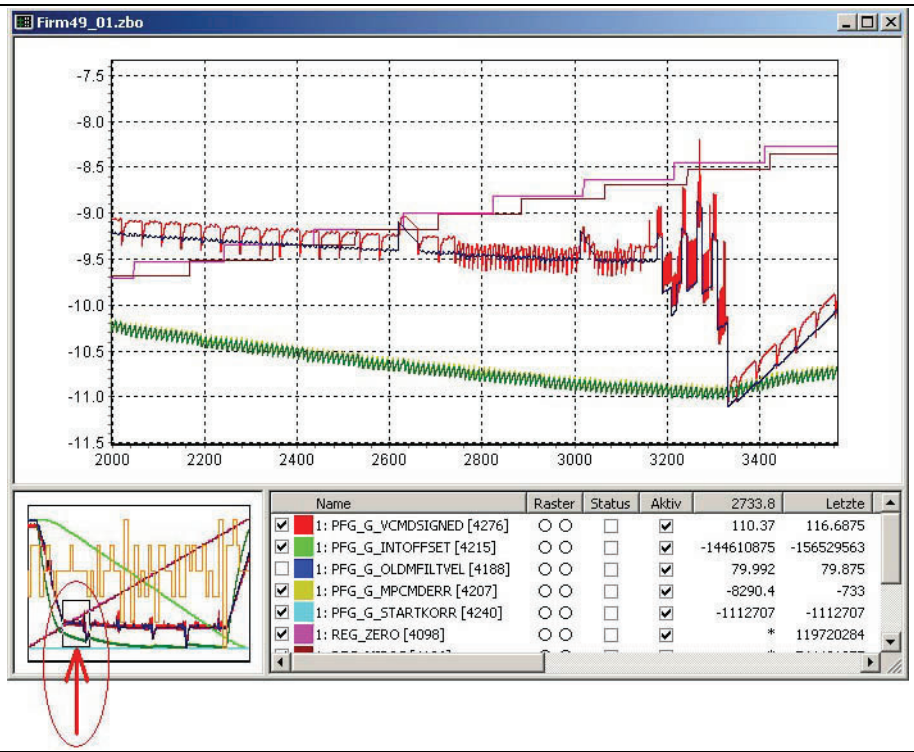
Wenn diese Cursor eingesetzt wurden, wird der Wert dieser Cursor-Position von jeder Kurve in der *Kurvenauswahl* gezeigt, wie im Beispiel zu sehen ist Die Spaltenüberschrift enthält die X-Koordinate (Zeit) der Position von jedem Cursor. Eine weitere Spalte zeigt den Unterschied zwischen den Kurvenwerten der Cursor-Positionen (d.h. der Wert von Cursor „B" minus den Wert von Cursor „A").

A 12.94	B 17.99	B-A 5.048
-7	-7	0
0.0	0.0	0
228	-72	-300
30140	30686	546
360.1	0.0	-360.1

Navigationsfenster

Das *Navigationsfenster* zeigt eine Miniatur des gesamten Diagramms, so dass man die Position des gezoomten Bereichs sehen und relativ zum ganzen Diagramm bewegen kann.

Der gezoomte Bereich des Grafikfensters wird mit einem Rechteck im Navigationsfenster markiert, siehe unten. Falls Cursor A und Cursor B benutzt wurden, werden diese auch im Navigationsfenster gezeigt.



Mit der Maus navigieren

Linke Maustaste (Auswahl) – Wählen Sie den gewünschten Bereich aus, indem Sie mit gedrückter linker Maustaste ein Rechteck ziehen. Dieser Bereich wird dann im Grafikfenster dargestellt.

ACHTUNG!:
Das ausgewählte Rechteck wird so gestreckt, dass es genau in das Fenster passt. Ist das nicht gewünscht ist, drücken und halten Sie die Taste [Strg], bevor Sie die linke Maustaste drücken.

Rechte Maustaste (Verschieben) – Stellen Sie den Mauscursor innerhalb des Rechtecks und verschieben Sie den Ausschnitt mit gedrückter rechter Maustaste. Das Grafikfenster folgt der Bewegung des Auswahlrechtecks.

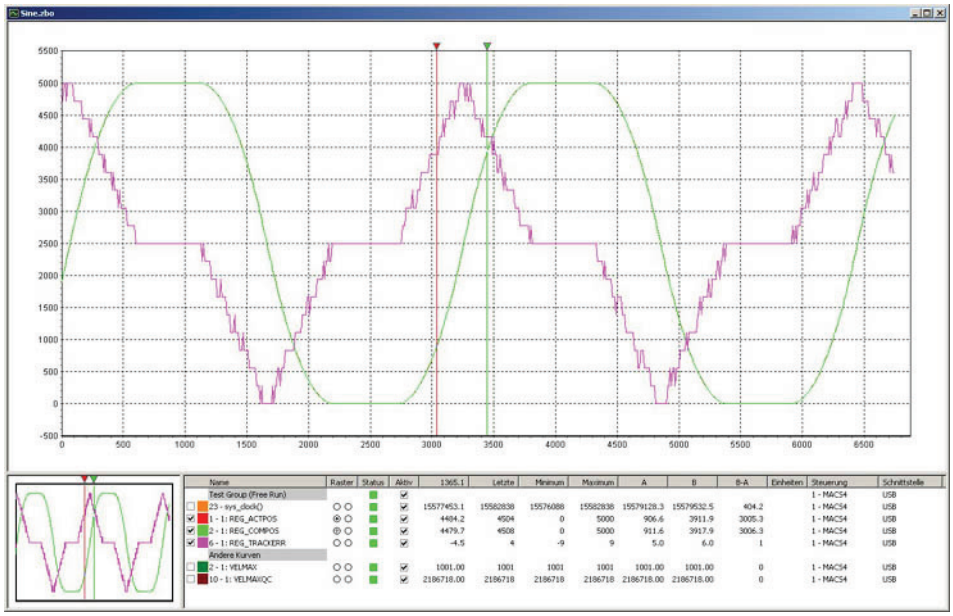
MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

□ Kurvenauswahl im Oszilloskop-Fenster

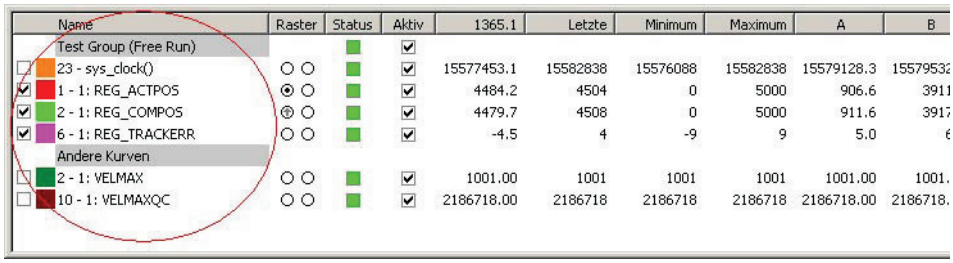
Das Fenster *Kurvenauswahl* enthält eine Liste aller vom Anwender definierten Kurven für die Daten erfasst wurden, zusammen mit den Details jeder Kurve wie Werte, Variablen, Status usw. Alle Eigenschaften jeder individuellen Kurve, wie Darstellung, Ausblenden, Aktivieren, Deaktivieren, Raster, Farbe usw. werden in diesem Fenster gesteuert.

Alle Eigenschaften für die Anzeige, Datenerfassung usw. werden mittels dieser Liste gesetzt. Wenn die Liste zu lang oder zu breit wird, um in das Fenster zu passen, können Sie die Liste scrollen. Um die Spaltenbreite zu ändern, zeigen Sie mit der linken Maustaste auf die Spaltentrennung in der Überschriftenzeile und ziehen die Spalte auf die gewünschte Breite.

Beispiel einer kompletten Liste:



Spalte Name



Die Spalte „Name" enthält drei Teile:

Kontrollkästchen – Wenn aktiviert, wird diese Kurve im Grafik- und Navigationsfenster dargestellt. Andernfalls wird die Kurve ausgeblendet. Zum Ein- und Ausschalten klicken Sie mit der linken Maustaste irgendwo innerhalb der Spalte Name.

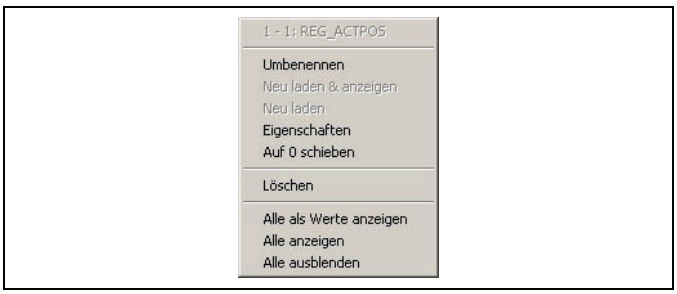
Farbfeld – Es zeigt die Farbe, die für die Kurve im Grafik- und Navigationsfenster benutzt wird. Die Farbe ist eine der Eigenschaften, die der Anwender ändern kann.

Name – Der Name der Kurve kann auch vom Anwender geändert werden. Ein Name auf einem grauen Hintergrund kennzeichnet die folgenden Kurven in der Liste als zusammengehörende Gruppe.
Beachten Sie, dass „Andere Kurven" eine extra Gruppe ist, deren Name nicht geändert werden kann.

Popup-Menü Funktionen für die Spalte Name

Zum Öffnen des folgenden Popup-Menüs klicken Sie mit der rechten Maustaste irgendwo in die Spalte Name:

Der oberste Eintrag ist der Name der Kurve und dient als Referenz. Er bleibt immer grau. Die anderen Befehle sind:



Umbenennen zum Umbenennen einen Kurve.

Neu laden & anzeigen lädt die Kurvendaten erneut von der Steuerung und stellt die Daten im Grafikfenster dar. Beachten Sie, dass dies in Abhängigkeit vom Kurventyp nicht immer möglich ist; dann ist der Befehl grau.

Neu laden lädt die Kurvendaten erneut von der Steuerung.

Beachten Sie, dass dies in Abhängigkeit vom Kurventyp nicht immer möglich ist; in diesem Fall ist der Befehl grau.

Eigenschaften – Dieser Befehl wird benutzt um die Eigenschaften einer Kurve darzustellen und zu modifizieren, siehe Kurveneigenschaften verändern.

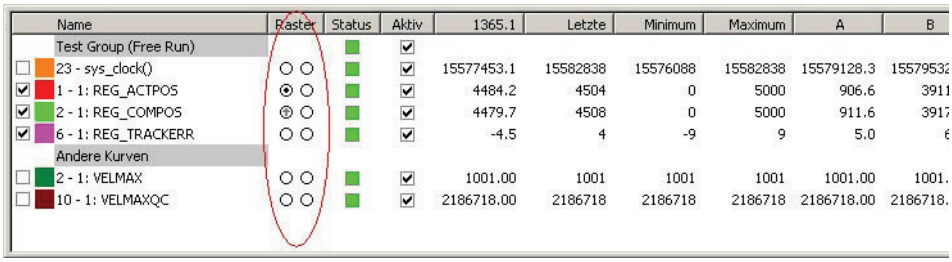
Auf 0 schieben – Die Kurvendaten werden basierend auf der internen Systemzeit der Steuerung erfasst. Da die Tests nicht immer bei 0 starten, kann man damit die Startzeit zwischen 0 und der tatsächlichen Startzeit hin- und herschalten. Dieser Befehl wirkt ähnlich wie das Schaltsymbol → *Kurven auf Zeit 0 verschieben,* außer dass er nur auf die eine ausgewählte Kurve wirkt.

Löschen löscht die ausgewählte Kurve.

Alle als Werte anzeigen – Dieser Befehl ändert die Anzeigeart aller Kurven auf diskrete Werte (ohne Interpolation zwischen den Datenpunkten). Siehe Kurveneigenschaften verändern.

Alle anzeigen und *Alle ausblenden* – Mit diesen Befehlen werden alle Kurven im Diagramm dargestellt bzw. ausgeblendet. So erübrigt sich das einzelne Anklicken.

Spalte Raster



Die Spalte „Raster" hat zwei Spalten mit Optionsfeldern, mit denen Sie festlegen können, auf welcher Seite die Y-Achse beschriftet wird:

Wenn Sie die linke Spalte markieren, erhält die Y-Achse des Grafikfensters auf der linken Seite ein Raster, das der ausgewählten Kurve entspricht. Im Beispiel oben wird das Raster für „REG_ACTPOS" dargestellt. Wenn Sie in die rechte Spalte klicken, wird die Y-Achse des Grafikfensters auf der rechten Seite mit dem Raster der ausgewählten Kurve beschriftet. Das Diagramm kann also auf beiden Seiten zur gleichen Zeit beschriftet werden.

ACHTUNG!:
Beachten Sie, dass alle anderen Kurven mit einem allgemeinen von „oben-nach-unten" Raster dargestellt werden, damit die gesamte Kurve sichtbar ist. Daher ist es wichtig, daran zu denken, dass das Raster der anderen Kurven NICHT dem eingezeichneten Raster der Y-Achse entspricht.

Da es zur gleichen Zeit nur eine Skala auf jeder Seite des Diagramms geben kann, deaktiviert jede neue Auswahl eine frühere.

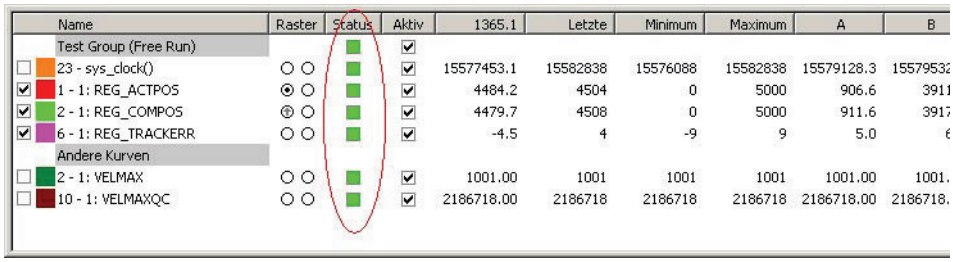
Achsverknüpfung

Ein einigen Fälle ist es wünschenswert, dass zwei oder mehr Kurven die gleiche Y-Achse gemeinsam benutzen. Zum Beispiel, wenn eine Istpositionskurve und eine Sollpositionskurve mit den gleichen Einheiten aufgezeichnet wird und mit der gleichen Skala im Grafikfenster dargestellt werden soll.

Klicken Sie mit der rechten Maustaste in das Optionsfeld, deren Kurve Sie verknüpfen wollen. Ein kleiner grauer Pfeil zeigt an, dass diese Kurve nun die Y-Achse gemeinsam nutzt mit der Kurve, die gerade der Y-Achse zugeordnet ist. Im Beispiel oben nutzt „REG_COMPOS" die Y-Achse gemeinsam mit „REG_ACTPOS".

Erneutes Rechts-Klicken schaltet die *Achsverknüpfung* aus.

Spalte Status



Die Spalte „Status" zeigt den aktuellen Status der Steuerung mit der dazugehörenden Kurve.

Beachten Sie, dass nicht alle Kurven zur gleichen Steuerung gehören müssen; verschiedene Kurven können zu verschiedenen Steuerungen gehören.

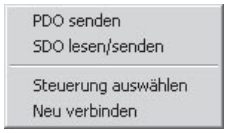
Die folgenden Status-Farben stehen für:

- Die Steuerung führt ein Programm aus.
- Die Steuerung führt kein Programm aus.
- Die Steuerung ist in einem Fehlerzustand.
- Die Steuerung hat keine Verbindung zu APOSS.
- Die Steuerung ist mit dieser Kurve verknüpft, ist aber zurzeit nicht verbunden.

Die Spalte Status ist leer bei jeder Kurve, die nicht explizit mit einer Steuerung verknüpft ist. Zum Beispiel sind „abgeleitete" Kurven eher mit anderen Kurven verknüpft als mit Steuerungen.

Popup-Menü Funktionen für Spalte Status

Rechts-Klicken in der Spalte Status öffnet ein Popup-Menü mit Funktionen für die Steuerung, die mit der ausgewählten Kurve verbunden ist:



PDO senden – sendet ein PDO zur Steuerung.

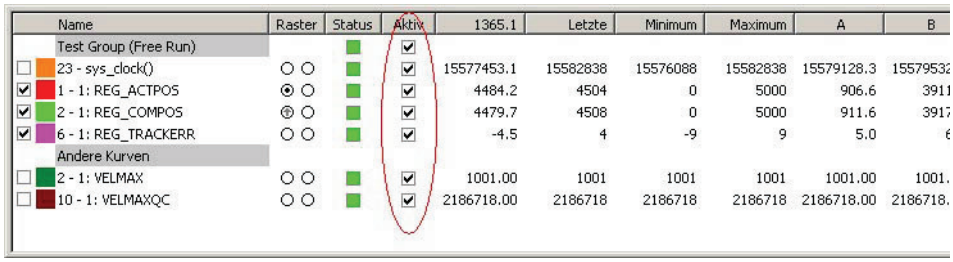
SDO lesen/senden – Die Funktion ermöglicht des Lesen oder Senden einer SDO-Nachricht in die Steuerung.

Steuerung auswählen – Damit kann eine andere Steuerung mit der ausgewählten Kurve verbunden werden. Andere Kurven, die mit dieser oder anderen Steuerungen verbunden sind, sind davon nicht betroffen.

ACHTUNG!:
Beachten Sie, dass in einigen Fällen mehrere Kurven mit der gleichen Steuerung verbunden sein müssen. Zum Beispiel müssen alle Kurven, die von dem gleichen TESTSETP-Array stammen, mit der gleichen Steuerung verbunden sein, da das Array von einer einzigen Steuerung kommt! In diesem Fall wird die gesamte Gruppe der Kurven geändert, wenn eine Kurve geändert wurde. Die Spalten Steuerung und Schnittstelle zeigen die Steuerung mit der die Kurve gerade verbunden ist.

Neu verbinden – Falls die Steuerung gerade nicht verbunden ist oder die Verbindung unterbrochen ist, kann man mit → *Neu verbinden* versuchen, die Steuerung wieder zu verbinden.

Spalte Aktiv



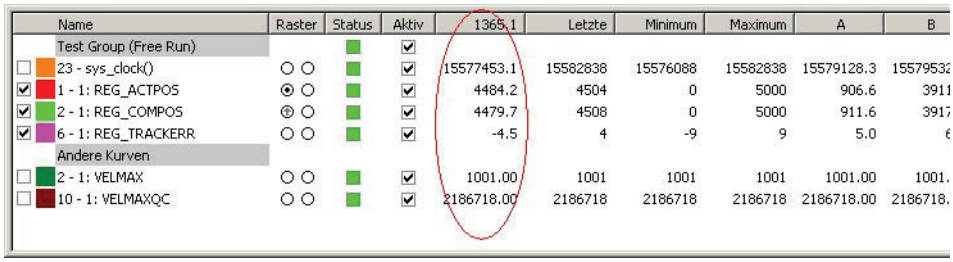
Name	Raster	Status	Aktiv	1365.1	Letzte	Minimum	Maximum	A	B
Test Group (Free Run)									
23 - sys_clock()				15577453.1	15582838	15576088	15582838	15579128.3	1557953
1 - 1: REG_ACTPOS				4484.2	4504	0	5000	906.6	3911
2 - 1: REG_COMPOS				4479.7	4508	0	5000	911.6	3917
6 - 1: REG_TRACKERR				-4.5	4	-9	9	5.0	6
Andere Kurven									
2 - 1: VELMAX				1001.00	1001	1001	1001	1001.00	1001.
10 - 1: VELMAXQC				2186718.00	2186718	2186718	2186718	2186718.00	2186718.

Die Spalte „Aktiv” enthält Kontrollkästchen für jene Kurven, die explizit mit einer Steuerung verbunden sind:

Nur wenn das Kästchen markiert ist, aktiviert das Oszilloskop die Datenerfassung für diese Kurve. Durch Klicken in die Spalte können Sie die Funktion aktivieren/deaktivieren.

Beachten Sie, dass durch das Oszilloskop Daten nur erfasst werden, wenn eine Oszilloskop „Fahrt” gestartet wurde; siehe „Test starten und beenden”.

Spalte Maus-Position



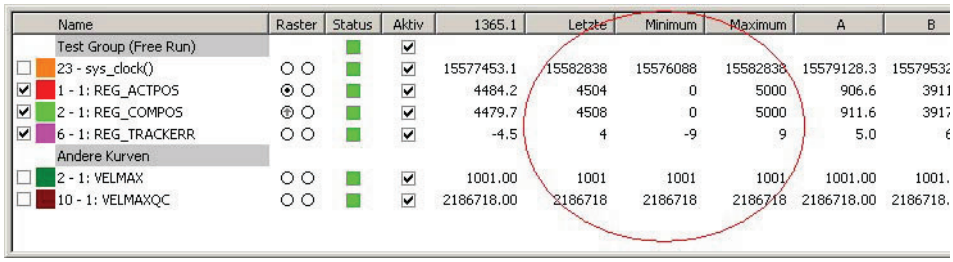
Name	Raster	Status	Aktiv	1365.1	Letzte	Minimum	Maximum	A	B
Test Group (Free Run)									
23 - sys_clock()				15577453.1	15582838	15576088	15582838	15579128.3	1557953
1 - 1: REG_ACTPOS				4484.2	4504	0	5000	906.6	3911
2 - 1: REG_COMPOS				4479.7	4508	0	5000	911.6	3917
6 - 1: REG_TRACKERR				-4.5	4	-9	9	5.0	6
Andere Kurven									
2 - 1: VELMAX				1001.00	1001	1001	1001	1001.00	1001.
10 - 1: VELMAXQC				2186718.00	2186718	2186718	2186718	2186718.00	2186718.

Wenn Sie die Maus über das Grafikfenster bewegen, sehen Sie in dieser Spalte den Wert jeder Kurve auf der Maus-Position.

Die Spaltenüberschrift enthält die X-Koordinate (Zeit) der Maus-Position.

Ein „*” zeigt, dass die Spalte zu eng ist, um den Wert anzuzeigen. Um die Spalte zu verbreitern, zeigen Sie mit der linken Maustaste auf die Spaltentrennung in der Überschriftenzeile und ziehen die Spalte auf die gewünschte Breite.

Spalten Letzter, Minimum und Maximum



Name	Raster	Status	Aktiv	1365.1	Letzte	Minimum	Maximum	A	B
Test Group (Free Run)									
23 - sys_clock()				15577453.1	15582838	15576088	15582838	15579128.3	1557953
1 - 1: REG_ACTPOS				4484.2	4504	0	5000	906.6	3911
2 - 1: REG_COMPOS				4479.7	4508	0	5000	911.6	3917
6 - 1: REG_TRACKERR				-4.5	4	-9	9	5.0	6
Andere Kurven									
2 - 1: VELMAX				1001.00	1001	1001	1001	1001.00	1001.
10 - 1: VELMAXQC				2186718.00	2186718	2186718	2186718	2186718.00	2186718.

Diese Spalten zeigen folgende Werte:

„Letzter” ist der letzte aufgezeichnete Wert jeder Kurve.

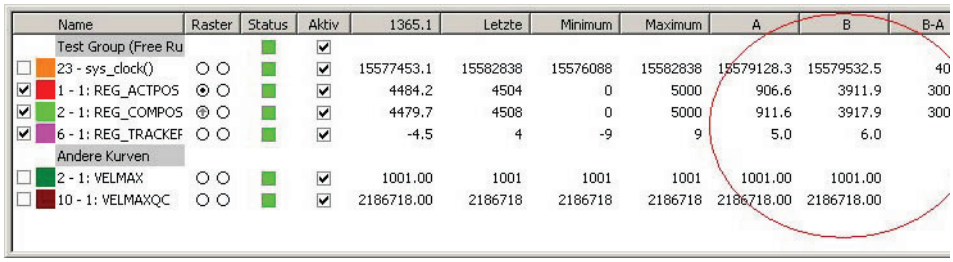
„Minimum” ist der kleinste aufgezeichnete Wert jeder Kurve.

„Maximum” ist der größte aufgezeichnete Wert jeder Kurve.

Ein „*” zeigt, dass die Spalte zu eng ist, um den Wert anzuzeigen. Um die Spalte zu verbreitern, zeigen Sie mit der linken Maustaste auf die Spaltentrennung in der Überschriftzeile und ziehen die Spalte auf die gewünschte Breite.

MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

Spalten Cursor A/B

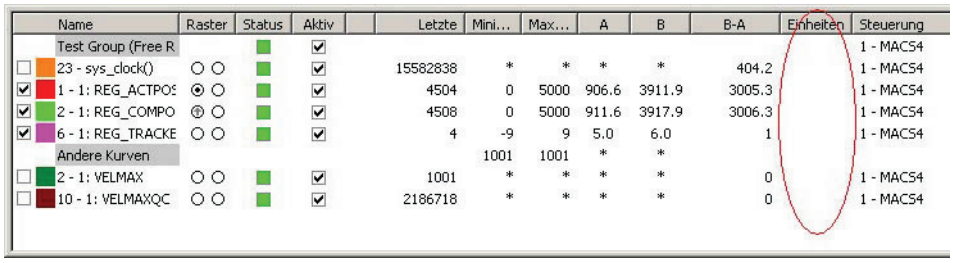


Name	Raster	Status	Aktiv	1365.1	Letzte	Minimum	Maximum	A	B	B-A
Test Group (Free Ru										
23 - sys_clock()				15577453.1	15582838	15576088	15582838	15579128.3	15579532.5	40
1 - 1: REG_ACTPOS				4484.2	4504	0	5000	906.6	3911.9	300
2 - 1: REG_COMPOS				4479.7	4508	0	5000	911.6	3917.9	300
6 - 1: REG_TRACKEF				-4.5	4	-9	9	5.0	6.0	
Andere Kurven										
2 - 1: VELMAX				1001.00	1001	1001	1001	1001.00	1001.00	
10 - 1: VELMAXQC				2186718.00	2186718	2186718	2186718	2186718.00	2186718.00	

Die Spaltenüberschriften enthalten die X-Koordinate (Zeit) jeder Cursor-Position. Wenn die Cursor benutzt wurden, wird der Wert jeder Kurve auf den Cursor-Positionen in den Spalten „A" und „B" angegeben; die Differenz in Spalte „B-A" daneben.

Wenn ein Wert als „*" gezeigt wird, ist die Spalte zu eng für die Darstellung. Um die Spalte zu verbreitern, zeigen Sie mit der linken Maustaste auf die Spaltentrennung in der Überschriftenzeile und ziehen die Spalte auf die gewünschte Breite.

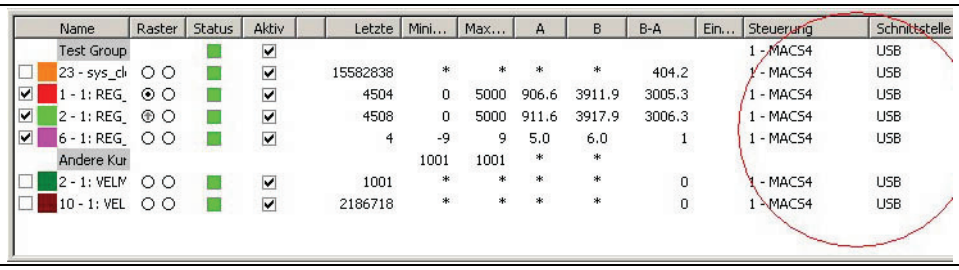
Spalte Einheiten



Name	Raster	Status	Aktiv	Letzte	Mini...	Max...	A	B	B-A	Einheiten	Steuerung
Test Group (Free R											1 - MACS4
23 - sys_clock()				15582838	*	*	*	*	404.2		1 - MACS4
1 - 1: REG_ACTPOS				4504	0	5000	906.6	3911.9	3005.3		1 - MACS4
2 - 1: REG_COMPO				4508	0	5000	911.6	3917.9	3006.3		1 - MACS4
6 - 1: REG_TRACKE				4	-9	9	5.0	6.0	1		1 - MACS4
Andere Kurven					1001	1001	*	*			
2 - 1: VELMAX				1001	*	*	*	*	0		1 - MACS4
10 - 1: VELMAXQC				2186718	*	*	*	*	0		1 - MACS4

Optional kann der Anwender die „Einheit" für jede Kurve bezeichnen; siehe Kurveneigenschaften verändern. Dieser Text wird dann in der Spalte dargestellt; er dient nur der Information und wird nicht vom Oszilloskop benutzt.

Spalten Steuerung und Schnittstelle



Name	Raster	Status	Aktiv	Letzte	Mini...	Max...	A	B	B-A	Ein...	Steuerung	Schnittstelle
Test Group											1 - MACS4	USB
23 - sys_cl				15582838	*	*	*	*	404.2		1 - MACS4	USB
1 - 1: REG_				4504	0	5000	906.6	3911.9	3005.3		1 - MACS4	USB
2 - 1: REG_				4508	0	5000	911.6	3917.9	3006.3		1 - MACS4	USB
6 - 1: REG_				4	-9	9	5.0	6.0	1		1 - MACS4	USB
Andere Kur					1001	1001	*	*				
2 - 1: VELM				1001	*	*	*	*	0		1 - MACS4	USB
10 - 1: VEL				2186718	*	*	*	*	0		1 - MACS4	USB

Wenn eine Steuerung verbunden ist, zeigt die Spalte „Steuerung" die ID-Nummer (im Allgemeinen die CAN ID) und den vom Anwender vergebenen Namen der Steuerung für jede zur Steuerung gehörenden Kurve. Wenn eine Steuerung nicht verbunden ist, wird nur die ID-Nummer gezeigt.

Die Spalte „Schnittstelle" zeigt zu jeder Kurve die dazugehörende Schnittstelle.

Kurveneigenschaften verändern

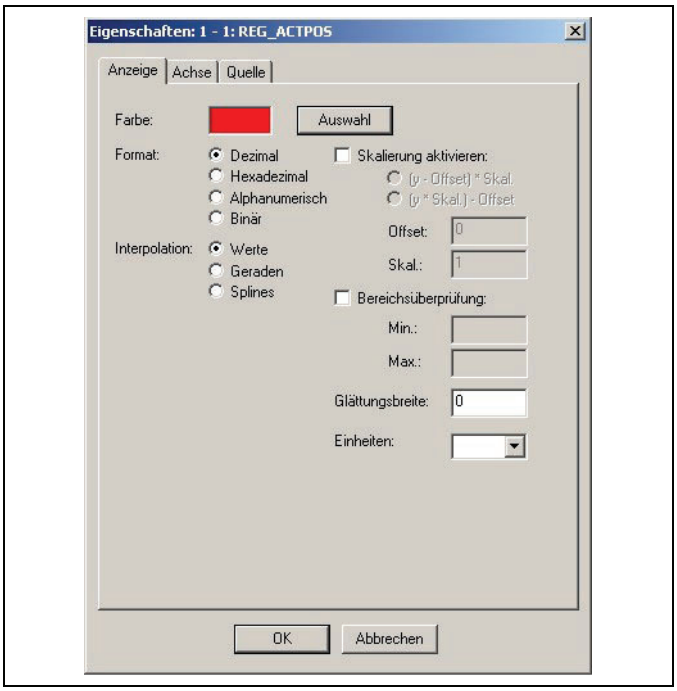
Sie können die Kurveneigenschaften sehen und verändern, wenn Sie in der Spalte „Name" mit der rechten Maustaste auf den Namen der Kurve und im Popup → *Eigenschaften* auswählen. Im darauf folgenden Dialogfeld finden Sie die Registerkarten:

Display zeigt die verschiedenen Attribute, die die Darstellung der Kurve im Diagramm und die Anzeige der Werte in der Liste betreffen. Diese Registerkarte ist nicht vorhanden, wenn eine nicht grafisch darstellbare Kurve ausgewählt wurde, zum Beispiel eine Gruppenüberschrift.

Achse zeigt die verschiedenen Attribute, die die Y-Achse einer Kurve betreffen. Diese Registerkarte ist auch nicht vorhanden, wenn nicht grafisch darstellbare Kurven ausgewählt sind, z.B. Gruppenüberschriften.

Quelle zeigt Details wie und wo die Daten erfasst wurden.

Kurveneigenschaften: Anzeige



Farbe

Mit → *Auswahl* können Sie die Farbe der Kurve im Grafikfenster ändern.

Format

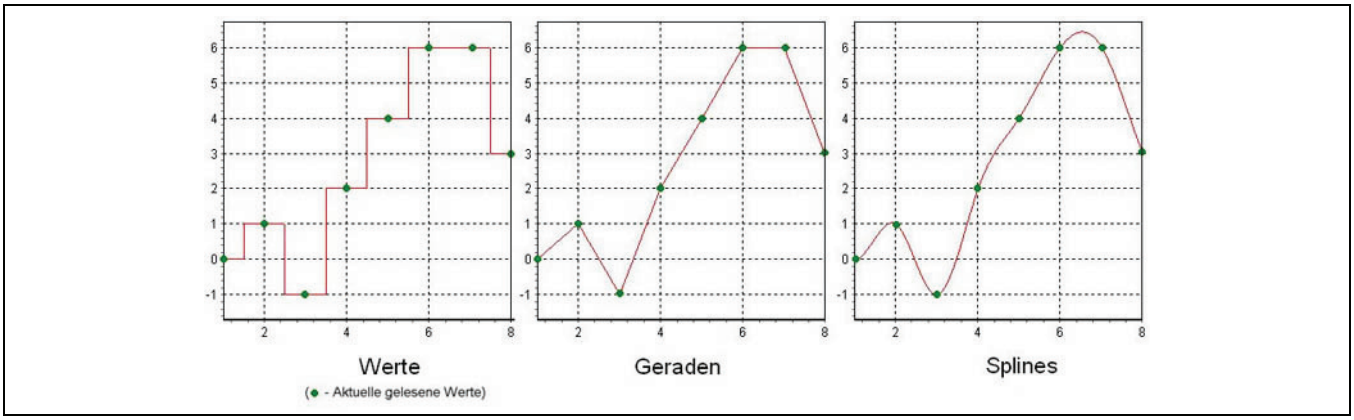
Bestimmen Sie, wie die Kurvenwerte (d.h. die Mausposition, Letzter, Minimum, Maximum usw.) in der Kurvenauswahl dargestellt werden: Dezimal, Hexadezimal, Alphanumerisch.

ACHTUNG!:
Werte sollten nicht alphanumerisch dargestellt werden, wenn nicht die Steuerung tatsächlich solche für diese Kurve erzeugt. Falls nicht, werden im Allgemeinen Leerzeichen dargestellt.

Beachten Sie auch, dass sich die Einstellung der *Interpolation* auf die Darstellung der Kurve auswirkt.

Interpolation

Für eine Kurve wird eine Serie von diskreten Abtastpunkten aufgezeichnet. Die Einstellung legt fest, wie das Oszilloskop die Werte zwischen den tatsächlichen Punkten interpoliert.



Normalerweise hängt die Wahl der Interpolation vom aufgezeichneten Wertetyp ab. Zum Beispiel wird jede Art von „Status"-Werten normalerweise als *Werte* dargestellt, denn naturgemäß sind Statuswerte diskret (d.h. ein Wert nach dem anderen und nie „dazwischen"). Abtastpunkte aus kontinuierlichen Werten wie Position, Geschwindigkeit usw. werden normalerweise mit *Geraden* verbunden.

MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

Eine *Spline*-Interpolation sollte nur in Ausnahmefällen benutzt werden, und zwar wenn eine Kurve nur eine sehr begrenzte Anzahl von Abtastpunkten hat und es ausdrücklich beabsichtigt ist, sie mit einem Spline darzustellen (z.B. CAM-Editor-Kurven). Im Allgemeinen sollte *Spline* nicht für normale zeitabhängige Abtastpunkte benutzt werden.

ACHTUNG!:
Diese Einstellung kann die *Anzeige*-Einstellungen für einige Spalten in der Kurvenauswahl überschreiben. Falls *Werte* ausgewählt ist, dann werden die Werte in der Kurvenauswahl entsprechend der *Anzeige*-Einstellung dargestellt. Wenn jedoch *Geraden* oder *Splines* ausgewählt sind, dann werden die Werte für die Spalte Mausposition und alle drei Cursor-Spalten interpoliert und *Dezimal* dargestellt, ungeachtet der *Anzeige*-Einstellung.

Skalierung aktivieren

Der Einfachheit halber können die von der Steuerung aufgezeichneten Werte im Oszilloskop skaliert werden. Zum Beispiel könnte der Anwender wollen, dass ein Wert, der in Millimeter erfasst wurde in Zentimeter dargestellt wird. Dies betrifft beides, die Kurve im Grafikfenster und die dargestellten Werte in der Kurvenauswahl. Zwei Alternativen des Skalierungsalgorithmus stehen zur Verfügung:
„(y-offset)*scale" und „(y*scale)-offset". Normalerweise wird der eine oder der andere für die spezielle Anwendung passender sein. Im Beispiel Millimeter-zu-Zentimeter würde der *Offset* auf 0 und die *Skalierung* auf 0,1 gesetzt werden.

Bereichsüberprüfung

Oft ist es nützlich zu wissen, ob ein Wert im „Aus" ist. Dies kann man erfahren, indem man die Bereichsüberprüfung einschaltet und die unterste und oberste Begrenzung des zugelassenen Bereichs festlegt. Jeder Wert, der nicht innerhalb des zugelassenen Bereichs ist, wird dann in der Kurvenauswahl mit einem roten Hintergrund markiert.

Beachten Sie: Wenn nur eine Begrenzung zutreffend ist, kann die andere Begrenzung leer bleiben. Hier sehen Sie ein Beispiel, bei dem der Maximalwert für die zweite Kurve auf 4900 gesetzt ist:

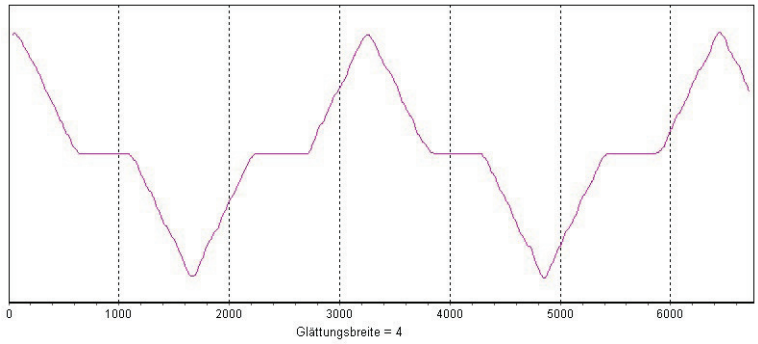
1204.9	Letzte	Minimum	Maximum	A 3769
15577292.9	15582838	15576088	15582838	15579857.5
4934.1	4504	0	5000	4987.6
4932.6	4508	0	5000	4988.6
-1.5	4	-9	9	1.0

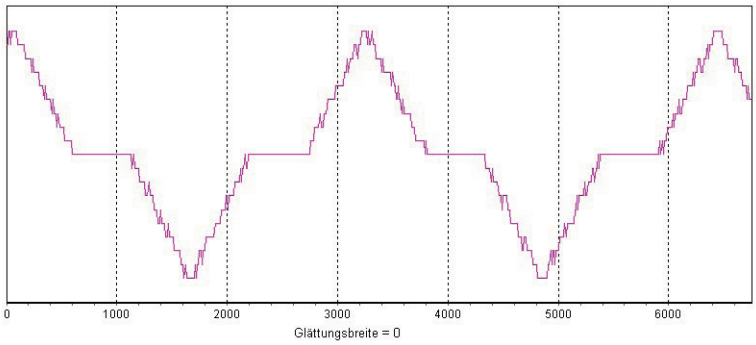
Glättungsbreite

In einigen Fällen können gemessene Werte eine gewisse Anzahl Schwankungen (Jitter) enthalten (d.h. Schwingungen die schnell, ein wenig zufällig und im Allgemeinen klein sind). Dies kann sich im Diagramm störend auswirken (siehe Beispiel unten). Das Setzen einer *Glättungsbreite* reduziert das Auftreten dieser Schwankungen durch das Mitteln der Werte der benachbarten Abtastpunkte.

Beachten Sie, dass dies nur ein visueller Kunstgriff im Oszilloskop ist; es tangiert NICHT irgendeinen tatsächlichen Wert in der Steuerung.

Die Breite legt die Anzahl der Punkte auf beiden Seiten von jedem Punkt fest, der für das Mitteln benutzt wird. Zum Beispiel bedeutet die Glättungsbreite = 2 je 2 Punkte auf der linken und auf der rechten Seite für insgesamt 5 Punkte in jedem gemittelten Punkt.

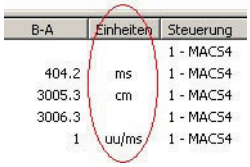




ACHTUNG!:
Mit Sorgfalt muss man vermeiden, die Glättungsbreite höher als notwendig zu setzen. Beispiel: Wenn eine Glättungsbreite = 2 zufrieden stellende Ergebnisse liefert, dann benutzen Sie nicht eine Breite = 4. Wird die Breite höher und höher gesetzt, beginnen die „wirklichen" Informationen der Kurve „weggemittelt" zu werden und gehen verloren.

Einheiten

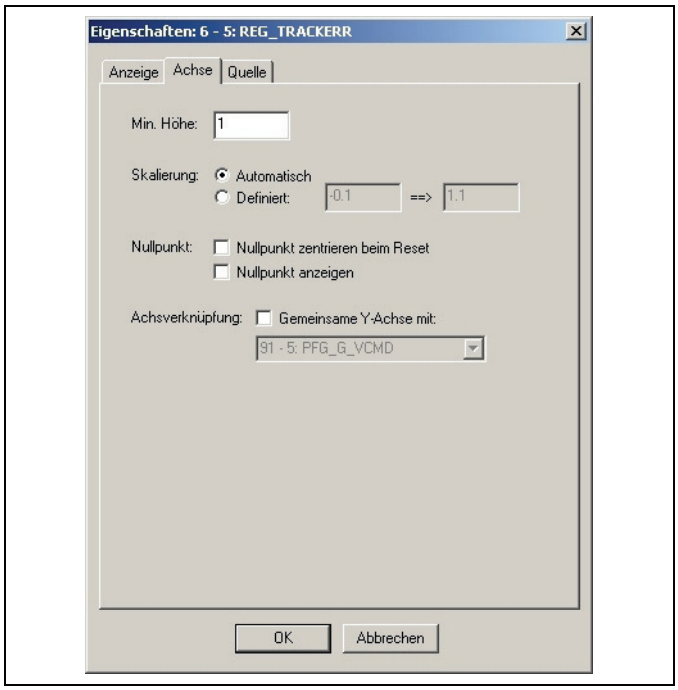
Optional kann der Anwender die „Einheit" für jede Kurve selbst bezeichnen. Dazu kann er entweder direkt einen Text eingeben oder im Auswahlmenü einen auswählen. Der Test erscheint in der Spalte „Einheiten" in der Kurvenauswahl.



B-A	Einheiten	Steuerung
		1 - MACS4
404.2	ms	1 - MACS4
3005.3	cm	1 - MACS4
3006.3		1 - MACS4
1	uu/ms	1 - MACS4

MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

Kurveneigenschaften: Achse



Min. Höhe

Die Y-Achse des Diagramms wird normalerweise so skaliert, dass die gesamte Kurve gezeichnet werden kann. Wenn aber die Kurve einen konstanten Wert hat oder Werte mit nur sehr kleinen Änderungen, kann sie nicht so skaliert werden, dass die Achsenbeschriftung leicht lesbar ist. In diesen Fällen wird dieser Wert als *Mindest Höhe* für die Y-Achse benutzt (d.h. als Größenbereich für die Y-Achse).

Skalierung

Wenn die Skalierung auf *Automatisch* gesetzt ist, wird die Y-Achse automatisch so skaliert, dass die gesamte Kurve im Diagramm mit maximal möglichen Details gezeichnet werden kann. Der Bereich der Y-Achse wird etwa 10 % höher als der maximale Wert der Kurve und 10 % niedriger als die minimale Wert der Kurve gesetzt (und auf die *Mindest Höhe* wie oben beschrieben angepasst).

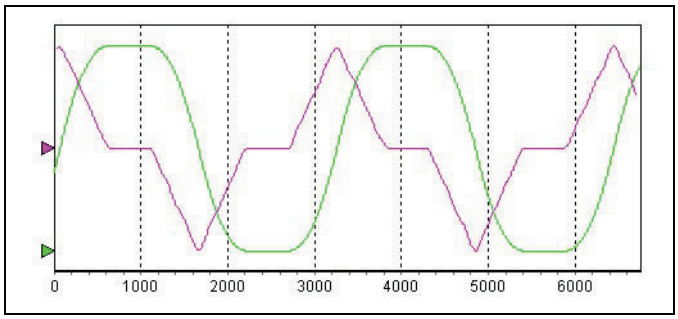
Ist die Skalierung auf Benutzer *definiert* gesetzt, muss der Anwender den Bereich der Y-Achse in den nebenstehenden beiden Feldern festlegen. Beachten Sie, dass dies automatisch angepasst wird, wenn *Nullpunkt zentrieren beim Reset* aktiviert ist (siehe unten).

Diese Einstellung der Y-Achse (d.h. entweder Automatisch oder definiert) wird bei *Zoom zurücksetzen* benutzt. Beachten Sie: Falls die Skalierung geändert wurde, dann hat die neue Skalierung so lange keine Auswirkung, bis der Anwender erneut *Zoom zurücksetzen* benutzt**.**

Nullpunkt

Wenn *Nullpunkt zentrieren beim Reset* aktiviert ist, wird der Bereich der Y-Achse für die Kurve so angepasst, so dass er symmetrisch über 0 ist. Die „0" Linie wird also durch die Mitte des Diagramms gezeichnet.

Wenn *Nullpunkt anzeigen* aktiviert ist, markiert ein kleines Dreieck auf der linken Seite des Diagramms die Position der Nulllinie. Es hat die gleiche Farbe wie die Kurve, so dass die Markierungen gut voneinander unterschieden werden können, falls solche für mehrere Kurven benutzt wurden. Sehen Sie hier zwei solcher Null-Markierungen:



Achsverknüpfung

Ein einigen Fälle ist es wünschenswert, dass zwei oder mehr Kurven die gleiche Y-Achse „gemeinsam" nutzen. Wenn zum Beispiel eine Istpositionskurve und eine Sollpositionskurve mit den gleichen Einheiten aufgezeichnet wird und mit der gleichen Skala im Grafikfenster dargestellt werden soll.

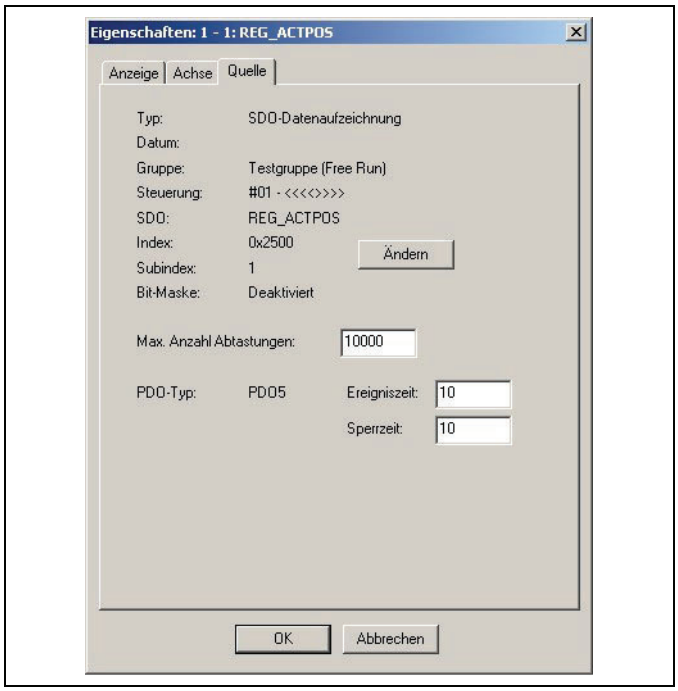
Aktivieren Sie *Achsverknüpfung* und wählen Sie die „andere" Kurve unter den angebotenen aus. Die Kurve wird dann die gleiche Y-Achse nutzen wie die ausgewählte „andere" Kurve.

Eine *Achsverknüpfung* kann auch mit den Optionsfeldern in der Spalte „Raster" der Kurvenauswahl bestimmt werden.

ACHTUNG!:
Ebenso wie eine explizite Verknüpfung (z.B. Kurven B und C teilen die Y-Achse mit der Kurve A), können auch implizite Ketten erzeugt werden (z.B. Kurve C teilt die Y-Achse mit Kurve B und Kurve B teilt die Y-Achse mit Kurve A … vorausgesetzt dass die Kurve C auch die Y-Achse der Kurve A teilt). Eine Änderung der Achsverknüpfung kann eine solche implizite Kette erzeugen oder brechen.

Kurveneigenschaften: Quelle

Die Registerkarte Quelle zeigt verschiedene Informationen über den Typ der Kurve und woher die Kurvendaten kommen. Der genaue Inhalt dieses Dialogfensters variiert in Abhängigkeit vom Kurventyp. In einigen Fällen können auch die Attribute, die sich auf die Datensammlung beziehen, verändert werden.



MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

□ Oszilloskop starten

Das Starten des Oszilloskops erfordert folgende Schritte:

1. APOSS Programm in einem APOSS Fenster öffnen
2. Steuerung verbinden
3. Wenn das TESTSETP Oszilloskop benutzt wird:
   Ein APOSS Programm ausführen, das online Daten durch eine TESTSETP / TESTSETINDEX Konfiguration aufzeichnet.
4. Das ausgewählte Oszilloskop starten

APOSS Programm öffnen

Beginnen Sie mit dem Öffnen eines APOSS Programms (.m) entsprechend dem Programm, das gerade in der Steuerung geladen oder ausgeführt wird. Dadurch wird ein normales APOSS-Fenster geöffnet. Es ist allerdings nicht zwingend notwendig, das entsprechende Programm zu öffnen. Jedes beliebige Programm oder auch ein neues Programm wäre möglich. Zum Beispiel könnte das eigentliche Programm gerade nicht greifbar sein. Arbeitet man allerdings mit dem richtigen Programm, wird die Diagnose leichter sein, weil man sich auf das Programm beziehen kann, um die Testergebnisse zu verstehen.

Steuerung verbinden

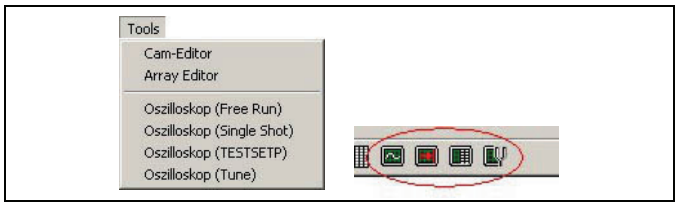
Verbinden Sie im APOSS-Fenster die entsprechende Steuerung mit einer der möglichen Methoden (z.B. [Esc]-Taste, um mit der Default-Steuerung zu verbinden, *Entwicklung → Steuerung auswählen,* oder , etc.).

Wenn das TESTSETP Oszilloskop benutzt wird

Führen Sie ein APOSS Programm aus, das online Daten durch eine TESTSETP / TESTSETINDEX Konfiguration aufzeichnet. Dieses Programm kann direkt mit *Entwicklung → Ausführen [F5]* gestartet werden.

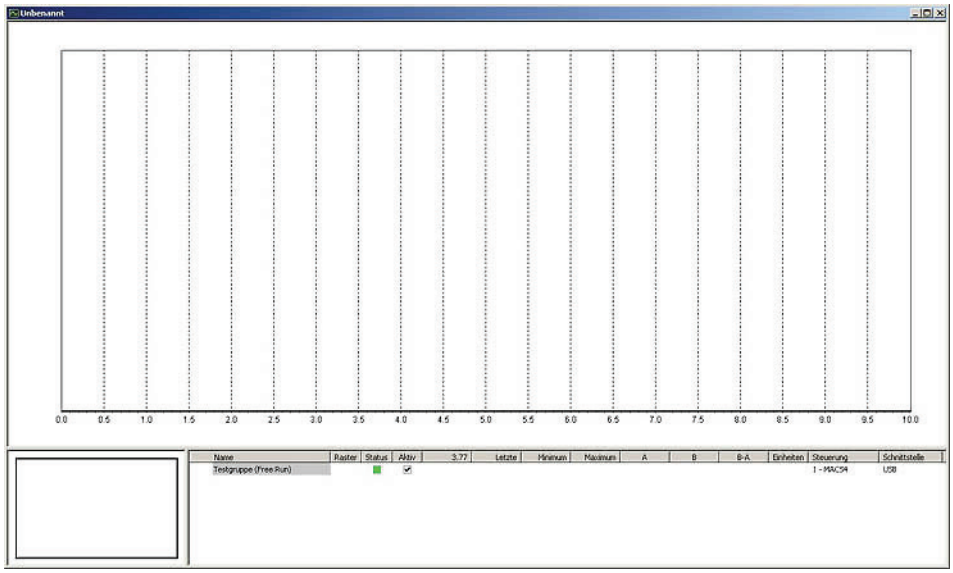
Oszilloskop starten

Jede der Oszilloskop-Versionen kann entweder im Menü ***Tools*** oder durch Klicken auf eines der Oszilloskop-Symbole in der Symbolleiste gestartet werden. Das entsprechende Oszilloskop wird auch gestartet, wenn eine zuvor gesicherte Oszilloskop-Datei wieder geöffnet wird.

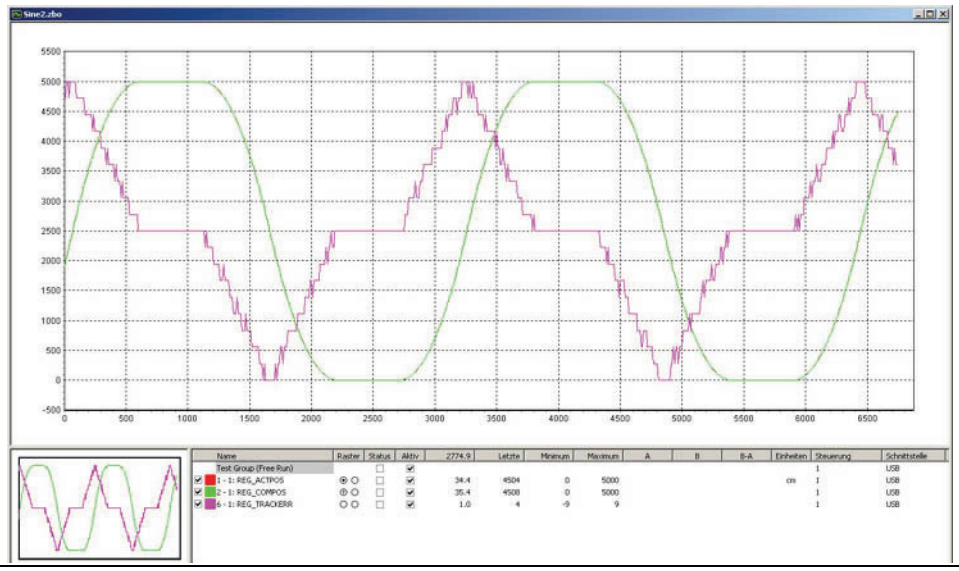


Das Oszilloskop wird automatisch mit der Steuerung verbunden, die im APOSS-Fenster verbunden war. Beachten Sie, wenn nicht schon im APOSS-Fenster eine Steuerung verbunden war, wird beim Starten des Oszilloskops eine Default-Steuerung verbunden.

Es wird ein Oszilloskop-Fenster ähnlich dem Folgenden geöffnet. Ganz rechts in der Kurvenauswahl sehen Sie die Steuerung, die verbunden ist. Das grüne Feld in der Spalte „Status" in diesem Beispiel zeigt, dass die Steuerung, die verbunden ist, ein Programm ausführt und nicht im Fehlerzustand ist.



Das Oszilloskop kann natürlich auch durch Öffnen einer vorhandenen Oszilloskop-Datei geöffnet werden. Wählen Sie einfach die Oszilloskop-Datei „.zbo" aus. Es wird ein Oszilloskop Fenster ähnlich dem Folgenden geöffnet:

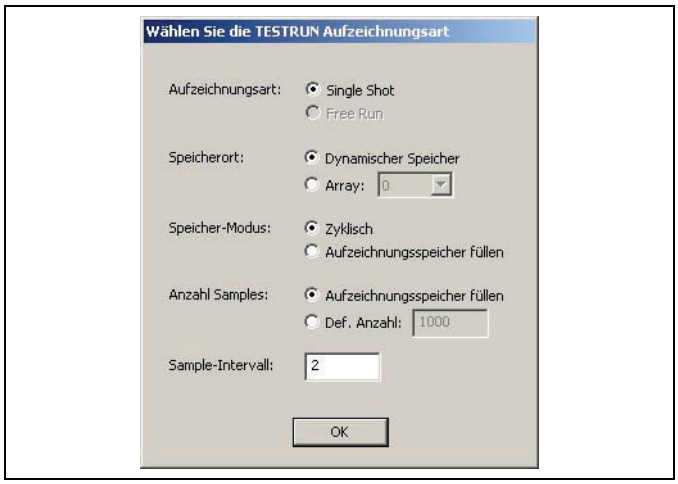


Die weißen Felder in der Spalte „Status" zeigen, dass die Steuerungen nicht verbunden sind. Aber ganz rechts in der Kurvenauswahl sieht man die ID der Steuerung und die Schnittstelle.

Wenn Oszilloskop-Dateien geöffnet sind, werden nie Steuerungen automatisch verbunden. Dies ermöglicht es Oszilloskop-Dateien zu öffnen und offline zu analysieren. Die Steuerungen können wieder verbunden werden, entweder mit den Schaltflächen der Symbolleiste oder mit der rechten Maustaste in der Kurven-auswahl (siehe Spalte „Status" in Kurvenauswahl).

Weiter, wenn Single Shot Oszilloskop

Wenn das *Single Shot Oszilloskop* startet, wird automatisch eine Gruppe in der gleichen Weise bei beim Free Run Oszilloskop hinzugefügt. Hier muss der Anwender jedoch einige Eigenschaften des TESTSETP-Arrays, das zum Aufzeichnen der Kurvendaten benutzt wird, festlegen. Dazu wird das folgende Fenster dargestellt:



Aufzeichnungsart

Single Shot ist bereits ausgewählt.

Speicherort

Bestimmen Sie wo das TESTSETP-Array gespeichert werden soll:

Wenn *Dynamischer Speicher* ausgewählt wird, wird das TESTSETP-Array im verfügbaren freien Speicher in der Steuerung gespeichert. Beachten Sie, dass die Menge des freien Speichers von der Größe und Anzahl der Programme und anderen Daten, die schon in der Steuerung gespeichert sind, abhängt.

Bei *Array* wird das TESTSETP-Array im speziellen benutzerdefinierten Array in der Steuerung gespeichert. Dieses Array muss zuvor vom Anwendungsprogramm des Benutzers definiert worden sein.

Speicher-Modus

Hiermit wird festgelegt, was passieren soll, wenn das TESTSETP-Array voll wird.

Bei *Zyklisch* überschreiben die neuen Daten die alten und die alten Daten gehen verloren. Das TESTSETP-Array enthält also nur den letzten Satz der aufgezeichneten Punkte.

Bei *Stoppen wenn voll* stoppt die Aufzeichnung sobald das TESTSETP-Array voll ist und die neuen Daten gehen verloren.

Anzahl Samples

Legt die maximale Anzahl der Abtastpunkte fest, die im TESTSETP-Array gespeichert werden.

Bei *Aufzeichnungsspeicher füllen* werden so viele Abtastpunkt wie möglich im Array gespeichert.

Bei *Definiere Anzahl* werden nicht mehr als diese definierte Anzahl von Sätzen mit den Abtastpunkten (d.h. ein Punkt je Kurve) gespeichert.

Sample-Intervall

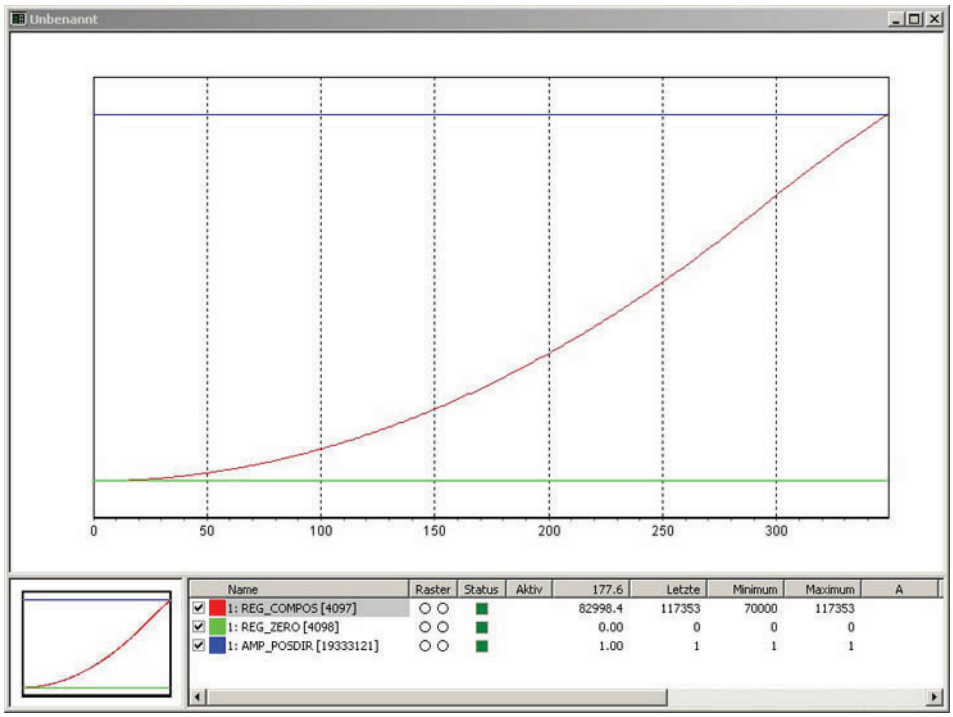
Legt das Zeitintervall in Millisekunden zwischen den Abtastpunkten fest.

Die Gruppe wird unter dem Namen „Testgruppe (Single Shot)" in die Liste der Kurvenauswahl eingereiht.

Weiter, wenn TESTSETP Oszilloskop

Wenn das *TESTSETP Oszilloskop* startet, wird automatisch versucht die TESTSETP-Arrays in der Steuerung zu lokalisieren. Wird nur ein TESTSETP-Array gefunden, wird das Array automatisch hochgeladen und dargestellt. Wenn mehr als ein TESTSETP-Array gefunden wird, wird der Anwender aufgefordert, das Array welches hochgeladen werden soll zu bestimmen. Die anderen TESTSETP-Arrays können hochgeladen werden, sobald das Oszilloskop startet.

Das TESTSETP Oszilloskop Fenster sieht dann etwa so aus:



Dieses Beispiel wurde mit folgendem APOSS Befehl erzeugt: `TESTSETP 1 4097 4098 0x01270001 test`

Bitte beachten Sie Folgendes bezüglich der Kurvenauswahl:

- Das TESTSETP-Array definiert drei Werte, so dass durch das Oszilloskop automatisch drei Kurven hinzugefügt werden. Das Oszilloskop fügt immer eine Kurve für jede Kurve, die im Array definiert ist, hinzu.
- Der Name der ersten Kurve im Satz dieser TESTSETP-Kurven hat immer einen grauen Hintergrund. Dadurch können Sie den Satz der TESTSETP-Kurven identifizieren, wenn mehrere TESTSETP-Arrays im Oszilloskop hinzugefügt wurden.
- Das Oszilloskop versucht die Kurvennamen von den Indices, die im Befehl TESTSETP spezifiziert sind, abzuleiten. Diese Namen haben das Format „Axis: Name [Index]" wobei „Index" der Index aus dem Befehl ist.
- Das Oszilloskop kann das TESTSETP-Array in fast jedem Status der Steuerung hoch laden. Die Steuerung muss dazu kein Testprogramm ausführen. Daher wird die Spalte Status nicht zwangsweise hellgrün sein.
- Da das Oszilloskop nicht kontrolliert, welche Werte aufgezeichnet werden oder die Aufzeichnung startet oder stoppt, ist die Spalte Aktiv bei TESTSETP-Kurven leer.
- Auch die ▶ *Start* Schaltfläche ist beim TESTSETP Oszilloskop deaktiviert ist, weil die Aufzeichnung nicht durch das Oszilloskop gestartet und gestoppt wird.

Oszilloskop Dateien speichern oder öffnen

Jederzeit kann der komplette Satz der Kurven, deren Daten und Eigenschaften und alle Oszilloskop-Einstellungen als „Oszilloskop-Datei" mit *Datei* → *Speichern* und → *Speichern als* oder mit auf dem PC gesichert werden. Oszilloskop-Dateien haben immer die „.zbo" Dateierweiterung. Diese werden im Windows-Order mit dem Symbol dargestellt.

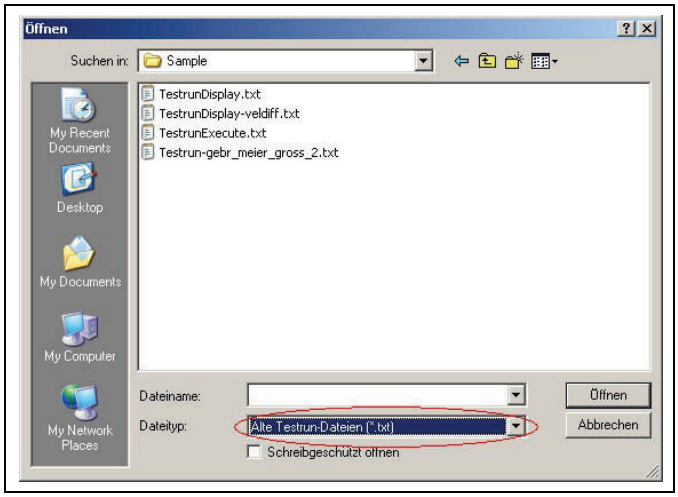
Jeder der Standardmethoden zum Öffnen von Dateien (d.h. *Datei* → *Öffnen* oder , Doppelklicken auf die Datei, die Datei in die APOSS-Anwendung oder in ein geöffnetes APOSS-Fenster „ziehen" usw.) kann benutzt werden um eine Oszilloskop-Datei wieder zu öffnen. Dann wird das Oszilloskop gestartet und die Kurven werden wieder dargestellt.

Das Oszilloskop selbst hat auch einen speziellen Befehl *Datei* → *Speichern ohne Daten*. Dieser kann benutzt werden um die vollständige Oszilloskop-Konfiguration (d.h. alle Kurven-Definitionen und –Einstellungen usw.) zu speichern, ohne jedoch die aktuell aufgenommenen Kurvendaten. So können Sie eine „Vorlage" für die Oszilloskop-Einstellungen erzeugen, die Sie später öffnen und wieder benutzen können.

Alte Testfahrt-Dateien lesen

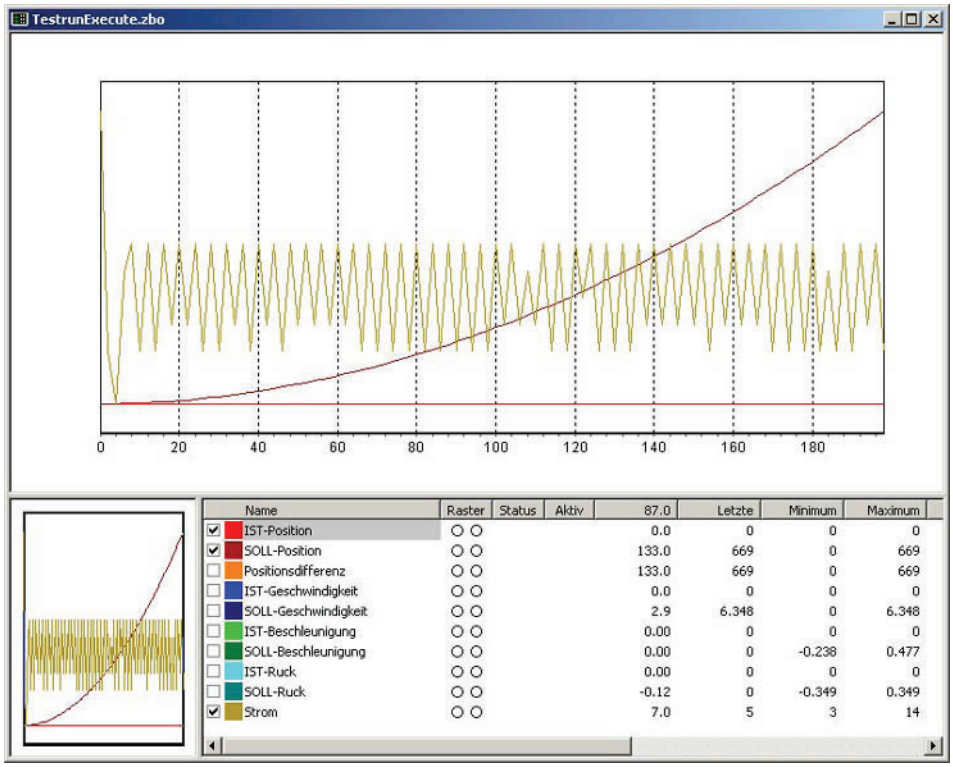
Die *Testfahrt*-Funktion in früheren APOSS Versionen ermöglichte es, die Ergebnisse von → *Testfahrt ausführen* und → *Aufzeichnung anzeigen* in „.txt" Dateien zu speichern. Diese „alten" txt-Dateien können von den neuen APOSS-Versionen gelesen und dargestellt werden.

Öffnen Sie dazu die txt-Datei. Um dies einfacher zu machen, können Sie dazu den *Dateityp* im Öffnen-Dialogfeld auf *„Alte Testrun-Dateien*" setzen, wie unten gezeigt.



Wenn diese Datei geöffnet ist, wird ein TESTSETP Oszilloskop Fenster geöffnet um die Kurven darzustellen. Siehe Beispiel unten. Die Datei kann dann als normale Oszilloskop-Datei „.zbo" gespeichert werden, falls gewünscht.

ACHTUNG!:
Wenn APOSS nicht erkennt, dass die Datei alte Testfahrt-Daten enthält, wird die Datei eher in einem normalen APOSS-Fenster geöffnet, als in einem Oszilloskop-Fenster.



Alte „.txt” Dateien enthalten nicht genug Informationen über die ursprüngliche Steuerung, um die Steuerung zuverlässig zu verbinden. Daher sind die Spalten Status, Steuerung und Schnittstelle leer. Informationen über die ursprünglichen Testparameter können jedoch in *Eigenschaften* → *Quelle* eingesehen werden.

MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

☐ Oszilloskop-Kurven hinzufügen

Kurven werden mit zum Oszilloskop hinzugefügt. Welcher Typ hinzugefügt werden kann, hängt von der Oszilloskop-Version ab, die eingesetzt wird.

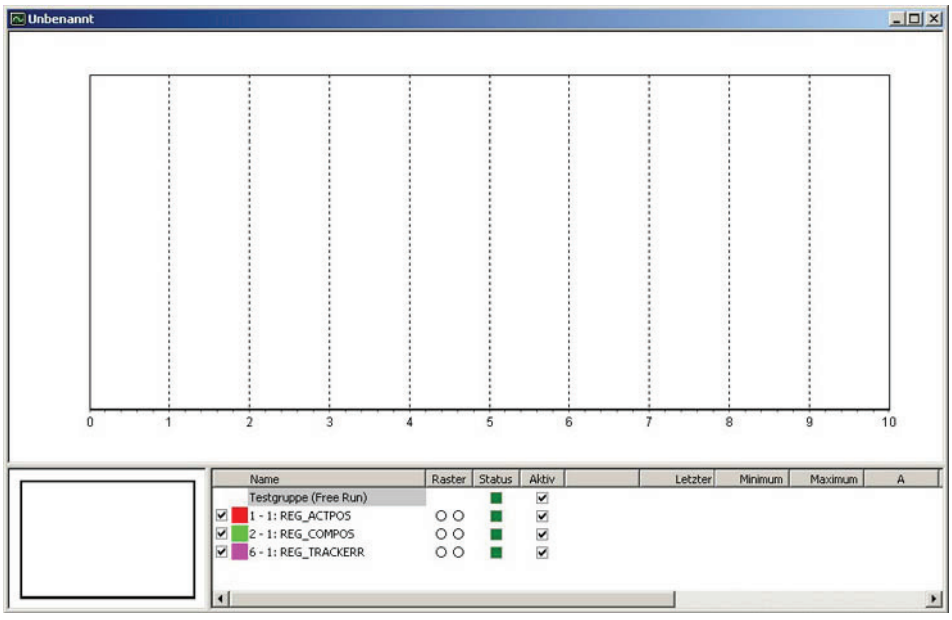
Das Oszilloskop benutzt „Gruppen" um die Kurvensätze zu managen. Wenn ein Test gestartet wurde, konfiguriert das Oszilloskop alle Kurven in einer Gruppe in ein PDO-Mapping in der Steuerung. Jedes Mal, wenn die Steuerung eine PDO-Nachricht generiert, wird der gesamte Satz der Kurvenwerte von der Steuerung in das Oszilloskop übertragen und dargestellt.

Sobald das Oszilloskop gestartet ist, wird eine einzelne Gruppe automatisch hinzugefügt, z.B. beim Single Shot Oszilloskop eine „Test Gruppe (Free Run)". Sie können dann beliebig Kurven zu dieser Gruppe hinzufügen. Die folgenden Kurvenarten könnten von besonderem Interesse sein, um Probleme zu erkennen:

- Achsprozessdaten (SDO 0x2500). Zum Beispiel Achsposition, Geschwindigkeit, Schleppfehler usw.
- Systemprozessdaten (SDO 0x2202). Zum Beispiel Status der digitalen I/O Pins.
- Benutzerparameter (SDO 0x2201) … falls von der Anwendung benutzt.
- Programmvariablen. Wenn das richtige APOSS Programm bekannt ist, kann jede Variable, die im Programm benutzt wird, aufgezeichnet werden.

Eine Gruppe ist normalerweise für die meisten Zwecke ausreichend. Wenn jedoch der Benutzer-Modus in *Einstellungen* → *Optionen* auf „Experte" gesetzt ist, unterstützt das Oszilloskop auch mehrere Gruppen. Mehrere Gruppen wären notwendig, wenn mehrere Steuerungen simultan getestet werden. Da alle Kurven in einer Gruppe mit der gleichen Steuerung verbunden sein müssen, müssen – falls zwei Steuerungen genutzt werden – auch zwei Gruppen definiert werden, für jede Steuerung eine Gruppe.

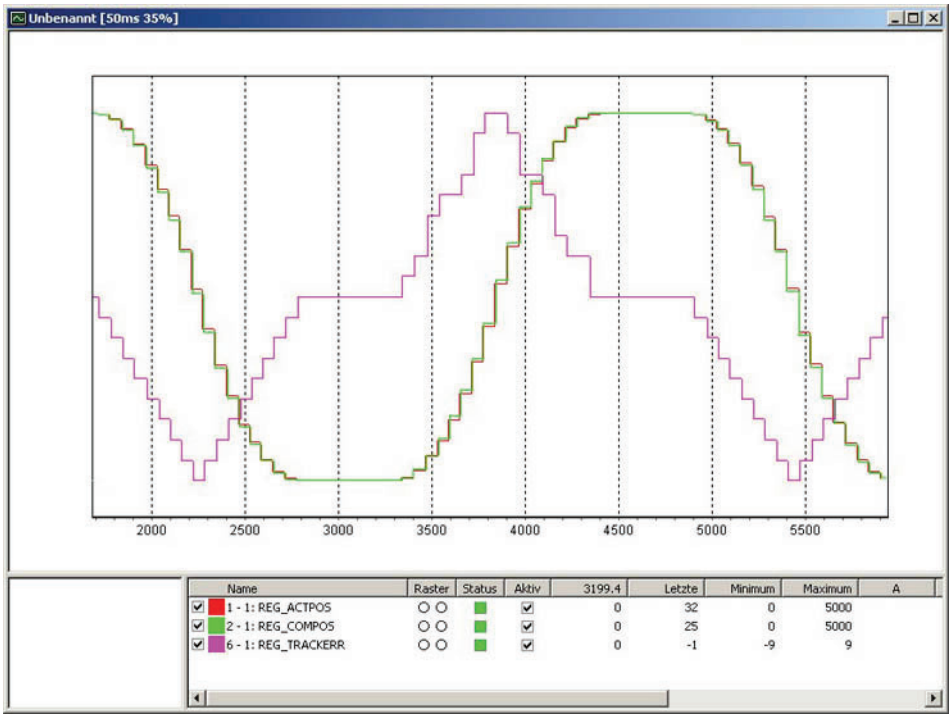
Sobald die Kurven hinzugefügt sind, wird das Oszilloskop-Fenster etwa so aussehen wie unten gezeigt. In diesem Beispiel wurden drei Kurven hinzugefügt, aber noch keine Daten aufgezeichnet, so dass das Diagramm leer ist. Das dunkelgrüne Feld in der Spalte Status zeigt, dass die Steuerung verbunden ist, aber noch kein Programm ausgeführt wird.



Weiter, wenn Polling Modus

Wenn das *FreeRun* Oszilloskop im Polling Modus betrieben wird, fügen Sie die Kurven wie sonst auch hinzu, aber statt in einer Gruppe wählen Sie → *Polled (keine Gruppe)* aus.

Die Aufzeichnung wird ganz normal mit ▶ gestartet. Das Oszilloskop fragt die Daten jeder Kurve mittels „Polling" in regelmäßigen Zeitintervallen ab. Jede Kurve wird durch Senden einer expliziten SDO-Nachricht für ihren Wert „gepollt". Sie können das Polling-Zeitintervall (d.h. die Polling-Abtastrate) in *Einstellungen* → *Oszilloskop* bestimmen. Das Oszilloskop-Fenster sieht dann etwa so aus:



Wenn das Oszilloskop gerade Daten aufzeichnet, zeigt die Titelleiste des Fensters (ganz oben links) die Abtastraste in Millisekunden und die Belastung des Netzwerks in Prozent, zum Beispiel [50ms 35%]. Die 35% sind eine Abschätzung, wie stark das Netzwerk belastet ist. Der Prozentsatz wird mehr und mehr ansteigen, je mehr Kurven gepollt werden. Ein Prozentwert höher als 100% zeigt an, dass das Oszilloskop nicht länger alle Kurven in jedem Zeitintervall pollen kann. Falls dies geschieht, sollten Sie entweder die Anzahl der Kurven die gepollt werden sollen oder das Polling-Zeitintervall reduzieren.

Die Anzahl der Kurven, die gepollt werden können und die Abtastrate mit der gepollt werden kann, verursachen eine deutliche Beschränkung des Oszilloskops, wenn es im Polling-Modus läuft. Es können viel weniger Kurven gepollt werden und deutlich weniger häufig, als dies mit PDOs möglich wäre. Das Ergebnis der aufgezeichneten Daten ist daher deutlich weniger detailliert.

SDO-Kurve

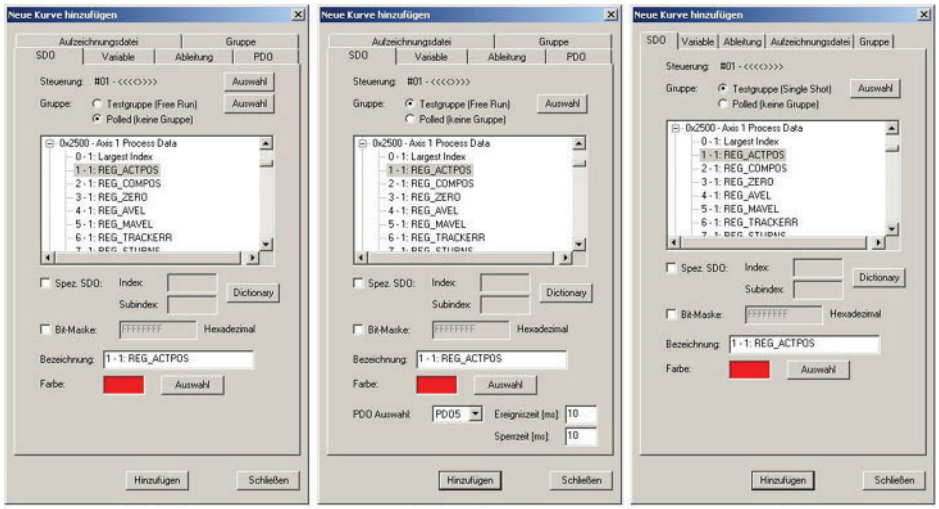
Für jedes SDO, das von der Steuerung gelesen werden kann, kann eine Kurve hinzugefügt werden.

Es gibt drei Methoden der Datenaufzeichnung: Als „polled" Werte, durch eine „Free Run"-Gruppe und durch eine „Single Shot"-Gruppe. „Free Run"-Gruppen können nur mit dem *Free Run Oszilloskop* und „Single Shot"-Gruppen nur mit dem *Single Shot Oszilloskop* benutzt werden. Die Datenaufzeichnung mittels einer Gruppenkurve ist die bevorzugte Methode, da sie die Systemnutzung sowohl von der Steuerung wie auch vom Netzwerk stark entlastet.

Polled (keine Gruppe) - Sobald eine Oszilloskop-Testfahrt gestartet ist, fragt das Oszilloskop regelmäßig die SDOs durch Senden einer „SDO Read Anfrage" zur Steuerung ab. „Polled" SDO-Kurven werden in der Kurvenauswahl unter der Überschrift „Andere Kurven" eingereiht.

Free Run Gruppe – Wenn eine Oszilloskop-Testfahrt gestartet ist, werden alle SDOs in der Gruppe in eine einzelne PDO konfiguriert. Die Steuerung reiht automatisch alle aneinander und sendet die PDO in einem regelmäßigen Zyklus zum Oszilloskop. Diese SDO-Kurven werden in der Kurvenauswahl unter der dazu-gehörigen Gruppenüberschrift eingefügt.

Single Shot Gruppe - Wenn eine Oszilloskop-Testfahrt gestartet ist, werden alle SDOs in der Gruppe in ein TESTSETP-Array konfiguriert. Wenn die Fahrt endet, lädt das Oszilloskop das TESTSETP-Array hoch und extrahiert die SDO-Daten. Diese SDO-Kurven werden in der Kurvenauswahl unter der dazugehörigen Gruppenüberschrift eingefügt.



Steuerung

Anzeige der Steuerung zu der die Kurve gehört.

Wenn eine Polled Kurve hinzugefügt wird, können Sie eine andere Steuerung für diese Kurve auswählen. Jede Polled Kurve kann mit einer anderen Steuerung verbunden werden.

Wenn eine Kurve zu einer Gruppe hinzugefügt wird, muss die Kurve mit der gleichen Steuerung verbunden sein, mit der die Gruppe verbunden ist.

Gruppe

Das SDO kann entweder abgefragt (polled) oder zu einer existierenden Gruppe hinzugefügt werden. Wenn mehr als eine Gruppe existiert, können Sie die Gruppe auswählen, zu welcher das SDO hinzugefügt werden soll.

SDO Liste

Liste zur → *Auswahl* des SDOs

Spezifiziere SDO

Falls das SDO nicht in der SDO Liste gefunden wird, aktivieren Sie das Kontrollkästchen und geben hier den genauen SDO Index und Subindex ein. Dieser kann entweder dezimal (z.B. 9472) oder hexadezimal (z.B. 0x2500) eingeben werden.

Bit-Maske

Wenn der SDO-Wert eine interne Kodierung hat (z.B. mehrere Werte in einen einzigen SDO-Wert gepackt), kann diese Maske benutzt werden um die dazu gehörenden Bits zu extrahieren. Zum Beispiel, ein Wert von 000000FF extrahiert die niederwertigen 8 Bits vom 32-Bit SDO Wert.

Bezeichnung

Diese Bezeichnung erscheint in der Kurvenauswahl. Ein Default-Eintrag wird generiert, kann aber beliebig geändert werden. Die Default-Bezeichnung wird als „Subindex - Achse: Bezeichnung" formatiert.

Farbe

Sie wird zum Zeichnen der Kurve benutzt und kann mit → *Auswahl* beliebig geändert werden.

PDO Auswahl

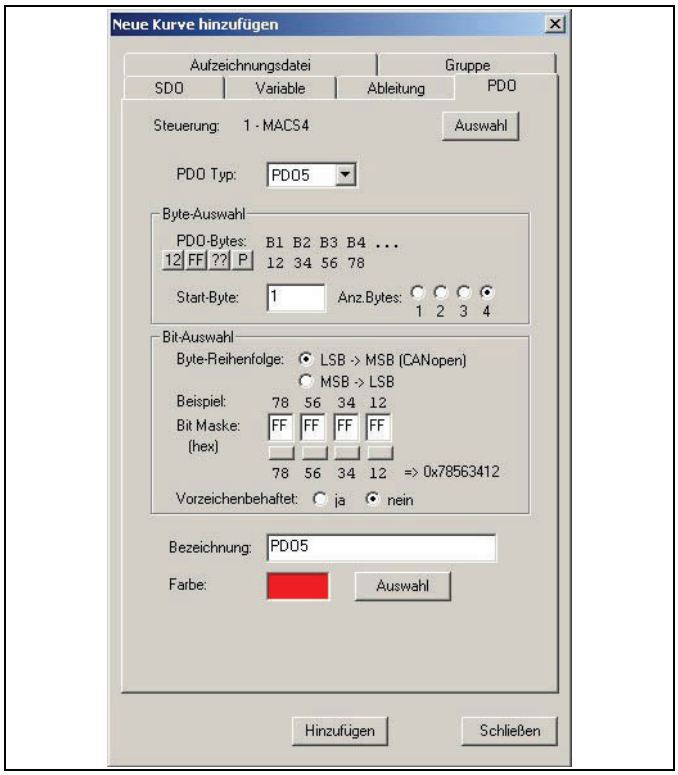
Wenn das SDO zu einer Free Run Gruppe hinzufügt wurde, dann können Sie das PDO auswählen, in welches das SDO konfiguriert wird. Die PDO *Ereigniszeit* und *Sperrzeit* kann ebenfalls festgelegt werden. (Siehe SDO 1800 in „SDO Object Dictionary" CANopen Basic Data im Handbuch „MCO 305 Befehlsreferenz".)

ACHTUNG!:
Die Felder *Spezifiziere SDO*, *Bit Maske* und *Sperrzeit* sind nur verfügbar, wenn der APOSS Benutzer-Modus im Menü *Einstellungen* → *Optionen*. auf → *Experte* eingestellt ist.

PDO-Kurve

Wenn die in der Steuerung laufende Anwendung PDOs sendet, kann eine PDO-Kurve benutzt werden, um Daten aus dem PDO zu extrahieren. Die Daten können von jeder Byte-Position innerhalb des PDOs extrahiert werden.

Beachten Sie, dass PDO-Kurven nicht gepollt und nicht zu Gruppen hinzugefügt werden können. Die Daten werden asynchron durch das Oszilloskop aufgezeichnet, wann immer ein PDO eintrifft. PDO-Kurven werden unter der Überschrift „Andere Kurven" der Kurvenauswahl hinzugefügt.



Steuerung

Anzeige der Steuerung, zu der die Kurve gehört. Sie können mit → *Auswahl* eine andere Steuerung für das PDO auswählen. Jede PDO-Kurve kann zu einer anderen Steuerung gehören.

PDO Typ

Wählen Sie im Auswahlmenü den → *PDO-Typ* aus.

Beachten Sie, dass nicht alle Schnittstellen alle PDO-Typen unterstützen und einige Schnittstellen PDOs überhaupt nicht unterstützen.

Byte-Auswahl

Dieser Abschnitt wird benutzt, um auszuwählen welche Bytes im PDO Daten von Interesse enthalten. Die Bytes werden nummeriert beginnend mit 1, welches das erste Datenbyte im PDO ist. Die erforderliche Nummer der Bytes kann 1 bis 4 sein. Der String „B1 B2..." mit „12 34..." darunter gibt dem Anwender einen Hinweis, welche Bytes augenblicklich ausgewählt sind.

Die vier kleinen Schaltflächen können Sie benutzen, um Test-PDOs zu generieren, die Ihnen helfen können, die richtige Byte- und Bit-Spezifikation zu verifizieren.

12 - Der Byte String 0x12 0x34 0x56 0x78 wird benutzt.
FF - Der Byte String 0xFF 0xFF 0xFF 0xFF wird benutzt.
?? - Ein String mit Zufallsbytes wird erzeugt.
P - Das Oszilloskop wartet auf ein ankommendes PDO; sobald erkannt, werden die Bytes aus dem PDO extrahiert.

Bit-Auswahl

Dieser Abschnitt wird benutzt und festzulegen wie der Wert aus den ausgewählten Bytes extrahiert werden soll. Die *Byte-Reihenfolge* kann entweder *LSB* > *MSB* sein (das ist die Standard CANopen Byte-Reihenfolge und wird für Befehle wie PDO und LINKSDO benutzt) oder *MSB* > *LSB* (was für Low-Level Befehle wie CANIN und CANOUT benutzt werden sollte).

Die *Beispiel* Bytes sind exakt die Test PDO Bytes des Abschnitts *Byte-Auswahl* darüber. Die Beispiel Bytes in diesem Abschnitt werden immer in der Standard-Reihenfolge *MSB* > *LSB* für einen einzelnen Multi-Byte-Wert dargestellt.

Das Ändern der *Byte-Reihenfolge* ändert auch die Reihenfolge des Beispiels, so dass man genau weiß, wie die ankommenden PDO-Bytes behandelt werden.

Das Feld *Bit-Maske* enthält hexadezimale Werte, die als Maske benutzt werden, um die Kurvendaten aus den ausgewählten Bytes zu extrahieren. Die kleine Schaltfläche unter jedem Feld dient dazu, die Maske zwischen 0xFF und 0x00 zu wechseln. Unter diesen Schaltflächen werden die gleichen Beispiel-Bytes wieder dargestellt, diesmal eingefügt in der Bit-Maske.

Und schließlich kann man festlegen, ob der Wert *Vorzeichenbehaftet* oder nicht behandelt wird.

Bezeichnung

Diese Bezeichnung erscheint in der Kurvenauswahl. Der Default-Eintrag wird generiert, kann aber beliebig geändert werden.

Farbe

Sie wird zum Zeichnen der Kurve benutzt und kann mit → *Auswahl* beliebig geändert werden.

Variablen-Kurve

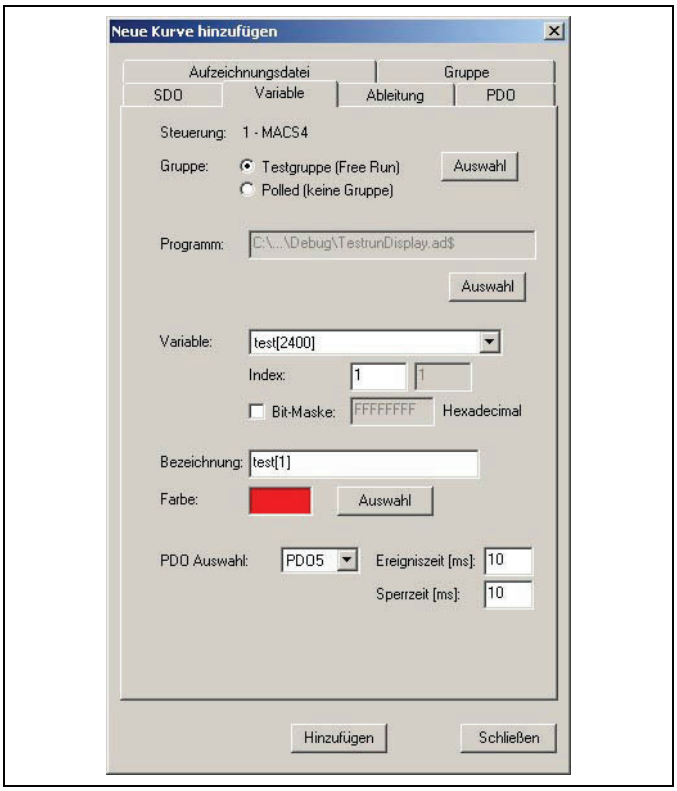
Wenn eine gerade in der Steuerung ausgeführte Anwendung bekannt ist und von APOSS kompiliert wurde, können die Kurven für jede Variable, die im Programm benutzt wird, zum Oszilloskop hinzugefügt werden.

Wie bei SDO Kurven gibt es drei Methoden der Datenaufzeichnung:

Polled (keine Gruppe) – Sobald eine Oszilloskop-Testfahrt gestartet ist, fragt das Oszilloskop regelmäßig die SDOs durch Senden einer „SDO Read Anfrage" zur Steuerung ab. „Polled" SDO-Kurven werden in der Kurvenauswahl unter der Überschrift „Andere Kurven" eingereiht.

Free Run Gruppe – Wenn eine Oszilloskop-Testfahrt gestartet ist, werden alle SDOs in der Gruppe in ein einzelnes PDO konfiguriert. Die Steuerung reiht automatisch alle aneinander und sendet das PDO in einem regelmäßigen Zyklus zum Oszilloskop. Diese SDO-Kurven werden in der Kurvenauswahl unter der dazugehörigen Gruppenüberschrift eingefügt.

Single Shot Gruppe – Wenn eine Oszilloskop-Testfahrt gestartet ist, werden alle SDOs in der Gruppe in ein TESTSETP-Array konfiguriert. Wenn die Fahrt endet, lädt das Oszilloskop das TESTSETP-Array hoch und extrahiert die SDO-Daten. Diese SDO-Kurven werden in der Kurvenauswahl unter der dazugehörigen Gruppenüberschrift eingefügt.



ACHTUNG!:
Das Oszilloskop kann nicht erkennen, ob das Programm das in der Steuerung läuft mit dem vom Anwender ausgewählten Programm identisch ist, wenn eine *Variablen-Kurve* hinzugefügt wird. Daher muss der Anwender sicherzustellen, dass dies der Fall ist. Eine Diskrepanz führt zwar nicht zum Abbruch, es werden aber auch nicht die richtigen Ergebnisse für die Kurve erzeugt.

Steuerung

Anzeige der Steuerung zu der die Kurve gehört.

Wenn eine Polled Kurve hinzugefügt wird, können Sie eine andere Steuerung für die Kurve auswählen. Jede Polled Kurve kann einer anderen Steuerung zugeordnet werden.

Wenn die Kurve zu einer Gruppe hinzugefügt wird, dann muss die Kurve der Steuerung zugeordnet werden, mit der die Gruppe verbunden ist.

Gruppe

Die Variable Kurve kann entweder als Polled oder zu einer existierenden Testgruppe hinzugefügt werden. Wenn mehr als eine Gruppe existiert, treffen Sie die → *Auswahl*.

Programm

Das ist das Programm das die Variablen enthält, die von Interesse sind. Es liegt in der Verantwortung des Anwenders, sicherzustellen, dass dies auch das Programm ist, das in der Steuerung läuft.

ACHTUNG!:
Die Dateierweiterung der Programmdatei ist „.ad$". Dies ist der Datei-Querverweis, der vom APOSS Compiler erzeugt wurde. (Siehe *Einstellungen* → *Compiler* im Kapitel „APOSS Benutzeroberfläche".)

Mit → *Auswahl* können Sie eine andere Programmdatei auswählen.

Variable

In dieser Liste mit allen im Programm gefundenen Variablen wählen Sie diejenige Variable aus, um die es geht. Wenn ein Array (entweder ein- oder zweidimensional) ausgewählt wird, dann müssen Sie mit → *Index* genau festzulegen, welches Array-Element durch die Kurve aufgezeichnet werden soll.

Wenn der Variablen-Wert eine interne Kodierung hat (z.B. mehrere Werte in einen einzigen Wert gepackt), dann benutzen Sie die → *Bit-Maske* um die zugehörigen Bits zu extrahieren. Zum Beispiel, extrahiert ein Wert von 000000FF die niederwertigen 8 Bits aus einem 32-Bit Variablen Wert.

ACHTUNG!:
Bit-Maske kann nicht mit „Double" Variablen benutzt werden.

Bezeichnung

Diese Bezeichnung erscheint in der Kurvenauswahl. Ein Default-Eintrag wird generiert, kann aber beliebig geändert werden.

Farbe

Sie wird zum Zeichnen der Kurve benutzt und kann mit → *Auswahl* beliebig geändert werden.

PDO Auswahl

Wenn die Variable zu einer Testgruppe (Free Run) hinzufügt wurde, dann können Sie das PDO auswählen, in welches die Variable konfiguriert wird. Die PDO *Ereigniszeit* und *Sperrzeit* kann ebenfalls festgelegt werden. (Siehe SDO 1800 in „SDO Object Dictionary" CANopen Basic Data im Handbuch „MCO 305 Befehlsreferenz".)

ACHTUNG!:
Das Feld *Sperrzeit* ist nur verfügbar, wenn der APOSS Benutzer-Modus im Menü *Einstellungen* → *Optionen* auf → *Experte* eingestellt ist.

Ableitung / abgeleitete Kurve

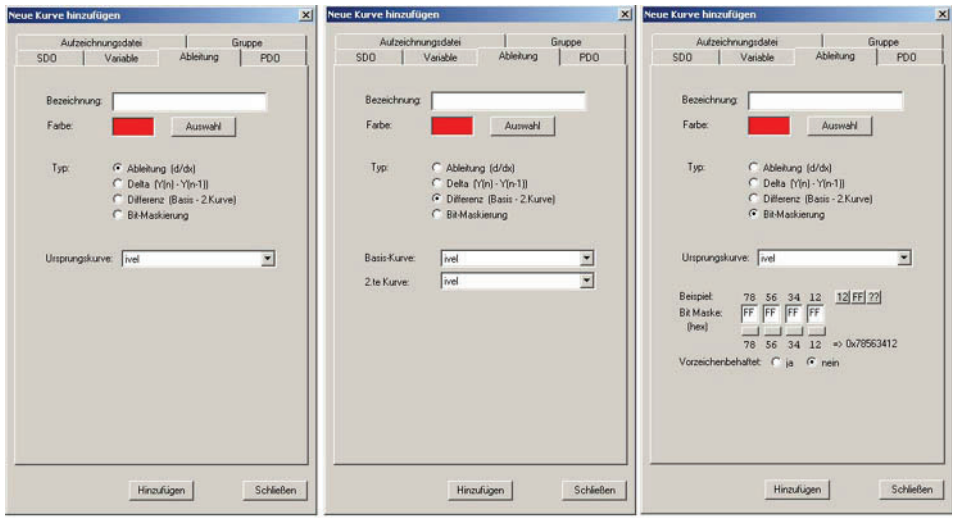
Wenn es schon Kurven gibt, können weitere hinzugefügt werden, die sich von diesen ableiten. Die abgeleitete Kurve wird immer nach der Kurve in die Kurvenauswahl eingefügt, von der sie abgeleitet ist. Es gibt folgende Arten von abgeleiteten Kurven:

Ableitung ist die mathematische Ableitung (d.h. d/dx) der ursprünglichen Kurve.

Delta zeigt die inkrementalen Differenzen entlang der ursprünglichen Kurve. Jeder Punkt auf der Kurve ist die Differenz zwischen dem entsprechenden und dem vorherigen Punkt der ursprünglichen Kurve.

Differenz ist die einfache Differenz (d.h. Subtraktion) zwischen den Werten zweier ursprünglicher Kurven.

Bit-Maskierung – Wenn eine Kurvenwert eine interne Kodierung hat (z.B. mehrere Werte in einen einzigen SDO-Wert gepackt), dann leitet diese Kurve ihre Daten von Bits ab, die aus den Werten der ursprünglichen Kurve extrahiert werden. Zum Beispiel extrahiert die Maske 000000FF die niederwertigen 8 Bits aus dem 32-Bit-Wert.



Ableitung / Delta | Differenz | Bit-Maskierung

Bezeichnung

Die Bezeichnung erscheint in der Kurvenauswahl und sollte daher aussagefähig sein.

Color

Die Farbe, in der die Kurve gezeichnet wird, können Sie mit → *Auswahl* ändern.

Typ

Wählen Sie den Typ der Ableitung für die Kurve, die hinzugefügt werden soll.

Ursprungskurve(n)

Bestimmen Sie die → *Ursprungskurve*, von der die neue abgeleitet werden soll.

2. Kurve - Wenn die neue Kurve aus zwei Kurven abgeleitet wird (z.B. bei einer Differenz-Kurve) bestimmen Sie auch die → *2. Kurve*.

Bit-Maske

Wird benutzt um festzulegen, wie der Wert einer Bit-Maske aus dem ursprünglichen Wert (der immer ein 32-Bit-Wert ist) extrahiert wird. Die *Beispiel* Bytes können als Hilfe benutzt werden, um die Richtigkeit der Bit-Maskierung zu prüfen. Mit den die drei kleinen Schaltflächen können Sie verschiede Testwerte für das Beispiel generieren.

12 Der Wert 0x78563412 wird benutzt.
FF Der Wert 0xFFFFFFFF wird benutzt
?? Ein Zufallswert wird generiert.

Das Feld *Bit-Maske* enthält hexadezimale Werte, die als Maske benutzt werden, um die Kurvendaten aus den ausgewählten Bytes zu extrahieren. Mit den kleinen Schaltflächen unter jedem Feld kann man die Maskierung zwischen 0xFF und 0x00 wechseln. Unter diesen Schaltflächen werden die gleichen Beispiel-Bytes wieder dargestellt, diesmal eingefügt in der Bit-Maske.

Und schließlich kann man festlegen, ob der Wert *Vorzeichenbehaftet* oder nicht behandelt wird.

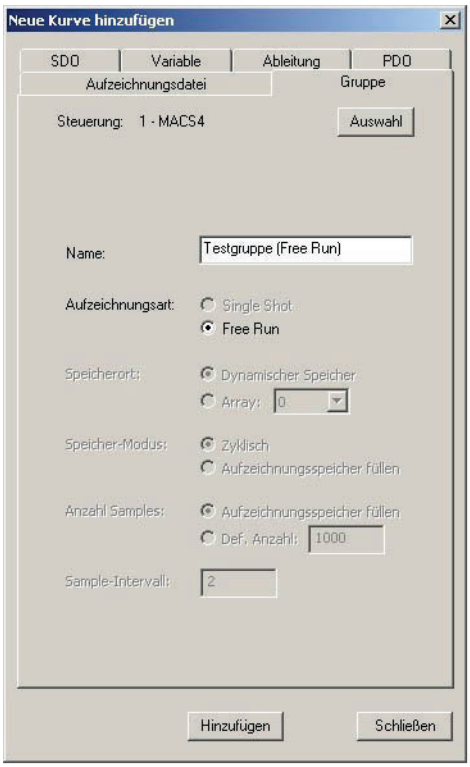
Gruppe / Gruppen-Kurve

Eine *Gruppen-Kurve* ist eigentlich keine Kurve und enthält auch keine aufgezeichneten Daten. Stattdessen enthält sie einfach eine Sammlung von anderen Kurven und ermöglicht es, dass der ganze Satz der Kurven zusammen gehandhabt werden kann. Zum Beispiel werden alle Kurven einer Gruppe durch ein *Free Run Oszilloskop* in ein einziges PDO konfiguriert, um die Werte von der Steuerung in das Oszilloskop zu senden. Oder alle Kurven einer Gruppe werden durch ein *Single Shot Oszilloskop* in ein einziges TESTSETP-Array für die Datenerfassung konfiguriert.

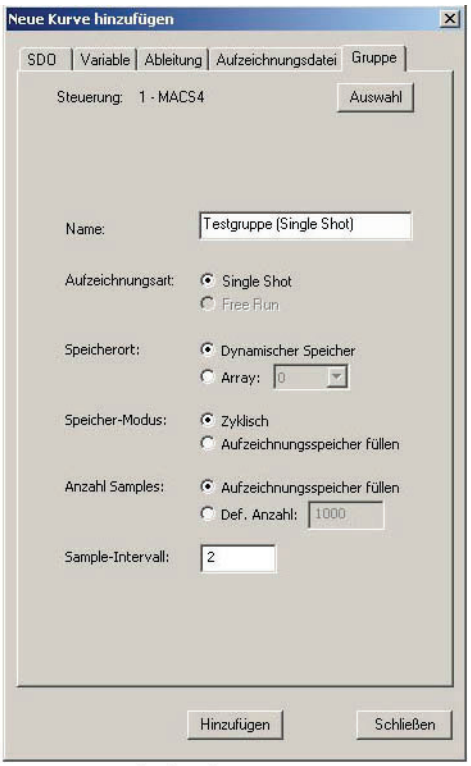
ACHTUNG!:
Beachten Sie: Eine Konsequenz davon ist, dass alle Kurven einer Gruppe mit der gleichen Steuerung verbunden sein müssen.

Die bevorzugte Methode beim Einsatz des Oszilloskops ist, erst eine Gruppen-Kurve hinzuzufügen und dann andere Kurven zu dieser Gruppe anzuhängen. Eine einzelne Gruppen-Kurve ist für die meisten Anwendungen des Oszilloskop geeignet. In manchen Fällen jedoch (z.B. Testen mit mehreren Steuerungen) kann das Hinzufügen mehrerer Gruppen-Kurven nützlich sein. Dies ist möglich, wenn der Benutzer-Modus auf „Expert" gesetzt ist (siehe *Einstellungen* → *Optionen* im Kapitel „PC Software Benutzeroberfläche".)

Gruppen-Kurven werden in der Kurvenauswahl mit einem grauen Hintergrund dargestellt. Gruppen werden in der Liste immer nach allen vorher vorhandenen Gruppen eingereiht und vor den „anderen Kurven", falls vorhanden.



Free Run Gruppe



Single Shot Gruppe

MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

Bezeichnung

Diese Bezeichnung erscheint in der Kurvenauswahl. Ein Default-Eintrag wird generiert, sollte aber vom Anwender mit einem treffenden Namen überschrieben werden.

Aufzeichnungsart

Die Aufzeichnungsart ist – je nachdem welches Oszilloskop benutzt wird – automatisch ausgewählt, also *Free Run,* wenn das *Free Run Oszilloskop* benutzt wird. Dann werden von der Steuerung in Echtzeit die Sample-Daten mit PDOs zum Oszilloskop gesendet.

Wird das *Single Shot Oszilloskop* benutzt, ist *Single Shot* markiert. In diesem Fall werden die Abtastpunkte in ein TESTSETP-Array aufgezeichnet und am Ende des Tests zum Oszilloskop gesendet.

Alle folgenden Eigenschaften betreffen nur *Single Shot Gruppen* und damit verschiede Aspekte des TESTSETP-Arrays, das für die Aufzeichnung der Kurvendaten während des Tests konfiguriert ist.

Speicherort

Bestimmen Sie, wo das TESTSETP-Array abgelegt werden soll. Falls auf *Dynamischer Speicher* gesetzt, wird das TESTSETP-Array im verbleibenden nicht benutzten Speicher der Steuerung abgelegt.

Beachten Sie, dass die Menge des nicht genutzten Speichers in Abhängigkeit von solchen Dingen wie die Größe des geladenen Programms, die Anzahl und Größe von allen vordefinierten Benutzer-Arrays, der Anzahl der gespeicherten Programme usw. variiert. Wenn ein *Array* ausgewählt ist, wird das TESTSETP-Array im spezifizierten Array abgelegt. Dieses Array muss in der Steuerung vordefiniert sein (d.h. durch Definition in einem Anwendungsprogramm oder mittels *Array Editor* usw.).

Speicher-Modus

Hiermit wird festgelegt, was passieren soll, wenn das TESTSETP-Array voll wird.

Wenn auf *Zyklisch* gesetzt, geht der älteste vorhandene Datenpunkt verloren, um Platz für den neuesten Datenpunkt zu machen. Das TESTSETP-Array enthält also nur den letzten Satz der aufgezeichneten Punkte.

Wenn auf *Stoppen wenn voll* gesetzt, werden neue Datenpunkte übersprungen und das TESTSETP-Array enthält nur die ersten und ältesten Sätze der aufgezeichneten Punkte.

Anzahl Samples

Legt die Anzahl der Sätze der Abtastpunkte fest, die gespeichert werden können und bestimmt damit die Größe des TESTSETP-Arrays.

Beachten Sie, dass sich die Angabe auf die Anzahl der Sätze der Punkte bezieht (d.h. ein Sample-Punkt für jede Kurve in der Gruppe) und nicht auf die gesamte Anzahl der individuellen Abtastpunkte.

Wenn auf *Aufzeichnungsspeicher füllen* gesetzt, wird das TESTSETP-Array so groß wie möglich gemacht, unter Berücksichtigung, dass es immer noch entweder in den Dynamischen Speicher oder in das vordefinierte Benutzer-Array passt.

Alternativ können Sie in → *Definiere Anzahl* eine bestimmte Anzahl von Abtastpunkten festlegen.

Sample-Intervall

Legt das Zeitintervall in Millisekunden zwischen jedem gesicherten Satz von Abtastpunkten fest.

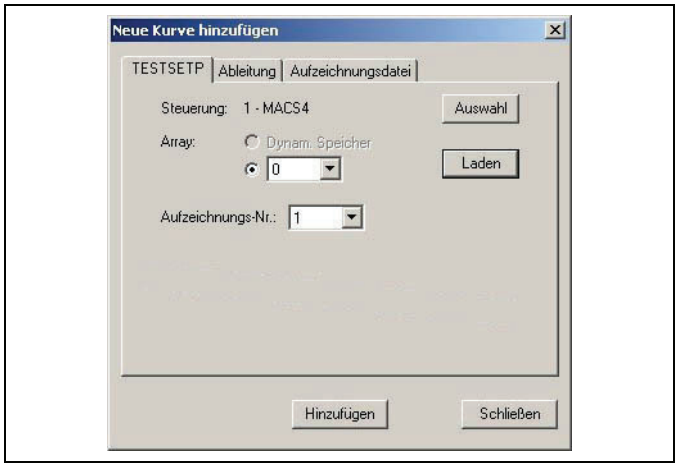
TESTSETP-Kurve

Wenn Sie ein kundenspezifisches TESTSETP-Array lieber in einem APOSS Programm konfigurieren, als mit dem Single Shot Oszilloskop automatisch, müssen Sie eine TESTSETP-Kurve in das Oszilloskop hinzufügen, um die Daten zu speichern. Das ist zum Beispiel für ältere Steuerungen notwendig, da diese die automatische Konfiguration von TESTSETP Arrays nicht unterstützen.

Sobald die TESTSETP-Kurve hinzugefügt ist, wird das Oszilloskop sofort versuchen, das TESTSETP-Array von der Steuerung zu lesen. Daher muss die Testfahrt schon beendet und das TESTSETP-Array in der Steuerung erzeugt worden sein, bevor die TESTSETP-Kurve in das Oszilloskop hinzugefügt wird.

Wenn eine TESTSETP-Kurve hinzugefügt wird, fügt das Oszilloskop tatsächlich eine eigene Kurve für jede Kurve, die im TESTSETP-Array definiert ist, hinzu. Die erste dieser Kurven wird immer mit einem grauen Hintergrund in der Kurvenauswahl markiert.

TESTSETP-Kurven können nur hinzugefügt werden, wenn das TESTSETP Oszilloskop benutzt wird.



Steuerung

Dies ist die Steuerung, die das TESTSETP-Array enthält. Mit → *Auswahl* kann eine andere Steuerung ausgewählt werden.

Array

Das Array kann entweder vom Dynamischen Speicher oder von einem vordefinierten Benutzer-Array geladen werden. Treffen Sie Auswahl und klicken Sie auf → *Laden*. Daraufhin lädt das Oszilloskop das TESTSETP-Array von der Steuerung hoch.

ACHTUNG!:
Die Auswahl *Dynamischer Speicher* ist deaktiviert, wenn die Steuerung diesen nicht unterstützt oder wenn der *Dynamischer Speicher* keine TESTSETP-Array enthält.

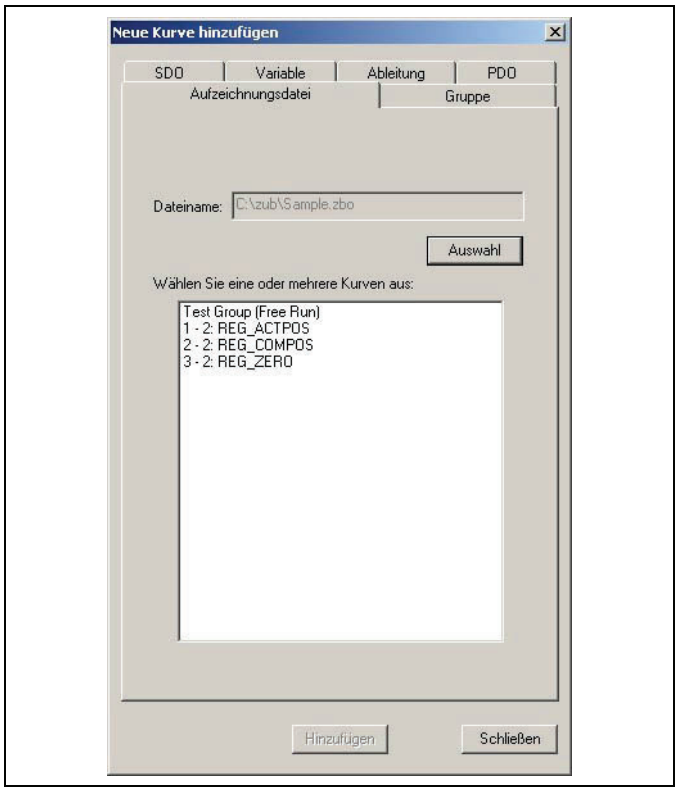
Aufzeichnungs-Nummer

Wenn das TESTSETP-Array mehrere Aufzeichnungen enthält, müssen Sie eine davon auswählen. Es können mehr als eine Aufzeichnung zum Oszilloskop hinzugefügt werden, aber für jede muss auch eine eigene TESTSETP-Kurve hinzugefügt werden.

Kurven von anderen Oszilloskop-Dateien hinzufügen

Kurven die in anderen Oszilloskop-Dateien „.zbo" definiert sind, können mittels der Registerkarte *Aufzeichnungsdatei* in das aktuelle Oszilloskop kopiert werden.

Beachten Sie: Diese Funktion ist nur verfügbar, wenn der Benutzer-Modus in *Einstellungen* → *Optionen* auf „Experte" gesetzt ist.



Dateiname

Dies ist die Oszilloskop-Datei mit der Kurve, die geöffnet wird. Wählen Sie die Oszilloskop-Datei aus → *Auswahl*.

Liste der Kurven

In diesem Feld werden die Kurven aufgelistet, die in der ausgewählten Oszilloskop-Datei gefunden wurden.

Markieren Sie die Kurve die kopiert werden soll mit der linken Maustaste. Mehrere Kurven können Sie auswählen, indem Sie die [Strg]-Taste drücken, während Sie die Auswahl treffen.

Einige Kurven basieren auf dem Vorhandensein anderer Kurven und können nicht als einzelne Kurve kopiert werden. Zum Beispiel kann eine Abgeleitete Kurve nicht ohne der Kurve, von der sie abgeleitet wurde, kopiert werden. Ebenso müssen alle Kurven in einer Gruppe gemeinsam kopiert werden. Wenn Sie eine dieser Kurven auswählen, werden die erforderlichen anderen automatisch ausgewählt.

ACHTUNG!:
Die Liste enthält nur jene Kurven der ausgewählten Oszilloskop-Datei, die kompatibel mit dem gerade benutzten Oszilloskop sind. Wenn zum Beispiel die ausgewählte Oszilloskop-Datei Single Shot Gruppen enthält, werden diese Gruppen nicht aufgelistet, falls gerade ein *Free Run Oszilloskop* benutzt wird.

□ Test starten und beenden

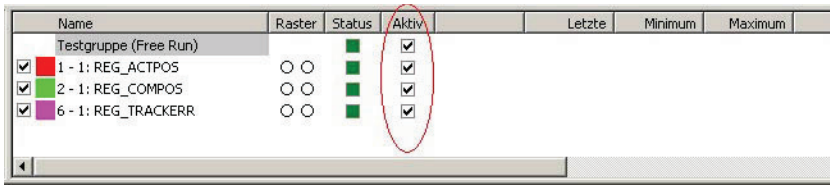
Sobald alle Kurven hinzugefügt wurden, kann der Test gestartet werden. Dies bedingt drei Schritte:

1. Kurve auswählen, die aufgezeichnet werden soll
2. Anwendungsprogramm in der Steuerung starten
3. Oszilloskop-Aufzeichnung starten

Kurve auswählen, die aufgezeichnet werden soll

Sie können viele Kurven im Oszilloskop definieren, aber es ist nicht notwendig die Daten für alle Kurven bei jedem Test aufzuzeichnen. Benutzen Sie das Kontrollkästchen der Spalte „Aktiv" in der Kurvenauswahl, um die Kurven auszuwählen, die aufgezeichnet werden sollen.

Im folgenden Beispiel werden die Daten nur für die Kurven „REG_ACTPOS" (aktuelle Position) und „REG_TRACKERR" (Schleppfehler) aufgezeichnet, jedoch nicht für „REG_COMPOS".

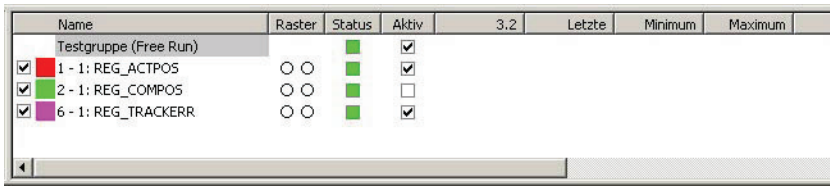


Beachten Sie, Gruppen werden immer als Ganzes ausgewählt oder nicht ausgewählt und folglich auch so behandelt: Es werden entweder Daten für alle Kurven einer Gruppe aufgezeichnet oder die Kontrollkästchen sind grau und so die Aufzeichnung für die ganze Gruppe deaktiviert.

Wenn mehrere Gruppen definiert sind, gibt es hinsichtlich der Gruppen, die zur gleichen Zeit ausgewählt werden können, eine Einschränkung: Keine zwei Gruppen können zur gleichen Zeit ausgewählt werden, wenn sie mit der gleichen Steuerung verbunden sind und das gleiche PDO benutzen. Sind Gruppen in dieser Art definiert, deaktiviert die Auswahl einer Gruppe automatisch die Auswahl der anderen.

Anwendungsprogramm starten

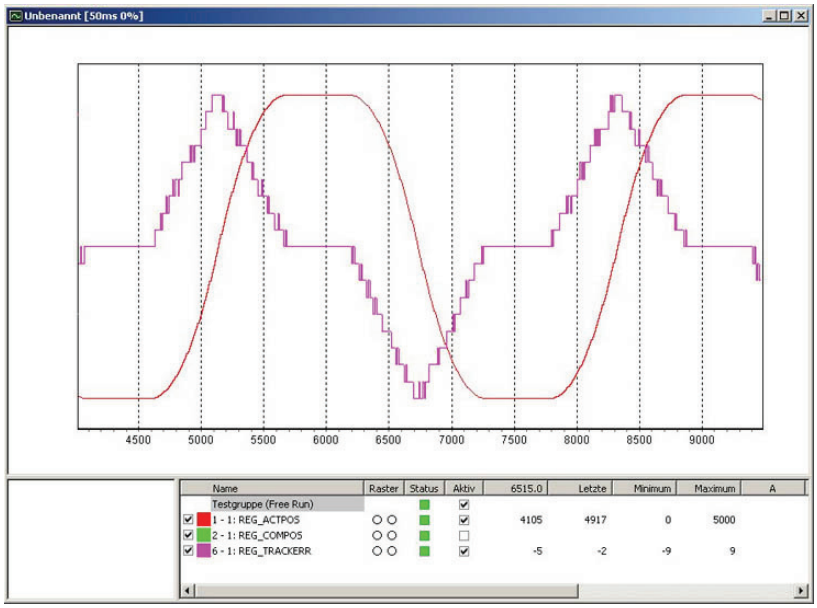
Im nächsten Schritt starten Sie das Anwendungsprogramm in der Steuerung, falls es nicht schon läuft. Wechseln Sie dazu in das APOSS-Fenster und starten Sie die Anwendung wie üblich. Dann wechseln Sie wieder in das Oszilloskop-Fenster. Die Kurvenauswahl sollte nun wie folgt aussehen: Die hellgrünen Felder in der Spalte Status zeigen, dass ein Programm ausgeführt wird.



Oszilloskop-Aufzeichnung starten

Starten Sie jetzt die Oszilloskop-Aufzeichnung mit ▶ in der Symbolleiste.

Zu diesem Zeitpunkt sollte die Aufzeichnung der Kurvendaten starten und im Grafikfenster dargestellt werden, und zwar in Echtzeit (wenn Free Run Oszilloskop) gemäß dem Fortschreiten des Tests oder nach dem Ende der Aufzeichnung (wenn Single Shot). Das Oszilloskop-Fenster sollte nun so aussehen:



Die Reihenfolge, in der die Ausführung des Testprogramms gestartet wird, ist nicht festgelegt. Es steht dem Anwender frei, erst die Programmausführung zu starten und dann die Aufzeichnung oder umgekehrt.

Jederzeit können während der Aufzeichnung die ***Zoom***-Schaltflächen benutzt werden; jede der Kurven kann mittels der Kontrollkästchen vor der Spalte Name im Grafikfenster ausgeblendet oder dargestellt werden, und die Y-Achse des Diagramms kann mittels der Spalte Raster gesetzt werden. Beachten Sie, dass das Navigationsfenster während der Aufzeichnung leer ist und also auch nicht benutzt werden kann.

Jeder Kurve hat eine festgelegte maximale Anzahl von Punkten, die aufgezeichnet werden können. Sobald diese Anzahl erreicht ist, werden die ältesten Punkte verworfen, um Platz für die neuen zu schaffen. Sie können die maximale Anzahl in *Eigenschaften* → *Quelle ändern*.

Die Aufzeichnung wird so lange fortgesetzt, bis sie mittels ■ gestoppt wird.

ACHTUNG!:
Das Stoppen der Aufzeichnung bricht nicht die Ausführung des Programms ab; das Programm läuft weiter. Die ■ *Stopp*-Schaltfläche stoppt nur die Aufzeichnung von neuen Daten.

ACHTUNG!:
Das ausführende Programm kann jederzeit ganz einfach abgebrochen werden: Dazu wechseln Sie in das APOSS-Fenster und drücken die [Esc]-Taste.
Das Abbrechen des Programms stoppt aber nicht die Aufzeichnung durch das Oszilloskop. Das Oszilloskop empfängt weiterhin ankommende PDOs, bis mit ■ tatsächlich gestoppt wird.

Weiter, wenn Single Shot Oszilloskop

Wenn mehrere Gruppen benutzt werden, kann nur eine Gruppe pro Steuerung für die Aufzeichnung ausgewählt werden. Wenn eine zweite Gruppe für die gleiche Steuerung ausgewählt wird, wird die andere automatisch abgewählt.

Starten Sie die Aufzeichnung wie oben beschrieben mit ▶ in der Symbolleiste. Es werden jedoch keine Kurven im Grafikfenster dargestellt, solange die Aufzeichnung läuft.

Sobald die Aufzeichnung gestoppt wird, erscheint ein Fenster ähnlich wie das folgende und zeigt, dass das TESTSETP-Array in das Oszilloskop hochgeladen wird. Die Kurven werden im Grafikfenster dargestellt, sobald der Upload fertig ist.

☐ Ergebnisse analysieren

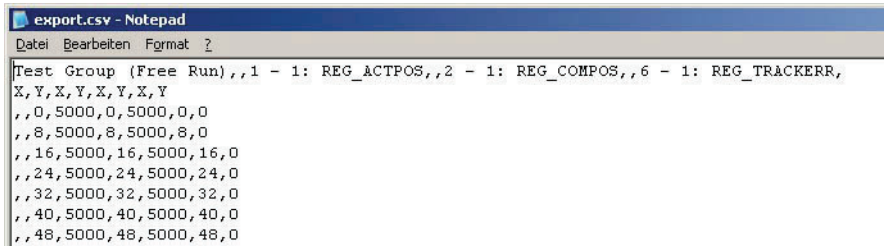
Jede der folgenden Funktionen und Eigenschaften des Oszilloskops kann helfen, die aufgezeichneten Daten zu analysieren:

1. Die Kurven im Grafikfenster könnten Korrelationen oder Anomalien aufweisen. Benutzen Sie die *Zoom* Schaltflächen und das Navigations-Fenster, um mehr Details der Kurven zu sehen.
2. In den Spalten Letzter, Minimum und Maximum in der Kurvenauswahl könnten nützliche Informationen stehen.
3. Benutzen Sie Maus und Cursor um die Kurven vorwärts und rückwärts abzusuchen. Die Kurvenwerte werden in der Kurvenauswahl dargestellt.
4. Benutzen Sie *Eigenschaften* → *Anzeige*, um die *Bereichsüberprüfung* für Kurven zu deaktivieren. Das macht es einfacher im „Aus"-Probleme zu lokalisieren.
5. Erzeugen Sie *abgeleitete Kurven*. Zum Beispiel kann eine abgeleitete Kurve die Differenz zwischen zwei Kurven zeigen oder den Grad wie sich die Kurve ändert.

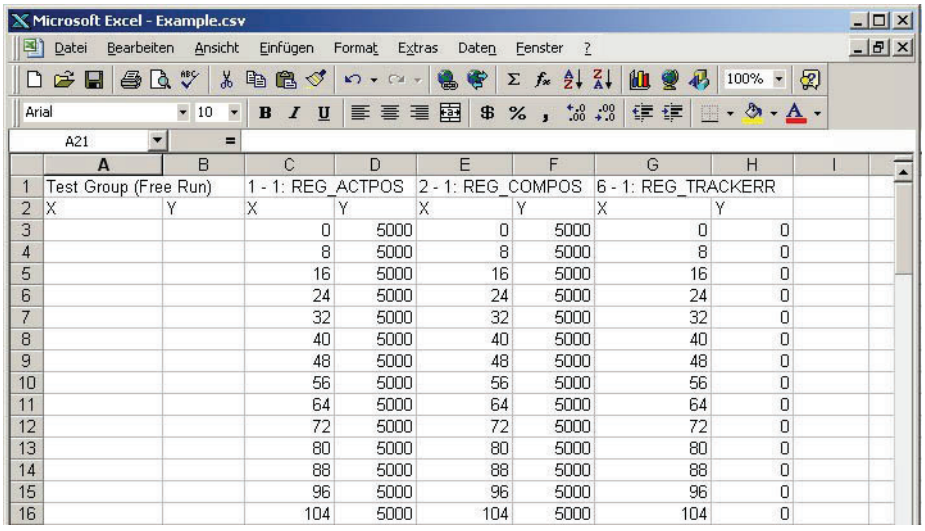
Kurvendaten in eine txt-Datei exportieren

Mit *Export* können Sie die Kurvendaten in eine Standard-Textdatei exportieren, um die Daten ggf. mit anderen Programmen zu analysieren.

Alle Kurvendaten werden als csv Datei (Werte durch Komma getrennt) exportiert. Das ist eine Standard-Textdatei, die in ein Tabellenkalkulationsprogramm importiert werden kann und mit einem normalen Texteditor geöffnet werden kann. Die Datei sieht etwa so aus:



Sobald sie in Microsoft Excel importiert ist, sieht sie etwa so aus:



	A	B	C	D	E	F	G	H
1	Test Group (Free Run)		1 - 1: REG_ACTPOS		2 - 1: REG_COMPOS		6 - 1: REG_TRACKERR	
2	X	Y	X	Y	X	Y	X	Y
3			0	5000	0	5000	0	0
4			8	5000	8	5000	8	0
5			16	5000	16	5000	16	0
6			24	5000	24	5000	24	0
7			32	5000	32	5000	32	0
8			40	5000	40	5000	40	0
9			48	5000	48	5000	48	0
10			56	5000	56	5000	56	0
11			64	5000	64	5000	64	0
12			72	5000	72	5000	72	0
13			80	5000	80	5000	80	0
14			88	5000	88	5000	88	0
15			96	5000	96	5000	96	0
16			104	5000	104	5000	104	0

☐ Test wiederholen

Sobald die Oszilloskop-Aufzeichnung angehalten wurde, können Sie beliebig die Kurven ändern, ergänzen und löschen. Die Aufzeichnung wird mit ▶ wieder gestartet.

ACHTUNG!:
Dies löscht alle vorher aufgezeichneten Daten! Die vorher aufgezeichneten Daten werden nur dann nicht gelöscht, wenn mit ⏸ *Pause* unterbrochen wurde, statt gestoppt. In diesem Fall zeigt eine vertikale rote Linie im Diagramm wo die Aufzeichnung unterbrochen wurde.

Weiter, wenn TESTSETP Oszilloskop

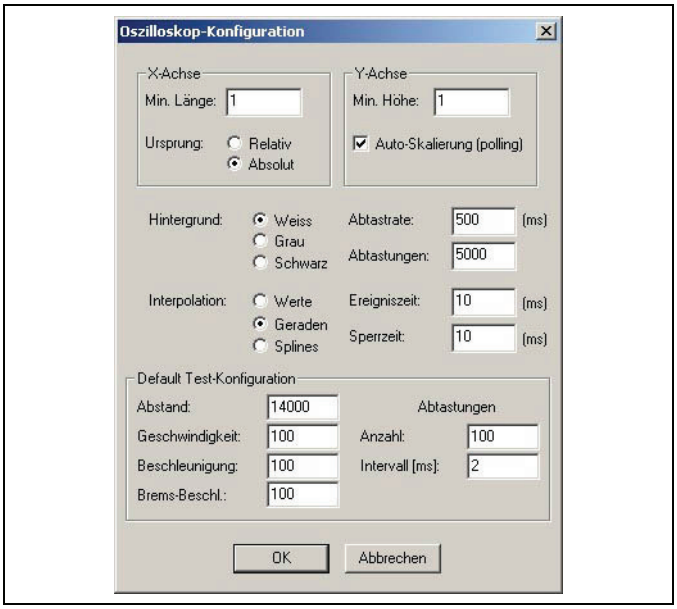
Ein TESTSETP Test wird komplett durch den Anwender außerhalb des TESTSETP Oszilloskop durchgeführt. Sobald fertig, hat der Anwender diese drei Möglichkeiten:

1. Ein völlig neues TESTSETP Oszilloskop öffnen.
2. Wenn die vorangegangen Werte nicht mehr gebraucht werden, können Sie die Kurven „wieder laden". Klicken Sie dazu mit der rechten Maustaste auf den Kurvennamen in der Kurvenauswahl und wählen Sie im Popup → *Neu laden* oder → *Neu laden & anzeigen*. Die alten Werte werden verworfen und die neuen Werte geladen.

Fügen Sie eine neue TESTSETP-Kurve mit hinzu. Dies hat den Vorteil, dass Sie dann die neuen Kurven mit den alten vergleichen können, entweder visuell oder durch Hinzufügen von abgeleiteten Kurven.

☐ Oszilloskop Einstellungen

Die globalen Oszilloskop-Eigenschaften und Defaults werden mit *Einstellungen* → *Oszilloskop* gesetzt:



X-Achse

Min. Länge bestimmt die Mindestlänge der X-Achse im Grafikfenster. Dieser Wert wird benutzt, wenn das Diagramm leer ist. Ansonsten basiert die Länge der X-Achse auf den für die darzustellende Kurve notwendigen Bereich.

Ursprung bestimmt den Anfangszustand für die Funktion *Kurven auf Zeit 0 verschieben*. Bei *Relativ* wird standardmäßig die X-Achse (Zeit) verschoben, damit die Kurven mit der Zeit 0 beginnen. Bei *Absolut* wird die X-Achse nicht verschoben. Diese Einstellung wirkt auf alle Kurven.

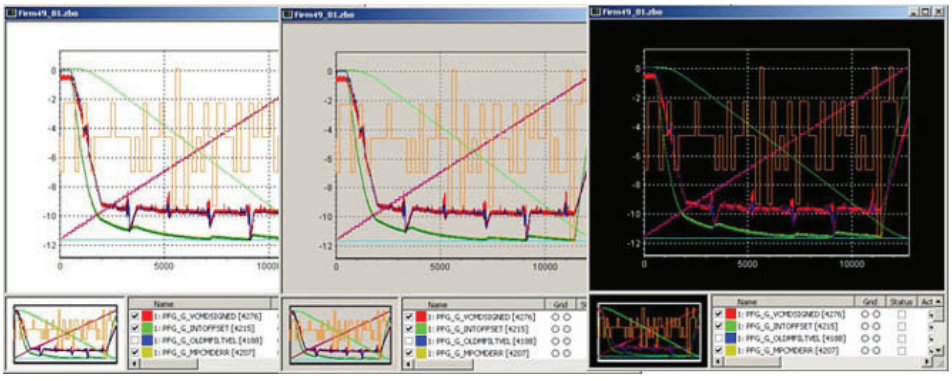
Y-Achse

Min. Höhe bestimmt die Mindesthöhe der Y-Achse, wenn neue Kurven hinzugefügt werden. Danach kann der Wert durch Setzen der *Eigenschaften* → *Achse* in der Kurvenauswahl geändert werden.

Wenn *Auto-Skalierung (polling)* aktiviert ist, wird die Y-Achse im Grafikfenster während der Oszilloskop-„Fahrt" neu berechnet. Dies verhindert, dass Kurven „außerhalb" des Grafikfensters gezeichnet werden, falls die Y-Werte außerhalb des Bereichs der aktuellen Y-Achse sind. Falls dies geschieht, wird die Y-Achse des Grafikfensters skaliert, so dass immer die gesamte Kurve im Grafikfenster sichtbar bleibt.

Hintergrund

Wählen Sie einen der drei möglichen Hintergründe für das Grafik- und das Navigationsfenster:



Interpolation

Dies ist die Default-Darstellung für Kurven, wenn sie neu hinzugefügt werden. Danach können Sie die Darstellung dieser Kurve mit *Eigenschaften* → *Anzeige* in der Kurvenauswahl ändern.

Abtastrate

Die Abtastrate legt fest, wie oft das Oszilloskop versucht Daten für „Polled" Kurven zu lesen, wenn eine Oszilloskop-„Fahrt" gestartet ist. Sie sollten darauf achten, diese Zeit nicht kürzer als unbedingt notwendig zu setzen, da es die Leistungsanforderung an den PC-Prozessor, den Prozessor der Steuerung und an das Netzwerk stark erhöht.

Abtastungen

Dies ist die Default-Anzahl der Abtastungen bei neuen Kurven, wenn sie hinzugefügt werden. Dieser Wert ist die *Maximale Anzahl* der *Abtastungen*, die das Oszilloskop für eine Kurve speichert, bevor ältere Punkte gelöscht werden, um Platz für neue Punkte zu schaffen.

Ereigniszeit / Sperrzeit

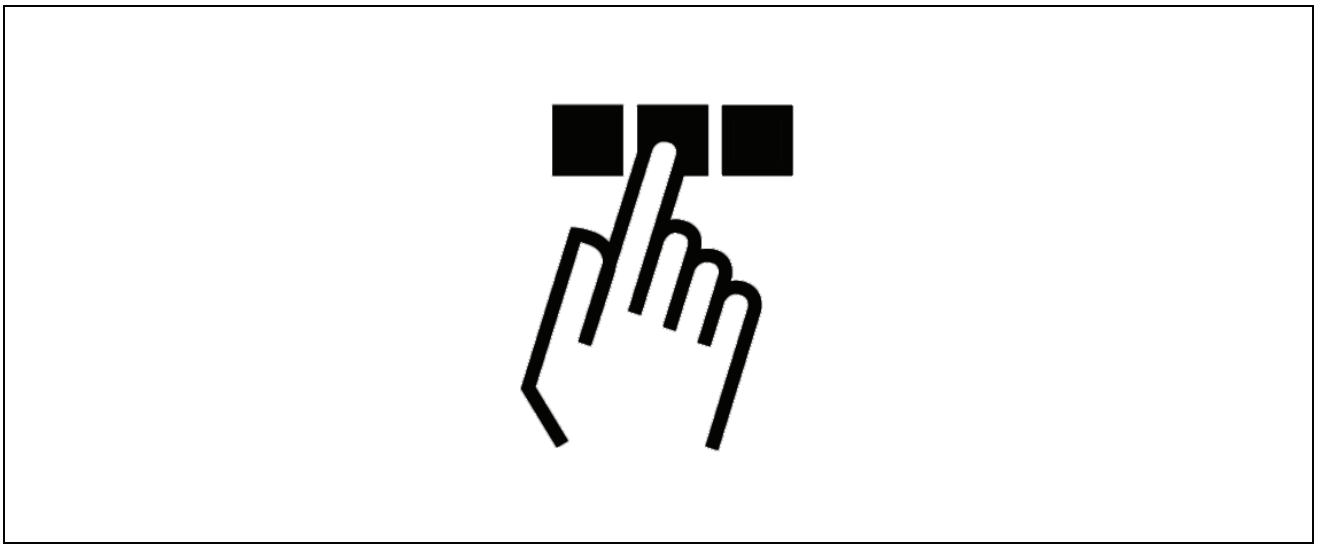
Default PDO Ereigniszeit und PDO Sperrzeit, die benutzt werden, wenn neue Kurven hinzugefügt werden.

Nachdem eine Kurve hinzugefügt wurde, können Sie diese Werte *Eigenschaften* → *Quelle* in der Kurvenauswahl ändern.

Default Test-Konfiguration

Diese Parameter bestimmen die Default-Werte die vom *Tune Oszilloskop* benutzt werden.

Programmieren mit APOSS



MCO mit der APOSS Makrosprache programmieren

Die folgenden Kapitel beschreiben, wie Sie mit der APOSS Makrosprache programmieren. Anfänger lesen bitte die Grundlagen 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.

In Gruppen wie Befehle zur Initialisierung, Steuerungsbefehle, usw. sind alle Befehle kurz beschrieben inklusive der Syntax und den Parametern. In der „Befehlsreferenz" im gleichnamigen Handbuch sind alle Befehle alphabetisch geordnet ausführlich beschrieben und mit kurzen Beispielen ergänzt. Wie diese genutzt werden, zeigen die Programmbeispiele in der Online-Hilfe.

In der APOSS-Benutzeroberfläche finden Sie alle Funktionen zum Programmieren in den Menüs *Datei* und *Bearbeiten, Entwicklung, usw.* 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.

Und in der Parameter-Referenz sind alle Parameter beschrieben, zuerst in der alphabetischen Übersicht und im Anschluss daran im Detail.

Grundlagen

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
```

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

```
SET POSERR 100000000        // Parameter
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
```

Funktion, Deklaration und Pragma Anweisung

Deklarationen und Definitionen der Funktion können überall im Programmcode stehen, außer im Unterprogrammbereich. Pragma-Anweisungen, falls vorhanden, müssen direkt nach der DIM Anweisung stehen.

Das heißt, ein Programm kann so aussehen:

```
DIM Anweisung
Pragmas
...
Variablen Deklarationen
...
Funktion Deklarationen
...
Programmcode
...
Funktion Definitionen
...
SUBMAINPROG
Subprogramm Definitionen
...
ENDPROG
```


MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

Oder es könnte auch so aussehen:

```
DIM Anweisung
Pragmas
...
SUBMAINPROG
Subprogramm Definitionen
...
ENDPROG
Variablen Deklarationen
...
Funktion Deklarationen
...
Programmcode
...
Funktion Definitionen
...
```

Oder es könnte so aussehen:

```
DIM Anweisung
Pragmas
...
Variablen Deklarationen
...
Funktion Definitionen
...
Programmcode
...
SUBMAINPROG
Subprogramm Definitionen
...
ENDPROG
```

Definitionen der Funktion können also überall im Programmcode stehen genauso wie die Deklarationen der Variablen oder Funktion.

ACHTUNG!:
Funktion, Deklaration und Pragma Anweisungen werden ab Compiler und Firmware MCO 5.00 unterstützt.

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
    RINT "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
```


MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

□ 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 in Abschnitt „Programme debuggen" auf Seite 89.

□ Interrupts

Generell gibt es diese Sorten von Interrupts:

ON INT	Interrupt bei Flanken eines Eingangs (flankengetriggert).
ON posint	Positions-Interrupt (ON APOS, …).
ON posint SETOUT (TOIN)	Simuliert eine CAM-Box (alle Arten von POSINTs).
ON PERIOD / ON TIME	Interrupt nach Ablauf einer Zeitspanne.
ON COMBIT / ON STATBIT	Interrupt wenn Bit n gesetzt wird.
ON KEYPRESSED	Interrupt wenn eine Taste des LCP gedrückt oder losgelassen wird.
ON PARAM	Interrupt wenn sich ein Parameter n ändert.
ON CANMSG	Interrupt bei Eintreffen einer CAN Nachricht.

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 posint (ON APOS, ON MAPOS, ON MCPOS, ON IPOS, ON MIPOS) vor
ON COMBIT vor
ON STATBIT vor
ON PARAM vor
ON CANMSG vor
ON KEYPRESSED vor
ON TIME / ON 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

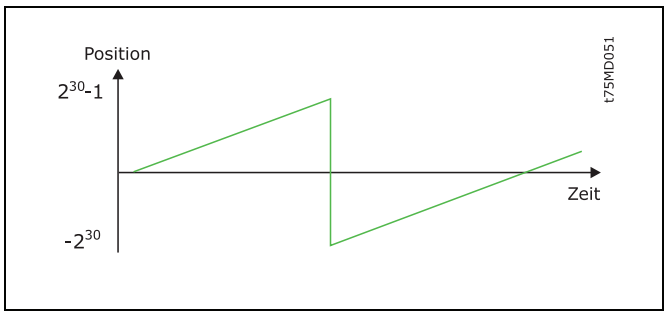
APOSS Zahlenformate

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]

2-Dimensionale Arrays

Zweidimensionale Arrays werden ab MCO 5.00 unterstützt:

```
DIM test[10][100]
```

erzeugt ein Array mit insgesamt 1000 Elementen. (Beachten Sie, dass es im Array Editor und in der zbc-Datei als ein eindimensionales Array mit der Größe 1000. erscheint.) Dieses Array kann wie folgt adressiert werden:

```
lin = 1
col = 1
wert = 1
WHILE(lin <= 10) DO
    WHILE(col <= 100) DO
     test[lin][col++] = wert++
    ENDWHILE
    lin++
ENDWHILE
```

Array Kopieren

APOSS erlaubt das Kopieren kompletter Arrays in einen einzelnen Befehl:

```
DIM test[100], feld[200]
...
test[] = feld[]
```

Für den Fall, dass die Array-Größe differiert, begrenzt das kleinere der beiden Arrays den Kopierprozess. So werden in dem oben gezeigten Fall nur die ersten 100 Longs von „feld" in „test" kopiert. Dies funktioniert für lokale Arrays (LONG, DOUBLE) genauso.

Variable Arrays

Zusätzlich zu DIM Arrays, die es außerhalb von APOSS-Programmen gibt und im Flash gespeichert and von APOSS gelesen werden können, ist es auch möglich, Arrays in der gleichen Weise zu definieren wie Variablen. Diese Arrays existieren nur innerhalb des Anwendungsprogramms und werden nach dem Verlassen des Programms wieder freigegeben. Solche Arrays werden durch normale Deklarations-Anweisungen für Variablen definiert.

```
long aTest[100], bTest[10]
```

Solche Arrays werden wie Variablen gespeichert. Sie können auch lokal innerhalb von Funktionen definiert werden, müssen sie sich aber dann im Stack befinden. Achten Sie daher darauf, dass der Stack nicht überläuft. (Siehe Compiler Optionen).

ACHTUNG!:
Double Arrays werden nicht unterstützt.

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.

Switch Anweisung

Switch Anweisungen werden ab MCO 5.00 unterstützt.

Folgende Ausdrücke sind gültig:

```
#define WITH 1
#define WITHOUT 2
LONG val

val = WITH

switch (val)
    case  WITH:  var = var + 1
                 break
    case WITHOUT: var = var - 1
                  // Bedingung nicht erfüllt
    default :     var = var % 2
endswitch
```

Break Anweisung

Break Anweisung werden ab MCO 5.00 unterstützt.

Die Break-Anweisung wird nicht nur für Switch-Anweisungen (siehe oben) verwendet, sondern auch um WHILE und REPEAT Loops zu verlassen. Wird die Break-Anweisung in einer in einer solchen Schleife benutzt, wird der Loop verlassen. In verschachtelten Loops wird nur der innere Loop verlassen.

Pragmas

Derzeit gibt es zwei Pragma Direktiven:

#pragma NOIMPLICIT
#pragma IMPLICIT

NOIMPLICIT teilt dem Compiler mit, dass implizite Variablen-Deklarationen nicht mehr erlaubt sind. Das heißt, dass jede Variable deklariert werden muss, bevor sie benutzt wird. IMPLICIT teilt dem Compiler mit, dass implizite Variablen-Deklarationen wieder erlaubt sind. Deklarations-Anweisungen sehen wie folgt aus:

```
long a,b,c,d
double u,v,y,z
     // x ist nicht erlaubt, weil x für Achsenkennung benutzt wird x(n)
```

Es wird empfohlen NOIMPLICIT zu benutzen, da dies die Bildung von ungewünschten Definitionen von neuen LONG-Variablen durch Fehler vermeidet (z.B. ein falsch geschriebener Variablenname).

ACHTUNG!:
Pragmas werden ab MCO 5.00 unterstützt.

Variablen Typen

Variablen Typen stehen ab MCO 5.00 zur Verfügung.

APOSS unterstützt die Variablen des Typs LONG und DOUBLE. LONG Variablen können implizit definiert werden, einfach durch das Benutzen derselben (außer wenn das NOIMPLICIT Pragma benutzt ist). DOUBLE Variablen müssen vor ihrem Einsatz deklariert werden.

Die Deklaration solcher Variablen sieht so aus:

```
long a,b,c,d
double u,v,y,z
     // x ist nicht erlaubt, weil für die Achsenkennung benutzt x(n)
```

Die Implementierung von DOUBLE Variablen ist abhängig von der Hardware und Firmware. MCO 305 (Firmware > 5.00) nutzt 64 Bit Genauigkeit (einige Double Funktionen wie sqrt, sin, cos .. jedoch nur 32 Bit Genauigkeit).

Zusätzlich werden die folgenden DOUBLE Funktionen unterstützt:

sqrt, sin, cos, tan, acos, asin, atan, grad, rad, ln, exp sowie die Konstante PI. Diese Funktionen besitzen für einige Hardware nur 32-Bit Genauigkeit (MCO 305). Sie können wie folgt benutzt werden:

y = sqrt(x) // gibt die Quadratwurzel von x zurück
y = sin(x) // gibt den Sinus von x zurück (entsprechend für cos, tan, acos, asin, atan)
y = grad(x) // konvertiert x (Bogenmaß) in Grad (0..360)
y = rad(x) // konvertiert Grade in Bogenmaß (0.. 2 *pi)
x = pi * 0.5 // Pi ist 3.1415….
y = ln (x) // gibt den natürlichen Logarithmus von x zurück
y = x exp u // gibt x hoch u (x^u) zurück

Weitere Double Funktionen:
FABS gibt den absoluten Gleitkommawert eines Gleitkomma-Arguments zurück:

```
double dx, dy
dy = -100.00 % 15.0  // wird -6.6666..
dx = fabs(dx)  // gibt 6.6666.. zurück
```

ROUND rundet das Gleitkomma-Argument auf den nächstmöglichen Integerwert auf oder ab. Diese Funktion gibt noch ein Double zurück:

```
dx = round(dy)  // wird -7.0000
```

Der RND Befehl für Ganzzahlen wurde korrigiert und arbeitet nun auch für negative Zahlen.

```
a = -100
b = a rnd 15  // liefert nun korrekt –7
```

Weiterhin wird ein Cast Operator unterstützt. So können Sie explizit Ausdrücke umwandeln, z.B.:

```
a = ((long) y) – 4
y = (double) a
```

Falls notwendig wandelt der Compiler intern Ausdrücke um. Wenn z.B.

```
y = a * 2.0
```

geschrieben wird, wobei a eine Long Variable und y eine Double Variable ist, wird das a in Double gewandelt, bevor die Multiplikation ausgeführt wird. Aber seien Sie vorsichtig! Wenn Sie

```
y = a % 2
```

schreiben, wird das Integer a zuerst durch 2 geteilt (Integer Arithmetik) und dann das Ergebnis nach Double konvertiert, bevor es dem y zugewiesen wird..

Funktionen

APOSS unterstützt Funktionen und lokale Variablen. Funktionen können vom Typ LONG oder DOUBLE sein. Zulässig sind maximal 100 Parameter. Funktionen müssen deklariert sein, bevor sie benutzt werden. Entweder wird die Funktion zuerst deklariert oder es wird eine Deklarations-Anweisung benutzt (siehe Beispiele unten).

Funktionen müssen immer einen Rückgabetyp von entweder LONG oder DOUBLE beinhalten. VOID Funktionen werden nicht unterstützt.

Der Anweisungsteil der Funktion muss in geschweiften Klammern stehen.

Alle Variablen, die innerhalb einer Funktion definiert sind, sind lokal. Das heißt, sie sind außerhalb der Funktion nicht zu sehen.

Lokale Variablen werden temporär im Stack gespeichert. Daher kann das Definieren vieler lokaler Variablen oder lokaler Arrays zum Überlaufen des Stacks führen (siehe Stack-Größe in Compiler Optionen).

```
long test_func (long a, long b, double c) ;  // Deklarations-Anweisung
…
wert = test_func(a,b,c)    // Einsatz der Funktion
…
long test_func(long a, long b, double c)  // Funktions-Definition
{
    long result, tmpval                 // local variables
    ….
    return(result)                      // Rückgabe des Wertes der Funktion
}
```

Funktionen können LONG oder DOUBLE Variablen (oder Ausdrücke) mittels „Wert” als Parameter handhaben. Das heißt, dass jene Parameter innerhalb der Funktion lokal definiert sind.

Arrays können auch ausgeführt werden, und zwar mittels „Referenz”. Dies ermöglicht es, das ursprüngliche Array innerhalb der Funktion zu modifizieren.

```
DIM test[100]
long modi(long[] myarray, long index)
{
    myarray[index] = myarray[index] * 10
    return(index)
}
…
modi(test,2)       // Element 2 von test wird mit 10 multipliziert
```

ACHTUNG!:
Funktionen werden ab MCO 5.00 unterstützt.

MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

☐ Arithmetik

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
++	plus 1	expr++ = expr + 13	Post-Inkrement (= Addition + 1)
–	minus	9 – 3 = 6	Subtraktion
––	minus 1	expr–– = expr – 13	Post-Dekrement (= Subtraktion – 1)
*	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 Integer-division)
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

Post-Inkrement

Der Compiler unterstützt Post-Inkrement und Post-Dekrement für Long und Double Variablen. Daher sind nachfolgende Ausdrücke erlaubt. (Abhängig nur von der Compiler-Version; das kompilierte Programm läuft mit neuen und alten Firmware-Versionen.

```
while(index < 100) do
   test[index] = index++
endwhile
value–
```

Dies funktioniert auch mit Arrays wie im Beispiel gezeigt:

Hier wird der Wert von testarray[5] in var kopiert und danach der Wert von testarray[5] um eins erhöht.

```
DIM testarray[20]
…
var = testarray[5]++
testarray[10]-
```

ACHTUNG!:
Der Operator Post-Inkrement wird ab MCO > 5.00 unterstützt.

Double Precision Operatoren

Alle Operatoren (+,-,*,%,++,--) stehen ab MCO 5.00 als Double Precision (mit doppelter Genauigkeit) zur Verfügung.

ACHTUNG!:
Bis zu diesen Compiler- und Firmware-Versionen waren alle Arithmetik-Operationen Integer-Operationen.

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
.i	Bit	expr1.i expr2 7.i1 = 1	Wie oben, nur ausführliche Schreibweise	1 … 32
.b	Byte	expr1.b expr2 0x027F.b1 = 127 0x027F.b2 = 2	Liefert das Byte expr2 von expr1 zurück	1 - 4
.ub		wert = 0x1FFFE b = value.ub1 // b ist 254	Liefert „Byte” ohne Vorzeichen	
.sb		wert = 0x1FFFE b = value.sb1 // b ist -2	Liefert „Byte” mit Vorzeichen	
.w	Word	expr1.w expr2 0x0010FFFF.w2 = 16	Liefert das Wort expr2 von expr1 zurück	1 - 2
.uw			Liefert „Word” ohne Vorzeichen	
.sw			Liefert „Word” mit Vorzeichen	
.l	Long	expr1.l expr2	Liefert das Long expr2 von expr1 zurück (Standard)	

Neue Bit Operatoren

Folgende weitere Bit Operatoren stehen zur Verfügung:

.ub, .uw, .sb, .sw

Die Bit Operatoren .b und .w liefern immer Ergebnisse ohne Vorzeichen. Mit diesen Operatoren können Sie explizit wählen, ob die Ergebnisse mit oder ohne Vorzeichen geliefert werden sollen.

ACHTUNG!:
Die neuen Bit Operatoren werden ab MCO 5.00 unterstützt.

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 von links nach rechts abgearbeitet.

Die Prioritäten sind in absteigender Folge erläutert:

exp						(exponential)
*	%	mod	rnd	.	BITP	(multiplikativ) BITP = Bit-Ausdruck, z.B. .b oder .u
+	–					(additiv)
>>	<<					(bitweises Schieben/shiften)
>=	<=	>	<			(Relation)
==	!=					(Gleichheit/Equality)
&						(bitweises und)
^						(exklusiv oder, binär Operation)
|						(bitweises inklusive oder)
AND						(logisches und)
OR						(logisches oder)

□ Preprozessor

Der Preprozessor ist dazu da, verschiedene Programm-Modifizierungen durchzuführen, bevor das Programm tatsächlich kompiliert wird. Seine wichtigsten Funktionen sind:

1. Handhabung der Folgezeilen.
2. Einfügen von „Header Dateien" oder anderen normalen Code-Blöcken, die in anderen Dateien gespeichert wurden.
3. Texteinsetzungen (Makro-Umwandlung).
4. Bedingtes Kompilieren.

ACHTUNG!:
Die Preprozessor Befehle müssen immer in Kleinbuchstaben geschrieben sein.

□ Folgezeilen

Jede Zeile, die mit einem „\" endet, wird in der nächsten Zeile fortgesetzt.

Beispiel:

```
PRINT " apos = ",apos,\
      " avel = ",avel
```

Ist das gleiche wie:

```
PRINT " apos = ",apos," avel = ",avel
```

□ #include

Dateien können in andere Dateien mit der folgenden Anweisung eingebettet werden:

```
#include "file"
#include <file>
```

wobei *"file"* der Name der Datei ist, die eingefügt wird und das Verzeichnis beinhalten kann. Für Verzeichnisse kann entweder „\" oder „/"benutzt werden.

Wenn Anführungszeichen benutzt werden, sucht der Preprozessor nach der Include-Datei im gleichen Verzeichnis, in dem die einzufügende Datei gefunden wurde. Das ist normalerweise das gleiche Verzeichnis, in dem die ursprüngliche .m Datei gefunden wurde. Es könnte jedoch ein anderes Verzeichnis sein, wenn verschachtelte Include-Dateien benutzt werden, die in verschiedenen Verzeichnissen stehen. Wird die Include-Datei nicht in diesem Verzeichnis gefunden, dann wird im APOSS „System Include" Verzeichnis gesucht; dies ist ein Unterverzeichnis des APOSS Installationsverzeichnisses.

Werden die Klammern benutzt, sucht der Preprozessor nach der Include-Datei nur im APOSS „System Include" Verzeichnis. Beispiele:

```
#include "defines.m"
#include "Library/Structures.m"
#include "Library\Structures.m"
#include <System.m>
```

ACHTUNG!:
Bitte beachten Sie, dass die Syntax der #include Anweisung für APOSS IDE Versionen vor Version MCO 5.00 anders ist. Davor enthielt die Anweisung keine Anführungszeichen. Wenn ein „altes" Programm mit dem neuen geöffnet wird, werden die Anführungszeichen automatisch hinzugefügt.

MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

☐ #define, #undef

Einfache Zeichenfolgen (Strings) werden mit folgender Anweisung ersetzt:

`#define` *name string*

Alle Vorkommen von *„name",* die der Anweisung folgen, werden durch *„string"* ersetzt.

Beachten Sie, dass *„string*" alle Zeichen (einschließlich der Leerzeichen) bis zum Ende der Zeile ersetzt. Falls notwendig kann die Zeile durch „\" für die Folgezeilen fortgesetzt werden.

Definitionen können mit folgender Anweisung wieder gelöscht werden:

`#undef` *name*

Die folgenden Namen sind vordefiniert:

`__DATE__` - Datum, wann das Programm kompiliert wurde (als Zeichenfolge).
`__TIME__` - Uhrzeit, wann das Programm kompiliert wurde (als Zeichenfolge).
`__FILE__` - Dateiname des Programms (als Zeichenfolge).
`__LINE__` - Aktuelle Zeilennummer im Programm (als Nummer).

Beispiel:

```
#define ARRAY_LEN 100
#define PRINT_MSG print " apos = ",apos," avel = ",avel
DIM prm[ARRAY_LEN]
PRINT_MSG
PRINT "Dies ist Zeile ",__LINE__
```

Ist das gleiche wie:

```
DIM prm[100]
PRINT " apos = ",apos," avel = ",avel
PRINT " Dies ist Zeile ",5
```

String-Ersetzungen werden rekursiv durchgeführt. Sobald der Originalstring ersetzt wurde, prüft der Preprozessor nach anderen Ersetzungen innerhalb des neuen Strings. Beachten Sie, dass String-Ersetzungen nicht innerhalb der **#define** Anweisung selbst und nicht bei Strings innerhalb von Anführungszeichen durchgeführt werden.

Beispiel:

```
#define wert 10
#define PRINT_MSG print "Wert = ",Wert
PRINT_MSG
#define wert 20
PRINT_MSG
```

ist das gleiche wie:

```
PRINT "Wert = ",10
PRINT "Wert = ",20
```

Define-Anweisungen können auch ein oder mehrere Argumente, getrennt durch Komma, enthalten:

`#define` *name*`(`*arg1*,*arg2*`)` *string*

In diesem Fall werden die Vorkommen von *„arg*" durch *„string*" ersetzt.

Beispiel:

```
#define SQUARE(x) ((x)*(x))
#define SUM_SQUARE(x,y) (SQUARE(x) + SQUARE(y))
i = SUM_SQUARE(3,4)
i = SUM_SQUARE(a,b)
```

ist das gleiche wie:

```
i = (((3)*(3)) + ((4)*(4)))
i = (((a)*(a)) + ((b)*(b)))
```

□ #ifdef, #ifndef, #if

Die bedingte Kompilierung des Programmcodes ist durch Anwendung einer der folgenden Anweisungen möglich:

`#ifdef` *name* – wenn "*name*" mit der Anweisung **#define** definiert ist
`#ifndef` *name* – wenn "*name*" nicht mit der Anweisung **#define** definiert ist.
`#if` *Integer-Ausdruck* - wenn "*Integer-Ausdruck*" als ungleich 0 ausgewertet wird.

Diesen Anweisungen kann jede beliebige Anzahl von Anweisungen folgen und dann **#endif**.
#else kann auch benutzt werden, wie in folgendem Beispiel gezeigt.

Beispiel:

```
#ifdef INCLUDE_DEBUG
PRINT "Position = ",apos
PRINT "Geschwindigkeit = ",avel
#endif

#ifdef START_AT_0
start = 0
#else
start = 1
#endif

#if 0
Dies wird nie kompiliert.
#endif

#if 1
Dies wird immer kompiliert.
#endif

#define OPTION 1
#if OPTION > 2
Dies wird nur dann kompiliert, wenn OPTION
größer als 2 ist.
#endif
```

Wenn **#if** verwendet wird, kann **#elif** (für „else if”) benutzt werden.

Beispiel:

```
#define CUSTOMER_A 1
#define CUSTOMER_B 2

#define CUSTOMER CUSTOMER_B

#if CUSTOMER == CUSTOMER_A
Für Kunde A kompilieren.
#elif CUSTOMER == CUSTOMER_B
Für Kunde B kompilieren.
#else
Ansonsten kompilieren.
#endif
```

□ APOSS Befehlsgruppen

Ausführliche Details zu den Befehlen finden Sie in der „MCO 305 Befehlsreferenz" in alphabetischer Reihenfolge 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)

Befehl	Beschreibung	Syntax	Parameter
DEFCORIGIN	Sollposition als Nullpunkt setzen.	DEFCORIGIN	–
DEFMORIGIN	Aktuelle Master-Position als Nullpunkt für den Master setzen	DEFMORIGIN	–
DEFORIGIN	Istposition als Nullpunkt setzen	DEFORIGIN	–
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	–
RSTORIGIN	Temporärnullpunkt löschen	RST ORIGIN	–
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	
SETMORIGIN	Aktuelle Position als Nullpunkt für den Master setzen	SETMORIGIN wert	wert = absolute Position
SETORIGIN	Temporärnullpunkt setzen	SET ORIGIN p	p = absolute Position

□ Steuerungsbefehle

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

Befehl	Beschreibung	Syntax	Parameter
CONTINUE	Abgebrochene Positionier- und Drehzahlbefehle 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 Unterprogramms
GOTO	Sprung zu einem Programmlabel	GOTO label	label = Zielposition
IF THEN	Bedingte einfache Programmverzweigung	IF Bedingung THEN Befehl	Bedingung = Verzweigungskriterium
… 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 Schleifenwiederholungen 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 Unterprogramms
… 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

MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

Befehl	Beschreibung	Syntax	Parameter
ENDWHILE	While-Schleife Ende	ENDWHILE	–
#INCLUDE	Einfügen des Inhalts einer Datei	#INCLUDE file	file = Name der Datei, die eingefügt wird

□ Ein-/Ausgangs-Befehle (I/O)

Befehle zum Setzen und Rücksetzen der Ausgänge und Abfragen der Eingänge (Gruppe I/O)

Befehl	Beschreibung	Syntax	Parameter
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)
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

□ Systemstatus-Funktionen (SYS)

Befehle zum Abfragen von Systemstatus-Informationen wie Position und Geschwindigkeit des Antriebs, Systemtakt, Fehlerstatus usw. Befehle zum Konfigurieren und Einsatz von Arrays für die Aufzeichnung von Testergebnissen.

Befehl	Beschreibung	Syntax	Parameter
APOS	Istposition lesen	erg = APOS	–
APOSDIFF	Overflow-Handling von Inkrementalgebern in Anwendungen.	erg = APOSDIFF oldpos	oldpos = APOS zu einem früheren Zeitpunkt
AVEL	Aktuelle Geschwindigkeit der Achse abfragen	erg = AVEL	–
AXEND	Status der Programmausführung abfragen	erg = AXEND	–
CPOS	Sollposition lesen	erg = CPOS	–
CPOSDIFF	Overflow-Handling von Inkrementalgebern in Anwendungen.	erg = CPOSDIFF oldpos	oldpos = CPOS zu einem früheren Zeitpunkt
ENCPOSOFFS	Synchronisiert den inkrementalen Positionszähler mit dem absoluten im Encoder.	erg = ENCPOSOFFS offset	offset = Liefert die Differenz zwischen der absoluten und inkrementalen Position
ENCTGREAD	Liest ein RS485 Telegramm vom Encoder.	erg = ENCTGREAD array	array = Benutzer-Array, in das die erhaltene Datenmenge geschrieben werden soll.

Befehl	Beschreibung	Syntax	Parameter
ENCTGWRITE	Sendet ein RS485 Telegramm zum Encoder.	erg = ENCTWRITE länge array	länge = Anzahl der zu sendenden Bytes (im Benutzer-Array). array = Benutzer-Array, das die zum Encoder zu sendende Datenmenge enthält.
ERRNO	Fehlernummer lesen	erg = ERRNO	–
IPOS	Letzte Index- bzw. Markerposition des Slaves abfragen.	erg = IPOS	–
IPOSDIFF	Overflow-Handling von Inkrementalgebern in Anwendungen.	erg = IPOSDIFF oldpos	oldpos = IPOS zu einem früheren Zeitpunkt
JERKFINVEL	Berechnet die Endgeschwindigkeit für einen ruckbegrenzten Stopp mit maximaler Beschleunigung/Verzögerung.	erg = JERKFINVEL	–
JERKSTOPDIST	Berechnet die notwendige Distanz für einen ruckbegrenzten Stopp mit maximaler Verzögerung.	erg = JERKSTOPDIST dec	dec = setzt die Verzögerung in % oder VELRES Einheiten
MAPOS	Aktuelle Istposition des Masters abfragen.	erg = MAPOS	–
MAPOSDIFF	Overflow-Handling von Inkrementalgebern in Anwendungen.	erg = MAPOSDIFF oldpos	oldpos = MAPOS zu einem früheren Zeitpunkt
MAVEL	Aktuelle Geschwindigkeit des Masters abfragen.	erg = MAVEL	–
MENCPOSOFFS	Synchronisiert den inkrementalen Positionszähler mit dem absoluten im Encoder.	erg = MENCPOSOFFS offset	offset = Liefert die Differenz zwischen der absoluten und inkrementalen Position
MENCTGREAD	Liest ein RS485 Telegramm vom Encoder.	erg = MENCTGREAD array	array = Benutzer-Array, in das die erhaltene Datenmenge geschrieben werden soll.
MENCTGWRITE	Sendet ein RS485 Telegramm zum Encoder.	erg = MENCTGWRITE länge array	länge = Anzahl der zu sendenden Bytes (im Benutzer-Array). array = Benutzer-Array, das die zum Encoder zu sendende Datenmenge enthält.
MIPOS	Letzte Index- bzw. Markerposition des Masters abfragen.	erg = MIPOS	–
MIPOSDIFF	Overflow-Handling von Inkrementalgebern in Anwendungen.	erg = MIPOSDIFF oldpos	oldpos = MIPOS zu einem früheren Zeitpunkt
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
TESTSETDEST	Speicherbereich in dem eine Datenaufzeichnung abgelegt wird, festlegen.	TESTSETDEST Arrayname	Arrayname = Name des Arrays, das für die Aufzeichnung verwendet wird oder Schlüsselwort DYNMEM um in den freien Speicherplatz aufzuzeichnen.
TESTSETINDEX	Systemvariablen für die Datenaufzeichnung festlegen.	TESTSETINDEX svi1, svi2, svi3, svi4, ..., svin	svi1…svin: Indizes der max. 20 Systemvariablen (sysvar[...]), deren Werte aufgezeichnet werden sollen.
TESTSETP	Aufzeichnungsdaten für Testfahrt festlegen	TESTSETP ms vi1 vi2 vi3 arrayname	ms = Intervall in ms vi 1 … 3 = Indizes der Werte, die aufgezeichnet werden sollen arrayname = Array, das für die Aufzeichnung benutzt wird
TESTSETTIME	Abtastperiode für die Datenaufzeichnung konfigurieren.	TESTSETTIME ms	ms = Abtastperiode in ms
TESTSETTYPE	„Einmalige" oder „zyklische" Aufzeichnung festlegen.	TESTSETTYPE typ	typ: 0 = Einmalige Aufzeichnung 1 = Zyklische Aufzeichnung
TESTSTART	Aufzeichnung der Testfahrt starten	TESTSTART anz	anz = 0 Die max. Anzahl der in den verfügbaren Speicher passenden Abtastpunkte wird aufgezeichnet anz > 0 Anzahl der aufzuzeichnenden Abtastpunkte
TESTSTOP	Datenaufzeichnung beenden	TESTSTOP methode	methode: 0 = beendet sofort die Datenaufzeichnung.
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 CANINPUT id GOSUB	Unterprogramm aufrufen, wenn ein CAN-Telegramm vom Typ 'id' ankommt.	ON CANINPUT id GOSUB name	id = 0 name = Unterprogramm
ON CANMSG GOSUB	Eintreffen einer gepufferten CAN-Nachricht.	ON CANMSG GOSUB name	name = Unterprogramm
ON COMBIT .. GOSUB	Unterprogramm aufrufen, wenn Bit n des Kommunikationspuffers gesetzt ist	ON COMBIT n GOSUB name	n = Bit n des Kommunikationspuffers name = Unterprogramm

Befehl	Beschreibung (Gruppe INT)	Syntax	Parameter
ON DELETE .. GOSUB	Löscht einen Positions-Interrupt: ON posint GOSUB.	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 KEYPRESSED GOSUB	Unterprogramm aufrufen, wenn eine Taste gedrückt oder losgelassen wird.	ON KEYPRESSED GOSUB name	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 = n > 20 ms (Zeit für Wiederaufruf) n = 0 (Funktion abschalten) name = Unterprogramm
ON posint .. GOSUB	Unterprogramm aufrufen, wenn ein Positions-Interrupt auftritt: ON APOS = wenn die Slave-Position xxx passiert ist ON IPOS = wenn der Abstand zwischen der letzten Markerposition und der Istposition erreicht ist ON MAPOS = wenn die Master-Position xxx [qc] passiert ist ON MCPOS = wenn die Master-Position xxx [MU] passiert ist ON MIPOS = wenn der Abstand zwischen zwei Markern erreicht ist	ON sign postype position GOSUB name	sign= Fahrtrichtung + = steigende Flanke oder wenn die Position in positiver Richtung passiert wurde – = fallende Flanke oder wenn die Position in negativer Richtung passiert wurde postype = APOS, IPOS, MAPOS MCPOS, MIPOS position = abhängig vom Befehl in Benutzereinheiten [BE], Master Benutzereinheiten [MU] oder Kurveneinheiten [CU] name = Unterprogramm
ON posint SETOUT	Simuliert ein Nockenschaltwerk (alle Typen von Positions-Interrupts: APOS, IPOS, MAPOS, MCPOS, MIPOS)	ON +/- typ position SETOUT outno	typ = alle POSINT position = abhängig vom Befehl in Benutzereinheiten [BE], Master Benutzereinheiten [MU] oder Kurveneinheiten [CU] outno = beliebige gültige Nummer eines Ausgangs (oder negative Nr. des Ausgangs)
ON posint SETOUT (TOIN)	Simuliert ein Nockenschaltwerk (alle Typen von Positions-Interrupts: APOS, IPOS, MAPOS, MCPOS, MIPOS)	ON +/- typ position SETOUT outno TOIN inno	typ = alle POSINT position = abhängig vom Befehl in Benutzereinheiten [BE], Master Benutzereinheiten [MU] oder Kurveneinheiten [CU] outno = beliebige gültige Nummer eines Ausgangs (oder negative Nr. des Ausgangs) inno = beliebige gültige Nummer eines Eingangs (oder negative Nummer des Eingangs)
ON STATBIT..	Unterprogramm aufrufen, wenn Bit n des Statuswortes gesetzt ist.	ON STATBIT n GOSUB name	n = Bit n des FU Status name = Unterprogramm

Befehl	Beschreibung (Gruppe INT)	Syntax	Parameter
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. (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 typ	parno = LCP Par. Nummer text = ASCII Text min = min. Wert max = max. Wert typ = Online / Offline Par.
LINKPDO	Systemvariable mit PDO verknüpfen und in die internen Parameter kopieren	LINKPDO no len indx pdo	no = rank order in RxPDO len = number of the bits to be imported indx = index of the system variable SYSVAR pdo= values between 1 .. 4 or 5 (serial)
LINKSDO	Internes Objekt in PDO kopieren	LINKSDO indx len no "text" pdo	indx = index of the system variable SYSVAR len = length of the bits to be imported no = rank order in the PDO text = " " (comment) pdo = values between 1 .. 4 -pdo = dummy entry of the link object into the mapping list
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 Drehzahl-modus 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 Geschwindig-keitswert
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 Geschwindig-keitswert

MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

☐ Synchronisationsbefehle

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

Befehle	**Beschreibung**	**Syntax**	**Parameter**
DEFSYNCORIGIN	Definiert das Verhältnis Master:Slave für den nächsten SYNCP oder SYNCM Befehl.	DEF SYNCORIGIN master slave	master = Sollposition in qc slave = Sollposition
MOVESYNCORIGIN	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	–
SYNCMARKERSTART	Setzt einen Marker oder die Markerbehandlung zurück.	SYNCMARKERSTART restarttype	restarttype = 0 or 1
SYNCP	Winkel-/Positionssynchronisation.	SYNCP	–
SYNCSTAT	Flag für Synchronisationsstatus abfragen.	erg = SYNCSTAT	
SYNCSTATCLR	Zurücksetzen der Flags MERR und MHIT.	SYNCSTATCLR wert	wert = 8 = SYNCMMHIT 16 = SYNCSMHIT 32 = SYNCMMERR 64 = SYNCSMERR
SYNCV	Geschwindigkeitssynchronisation.	SYNCV	–

☐ 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

□ CAN Basic Library

Raw Telegramm Modus

Diese Bibliothek enthält alle Funktionen zur Durchführung des „Raw CAN Telegram"-Transfers. So können Mailboxen mit den Befehlen DEFCANIN und DEFCANOUT zum Lesen und Schreiben von Telegrammen definiert werden. Diese Befehle erlauben die Auswahl der Länge und der Telegramm-ID (Standard 11-Bit-Identifier). Einmal definiert, können diese Mailboxen benutzt werden, um solche Telegramme mittels der Befehle CANOUT und CANIN zu empfangen und zu senden.

Der CANIN Befehl erlaubt dem Anwender Remote-Frames zu benutzen und zu definieren, ob er auf eine Antwort warten will oder nicht. Es kann auch der Timeout für das Lesen definiert werden.

Mit dem Befehl CANDEL können bereits definierte Mailboxen gelöscht werden.

Command	Description	Syntax	Parameter
CANDEL	Löscht alle oder einzelne CAN-Objekte	CANDEL objnr	objnr = Objektnummer, die beim Definition des Objektes zurück gegeben wurde –1= löscht alle Objekte (außer den Standardobjekten)
CANIN	Liest ein Objekt über den CAN-Bus.	status = CANIN objnr timeout control varhi varlo	objnr = Objektnummer, die beim Definition des Objektes zurückgegeben wurde timeout = –1= es wird nicht auf die Daten gewartet 0 = es wird gewartet, bis die Daten kommen > 0 = es wird *timeout* [ms]auf die Daten gewartet control = 0 = es wird geprüft, ob neue Daten gekommen sind. Anschließend werden neue Daten in die Variablen kopiert 1 = es wird ein Remote-Frame gesendet, und es wird in Abhängigkeit von *timeout* auf die Daten gewartet varhi = Bytes 0 bis 3 der CAN-Objektdaten varlo = Bytes 4 bis 7 der CAN-Objektdaten
CANOUT	Sendet ein Objekt (aktiv)	CANOUT nr valhi vallo	valhi = Byte 0 bis 3 der CAN-Objektdaten vallo = Bytes 4 bis 7 der CAN-Objektdaten
DEFCANIN	Definiert ein Empfangsobjekt	objnr = DEFCANIN id dlen	id = CAN-Identifikationsnummer dlen = Datenlänge des Objektes in Bytes (max. 8 Bytes)
DEFCANOUT	Definiert ein Sendeobjekt im CAN Controller	objnr = DEFCANOUT id dlen	id = CAN-Identifikationsnummer dlen = Datenlänge des Objektes in Bytes (max. 8 Bytes)

Gepufferte Nachrichtenübertragung

Diese Nachrichtenübertragung wird von APOSS benutzt. Falls aber APOSS nicht aktiv ist, könnte sie auch für Anwendungszwecke genutzt werden. Aber Sie müssen wissen, dass es die Handhabung von gepufferten Nachrichten erfordert, dass alle Telegramme wirklich gelesen werden. Andernfalls kann ein voller Puffer die Programmausführung stoppen, weil zum Beispiel ein OUTMSG warten muss bis der Puffer es erlaubt, wieder zu schreiben.

Es gibt drei Telegramme, die für diese Art von Kommunikation benutzt werden:

GlobalMessage	Telegramm ID = 0	8 Byte Länge
RxMessage	Telegramm ID = CANNR * 2 + 1	8 Byte Länge
TxMessage	Telegramm ID = CANNR * 2	

GlobalMessages werden von allen Steuerungen gelesen. Die ersten beiden Bytes dürfen von der Anwendung nicht benutzt werden. Diese Bytes müssen immer 0 sein oder es können unerwartete Nebeneffekte auftreten.

Die folgenden Funktionen handhaben gepufferte Nachrichten:

Command	Description	Syntax	Parameter
INGLB	Liest eine Glb-CAN-Nachricht	erg = INGLB (p)	p = maximale Wartezeit, definiert in 0 = warten bis eine Nachricht kommt > 0 = maximal p ms warten < 0 = nicht warten auf Nachricht
INMSG	Liest eine gepufferte CAN-Nachricht (Polling)	intval = INMSG timeout	timeout = < 0 = nicht auf Daten warten = 0 = auf Daten warten > 0 = timeout [ms] auf Daten warten
MSGVAL	Liefert den Long-Wert der letzten Nachricht	longval = MSGVAL	–
ON CANINPUT	Ruft ein Unterprogramm auf, wenn ein CAN-Telegramm vom Typ 'id' eintrifft.	ON CANINPUT id GOSUB name	id = 0 name = Name des Unterprogramms
OUTMSG	Sendet eine gepufferte CAN-Nachricht (Polling)	OUTMSG intval longval	intval = Byte 2 und 3 der CAN-Nachricht longval = Byte 4 und 7 der CAN-Nachricht
USRSTAT	Setzt den CAN-User-Status	USRSTAT val	val = Wert der gesetzt werden soll

SDO2

Ab MCO 5.00 wird ein zweites SDO unterstützt, um mit Debug-Einrichtungen wie einem SDO-Monitor zu kommunizieren.

Dieses zweite SDO benutzt die folgende CobId:

SDO2-Tx = 0x681 – 0x6FF
SDO2-Rx = 0x101 – 0x1FF

Um den Gebrauch von SDO2 zu aktivieren, müssen Sie zuerst die oben genannten Adressen zu den Objekten schreiben:

ServerSdo2Parameter - Index 0x1201 Subindex 0x01 (CobId Rx) und 0x02 (CobId Tx).

Wenn diese Parameter gelesen werden, erhalten Sie die oben erwähnten CobIds mit dem höchsten gesetzten Bit. Dies ist das NOT_VALID Bit.

Es ist nicht möglich diese CobIds zu ändern; sie können nur mit dem gleichen Wert mit dem NOT_VALID Bit auf null gesetzt überschrieben werden.

Es ist möglich diese SDOs durch Reset des NOT_VALID Bits wieder zu deaktivieren.

Der Gebrauch der SDO2 benötigt keine extra Objekte im CAN-Memory. Senden und Empfangen werden von der universellen Tx-Mailbox bzw. Rx-Mailbox ausgeführt.

Das SDO2 könnte auch innerhalb des APOSS-Programms aktiviert oder deaktiviert werden:

```
SDO2_txid = 0x680 + (get CANNR)
SDO2_rxid = 0x100 + (get CANNR)
COB_NOT_VALID = 0x80000000
     // enable SDO2
     sysvar[0x01120101] = SDO2_txid
     sysvar[0x01120102] = SDO2_rxid

     // disable SDO2
```

```
sysvar[0x01120101] = (SDO2_txid | COB_NOT_VALID)
sysvar[0x01120102] = (SDO2_rxid | COB_NOT_VALID)
```

□ CANopen Master Library

Diese Bibliothek bietet Funktionen um den Einsatz von CANopen-Slaves wie I/O-Module oder Drehgeber oder sogar Antriebe zu unterstützen. Diese Funktionen erlauben nicht nur die Konfiguration mit SDOs sondern auch den Einsatz von Prozessdaten durch PDOs.

SDOREAD und SDOWRITE werden benutzt, um die Object Dictionary eines beliebigen Slaves zu lesen oder zu schreiben. Auch SDOREADSEG bzw. SDOREADSEGP, die das segmentierte Lesen erlauben (entweder ungepackt oder gepackt), werden bei der Ausführung unterstützt.

Solche Befehle erlauben dem Anwender CAN-Slaves über den Bus zu konfigurieren und zu steuern. So könnte zum Beispiel ein weiterer Antrieb auf ein spezielles Modul gesetzt oder der Antrieb mit einer vorgegebenen Drehzahl gestartet werden (falls der Antrieb über eine CAN-Option verfügt).

Um den einfachen Gebrauch von externen I/Os über den CAN-Bus zu ermöglichen, wurden die normalen IN und OUT Befehle erweitert. Sobald eine Nummer größer als 256 bestimmt wird, wird automatisch ein CAN-I/O angenommen. Daher wird mit dem Befehl OUT 257 1 der Ausgang Nummer 1 auf das CAN-I/O Modul mit der Node ID 1 gesetzt.

Die Nummer des Ausgangs entspricht (Node_Id * 256 + Ausgangs-Nr.). Das gleiche gilt für INAD und OUTDA. Daher können beliebige analoge I/O-Module benutzt werden.

Des weiteren kann ein CANopen-Modul mit dem Befehl CANINI angemeldet werden. Dabei wird das Modul automatisch im Hintergrund überwacht. Dies macht auch den I/O-Traffic schneller und man kann sogar ein ON INT für einen CANopen-Eingang verwenden.

Folgende Befehle und Parameter realisieren Master-Funktionalität:

Befehl	Beschreibung	Syntax	Parameter
CANIN (>100)	Sammelt alle Transmit-PDOs von digitalen Eingangs-Modulen oder den CAN-Antriebstatus mit einem einzigen CAN-Telegramm.	erg = CANIN (id * 100) timeout control varhi varlo	id = CAN id timeout = -1 = wartet nicht auf Daten 0 = wartet bis die Daten ankommen >0 = wartet timeout [ms] auf Daten control = 0 = kopiert die neu erhaltenen Daten in die Variablen 1 = sendet einen Remote-Fame and wartet auf Daten abhängig von *timeout* varhi = Byte 0 bis 3 der CAN Objektdaten varlo = Byte 4 bis 7 der CAN Objektdaten
CANINI,	Initialisiert die notwendigen Objekte (PDOs) für den Datenaustausch mit CANopen-Teilnehmern oder aktiviert die erweiterten Funktionen CANINI und CANIN.	CANINI nr, nr, …, CANINI 999 CANINI nr, 999, …	nr = guard * (busoffset * 1000 + id) guard = -1, +1 (mit / ohne Guarding) busoffset = 100000 , 0 (Slave-Bus, Master-Bus)
SDOREAD	Liest SDO eines angeschlossenen CANopen Gerätes.	val = SDOREAD id index sub	id = CAN ID (1…127) -id = führt den Befehl ohne Warten auf eine Antwort aus index = Index des Objekts sub = Subindex (0x00 – 0xFF)

MG.33.L5.03 – VLT® ist ein eingetragenes Warenzeichen von Danfoss.

Befehl	Beschreibung	Syntax	Parameter
SDOREADSEG	Segmentiertes Lesen von SDOs (ungepackt).	res = SDOREADSEG id, index, subindex, arrayname	id = CAN ID Nummer -id = führt den Befehl ohne Warten auf eine Antwort aus index = 0x2000 subindex = Parameternummer arrayname = Name des vorhandenen Arrays
SDOREADSEGP	Segmentiertes Lesen von SDOs (gepackt).	res = SDOREADSEGP id, index, subindex, arrayname	id = CAN ID Nummer -id = führt den Befehl ohne Warten auf eine Antwort aus index = 0x2000 subindex = Parameternummer arrayname = Name des vorhandenen Arrays
SDOSTATE	Ergebnis einer aktiven Kommunikation prüfen.	res = SDOSTATE id value	id = CAN ID value = Parameterwert
SDOWRITE	Setzt SDO eines angeschlossenen CANopen Gerätes.	SDOWRITE id index sub val	id = CAN ID (1…127) -id = führt den Befehl ohne Warten auf Antwort aus index = Index des Objekts sub = Subindex val = Parameterwert

Folgende Funktionen unterstützen die Master-Funktionalität:

Befehl	Beschreibung	Syntax	Parameter
IN	Zustand eines digitalen Eingangs abfragen	res = IN n	n = Nummer des Eingangs bzw. für CANopen I/O Module: CAN-Bus + (Modul-CAN-ID * 256) + Eingangsnummer (bzw. Eingangsbyte)
INB	Zustand der digitalen Eingänge byteweise abfragen.	res = INB n	n = Eingangsbyte oder mit CANopen I/O Modulen: wie oben
OUT	Digitale Ausgänge setzen oder zurücksetzen.	OUT n s oder OUT X/n s	n = Nummer des Ausgangs bzw. für CANopen I/O Module: CAN-Bus + (Modul-CAN-ID * 256) + Ausgangsnummer (bzw. Ausgangsbyte) X/n = Klemmenblock / Pin Nummer s = Zustand (0 = Aus, 1 = Ein)
OUTB	Zustand der digitalen Ausgänge byteweise verändern.	OUTB n w	n = Ausgangsbyte oder mit CANopen I/O Modulen: wie oben w = Wert (0 ... 255)
SET ENCODERTYPE	SET Par. 32-00 *Inkrementalgeber Signaltyp.*		

□ CANopen Slave Library

Diese Bibliothek enthält alle Funktionen, die notwendig sind um als CANopen-Slave zu arbeiten. Mit spezieller Anwendungs-Software wäre es sogar möglich, sich wie ein DS402 Antrieb zu verhalten.

Hierfür ist eine komplette SDO-Dictionary mit allen Funktionen implementiert, die für den Support von PDO1..4 und dynamisches Mapping von PDOs notwendig sind, mit Übertragungstypen, die den SYNC-Transfer unterstützen sowie Sperrzeit und Ereigniszeit.

Um diese Funktionen zu benutzen darf es natürlich nur eine einzige Node-ID (par. 33-90 CANNR) auf dem Bus geben.

Diese Funktionalität erlaubt zum Beispiel eine sehr einfache Kommunikation zwischen zwei Steuerungen. So kann eine Steuerung einfach in die andere Object Dictionary schreiben und umgekehrt. Oder der PDO Befehl könnte benutzt werden um PDOs zu senden, die von allen anderen Steuerungen leicht gelesen werden können, zum Beispiel unter Verwendung der erweiterten Funktion CANINI 999. Das bietet fast unbegrenzte Möglichkeiten für eine schnelle und effiziente CAN-Kommunikation zwischen mehreren Steuerungen.

Die Slave-Bibliothek unterstützt auch ein SDO2 Objekt, mit dem APOSS über ein zweites SDO kommunizieren kann, während eine SPS (oder andere Steuerung) über SDO kommuniziert. Dies ist besonders beim Debuggen und für die Entwicklung von Anwendungen nützlich.

Die mächtigen Befehle LINKSDO und LINKPDO erlauben die Festlegung, welches SDO in ein abgehendes PDO abgebildet werden soll und welcher Teil eines ankommendes PDO automatisch in welches Objekt im Dictionary abgebildet wird (Mapping). Weil das Object Dictionary zum Beispiel auch die Benutzerparameter enthält, ist es sehr einfach diese im Hintergrund zu aktualisieren, wenn ein PDO ohne eine Aktion der Anwendung empfangen wurde.

Folgende Befehle realisieren die CANopen-Slave-Funktionalität:

Command	Description	Syntax	Parameter
LINKPDO,	Systemvariable mit RxPDO verknüpfen und in die internen Parameter kopieren oder Teil eines Arrays in das PDO verknüpfen.	LINKPDO nr len indx pdo	nr = Reihenfolge im RxPDO beginnend bei 1 len = Anzahl der Bits, die übernommen werden sollen indx = Index der Systemvariablen SYSVAR pdo = Werte zwischen 1 .. 4 oder 5 (seriell)
LINKSDO,	TxPDO mit interner Systemvariable verknüpfen oder Teil eines Arrays aus dem PDO verlinken.	LINKSDO indx len nr "text" pdo	indx = Index der Systemvariable SYSVAR len = Länge der Bits, die übernommen werden sollend nr = Reihenfolge im PDO text = " " (Kommentar) pdo = Werte zwischen 1 .. 4 -pdo = Dummy Eintrag des Link-Objektes in die Mapping-Liste
PDO[ ]	Pseudo-Array für den direkten Zugriff auf die CANopen-PDOs..	PDO[n]	n = 1001. für 1. PDO 2001. for 2. PDO … 5001. für 5. PDO (serielles PDO)
SYSVAR[0x01…]	Systemvariable (Pseudo-Array) liest Systemwerte; siehe SDO-Dictionary		

□ Verschaffen Sie sich einen Überblick über alle Programmbeispiele

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. Die meisten der der Beispiele werden auch mit der APOSS Software geliefert: Öffnen Sie diese mit *Datei* → *Beispiel*.

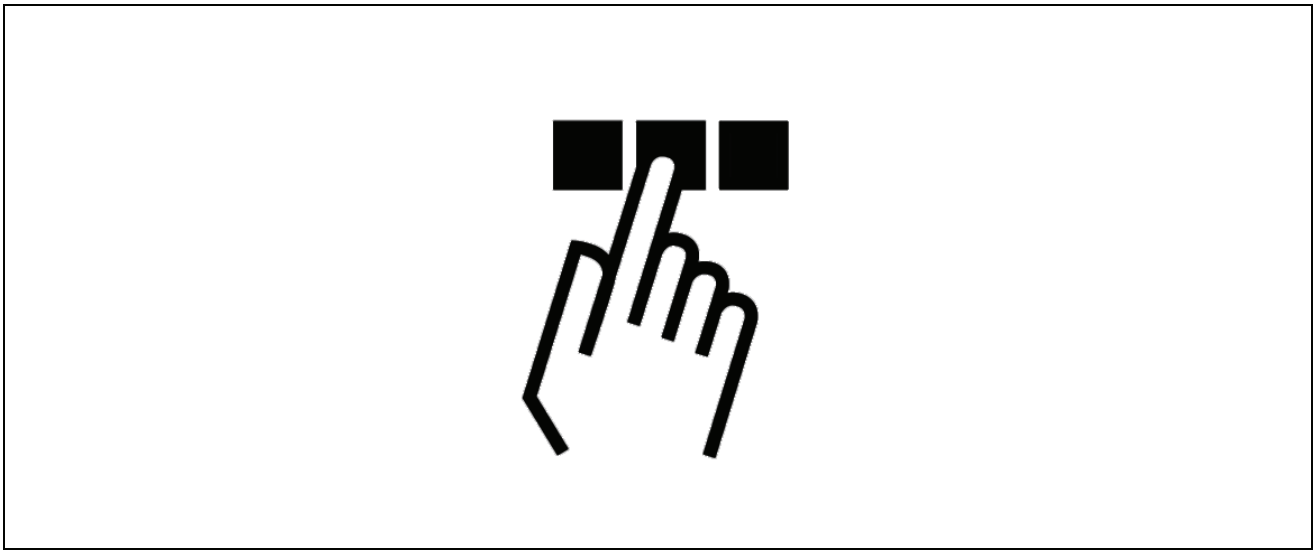
Alle Beispiele können kopiert und in die APOSS-Software eingefügt werden. Bitte lesen Sie die Sicherheitshinweise, bevor Sie die Beispiele benutzen!

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	APOSS Software	Vergleich verschiedener Positioniervorgänge bei konstanter Geschwindigkeit, aber mit unterschiedlicher Beschleunigung.
APOS_01.M	x	APOSS Software	Anzeige der aktuellen Positionswerte während eines Positioniervorgangs.
AXEND_01.M	x	APOSS Software	Anzeige des Achsstatus bei verschiedenen Bewegungszuständen.
CANIN.M	x	APOSS Software	Eine Interrupt Prozedur wird definiert die aufgerufen wird, wenn eine CAN-Message eintrifft
CANOUT_01.m	x	APOSS Software	Sendeobjekt definieren
CAN_sample.m	x	APOSS Software	Kommunikation zwischen 2 Steuerungen
CAM Box	x	Projektierungs-handbuch Seite 47	Nockenschaltwerk: Nach dem Bedrucken eines Kartons soll der frische Druck sofort im Luftstrom getrocknet werden.
CHR_01.M	x	APOSS Software	Bildschirmsteuerzeichen (ASCII) ausgeben.
CMODE_01.M	x	APOSS Software	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	APOSS Software	Anzeige der intern errechneten Sollpositionen während einer Bewegung im Drehzahl- oder Positioniermodus.
DELAY_01.M	x	APOSS Software	Demonstration einer Wartezeit und des Einflusses von eventuell auftretenden Interrupts. Definition der Interrupt-Quellen und entsprechenden Unterprogramme (Interrupthandler).
DIM_01.M	x	APOSS Software	Aufzeichnung einer Geschwindigkeitskurve und anschließende Ausgabe als horizontales Balkendiagramm.
DORIG_01.M	x	APOSS Software	Definieren des Realnullpunktes an verschiedenen Positionen.
Drehg-S.m	x	APOSS Software	Drehgeber-Testprogramm.
ENCODERGEAR.m		APOSS Software	Encoder-Signalumsetzung
ERROR_01.M	x	APOSS Software	Auswertung eines Fehlerzustands mittels der Fehlernummer.
EXIT_01.M	x	APOSS Software	Ratespiel: Eine Ziffer muss erraten werden.
Feed-forward Berechnung	x	MCO 305 Produkthandbuch	Feedforward Berechnung.
GETP_01.M	x	APOSS Software	Ausgabe der aktuellen Filterparameter.

Dateiname oder Thema	Sie finden das Beispiel:		Beschreibung
	Online Hilfe	Handbuch oder Software	
GOSUB_01.M	x	APOSS Software	Zeit- und Weginformationen während einer Bewegung anzeigen.
GOTO_01.M	x	APOSS Software	Innerhalb einer Endlosschleife wird alle 5 Sekunden die aktuelle Zeit seit dem Programmstart ausgegeben.
HOME_01.M	x	APOSS Software	Anfahren der HOME-Position (Referenzschalter + INDEX).
IF_01.M	x	APOSS Software	Auswertung eines Fehlerzustands anhand der Fehlernummer.
IN_01.M	x	APOSS Software	Anzeige der Signalpegel an den digitalen Eingängen.
INB_01.M	x	APOSS Software	Anzeige der Signalpegel an den digitalen Eingängen.
INB_02.M	x	APOSS Software	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	APOSS Software	Zeigt verschiedene Systeminformationen an.
INCIN01.M	x	APOSS Software	Include-Datei: Zustand der Eingänge grafisch anzeigen.
INCPOS01.M	x	APOSS Software	Include-Datei: Anzeige der aktuellen, sowie der Sollposition.
INCSTA01.M	x	APOSS Software	Include-Datei: Unterprogramm zum Anzeigen des Systemstatus in Klartext.
INDEX_01.M	x	APOSS Software	Anfahren der Indexposition des Drehgebers.
INKEY_01.M	x	Siehe EXIT_01.M	
JerkMinTest.m		APOSS Software	Testprogramm zur Aufzeichnung von Bewegungsvorgängen mit den verschiedenen Rampentypen. Siehe *Datei → Beispiel.*
LOOP_01.M	x	APOSS Software	Darstellung zufälliger horizontaler Balken.
Marker.m	x	APOSS Software	Kurvenscheibensteuerung (CAM): Kartons bedrucken mit Markerkorrektur.
Marker count	x	APOSS Software	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 48	Relative Positionierung mit mechanischer Bremse.
MOTOR_01.M	x	APOSS Software	Manuelle, überwachte Bewegung des Antriebs bei abgeschalteter Lageregelung.
Fahrtst.m	x	APOSS Software	Fahrtestprogramm.
MSTOP_01.M	x	APOSS Software	Positioniervorgang unterbrechen und nach Warten wieder fortsetzen.
NOWAI_01.M	x	APOSS Software	Ausgabe von Informationen während eines Positioniervorgangs.
ONINT_01.M	x	APOSS Software	Abarbeitung von Interrupt-Signalen während Positioniervorgängen.
ORIG_01.M	x	APOSS Software	Anfahren des Realnullpunktes nach einer undefinierten Bewegung im Drehzahlmodus.

Dateiname oder Thema	Sie finden das Beispiel:		Beschreibung
	Online Hilfe	Handbuch oder Software	
OUT_01.M	x	APOSS Software	Setzen verschiedener Ausgänge in Abhängigkeit von der aktuellen Position.
OUTB_01.M	x	APOSS Software	Zustand der digitalen Eingänge auf digitale. Ausgänge übertragen.
POS_01.M	x	APOSS Software	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	APOSS Software	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	APOSS Software	Lesen und Auswerten der Statusinformation.
syncc_msim.m	x	APOSS Software	Master-Simulation per Software.
TORIG_01.M	x	APOSS Software	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	APOSS Software	Variation der Geschwindigkeit während eines Positioniervorgangs.
WAIT_01.M	x	APOSS Software	Vorführung verschiedener Wartevorgänge: Zeitverzögerung, Warten auf Positionierung, Warten auf Eingang.
WHILE_01.M	x	APOSS Software	Auf einen High-Pegel am Eingang 4 warten und anschließend eventuell vorhandene Eingaben ausgeben.

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* auf die Werkseinstellungen zurückgesetzt werden. Das Löschen des gesamten Speichers im Menü *Steuerung* → *Speicher* → *EPROM 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
- CAN-Bus

□ 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 3-17	-	[0] Keine Funktion

∗ Werkseinstellung	[ ] bei Kommunikation über serielle Schnittstelle benutzter Wert

Par. Nummer	Neue Auswahlmöglichkeiten	Neue Standardwerte
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.

□ 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 für die einfache Auswahl in Gruppen organisiert.

32-**	**MCO Grundeinstellungen**	
32-0*	Drehgeber 2 - Slave	Seite 204
32-3*	Drehgeber 1 - Master	Seite 206
32-5*	Rückführungsquelle	Seite 213
32-6*	PID-Regelung	Seite 214
32-8*	Geschwindigkeit & Beschleunigung	Seite 217

33-**	**MCO weitere Einstellungen**	
33-0*	Homefahrt	Seite 220
33-1*	Synchronisation	Seite 221
33-4*	Grenzwertbehandlung	Seite 233
33-5*	I/O Konfiguration	Seite 235
33-9*	MCO Port Einstellungen	Seite 241

34-**	**MCO Datenanzeigen**	Seite 243
34-0*	PCD Schreib-Parameter	Seite 243
34-2*	PCD Lese-Parameter	Seite 243
34-4*	Eingänge & Ausgänge	Seite 243
34-5*	Prozessdaten	Seite 243
34-7*	Diagnoseanzeigen	Seite 244

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	Seite 204
32-3*	Drehgeber 1 – Master	Seite 206
32-5*	Rückführungsquelle	Seite 213
32-6*	PID-Regelung	Seite 214
32-8*	Geschwindigkeit & Beschleunigung	Seite 217

□ 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]
	CAN Drehgeber	[3]

Funktion

Legt den Typ des Inkrementalgebers fest, der mit der Drehgeber 2 Schnittstelle verbunden ist (X55 und X62, wenn ein CAN Drehgeber verwendet wird).

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.

Wählen Sie *CAN Drehgeber* [3], wenn ein MCO CAN Drehgeber verwendet wird.

ENCODERTYPE == 9 wird ab MCO 5.00 unterstützt. Diese Slave-Simulation kann zum Testen benutzt werden. Dieser Parameter kann nicht im Menü ausgewählt werden, sondern wird im APOSS-Programm gesetzt.

Mittels zweistelliger Parameter kann man den Encoder zuordnen, wenn man zwei Motoren abwechselnd mit einer Steuerung betreiben will. (statt des früheren Befehls SWAPMENC).

SET ENCODERTYPE 1n oder
SET ENCODERTYPE 2n

ACHTUNG!:
Vor einem Befehl den Encoder zu wechseln muss immer ein MOTOR OFF ausgeführt werden, um zu vermeiden, dass ein Schleppfehler entsteht. Es müssen auch die Parameter der Steuerung oder die Achsenparameter geändert werden, falls beide Motoren unterschiedlich sind.

Die Motorzuführung kann via Relais getauscht werden.

ACHTUNG!:
Bei diesem Wechsel gehen keine Positionen verloren, auch dann nicht, wenn die Motoren manuell bewegt werden, während der andere Motor gesteuert wird. Es ist jederzeit möglich, auf den nicht gesteuerten Motor ebenso mit MAPOS zuzugreifen.

Diese Parametereinstellung kann nicht im Menü ausgewählt werden, sondern wird im APOSS-Programm gesetzt.

Beispiel

```
OUT 1 1              // Motorzuführung wechseln
SET KPROP ...        // Achsparameter ändern
SET ENCOERTYPE 11    // nun ist Encoder 1 aktiv
MOTOR ON  // Steuerung erneut einschalten
POSA 10000
   // den Motor benutzen (fahren),
   // der mit dem Master-Encoder verbunden ist
MOTOR OFF
OUT 1 0              // Motorzuführung wechseln
SET KPROP ...        // Achsparameter ändern
SET ENCOERTYPE 21    // nun ist Encoder 2 aktiv
MOTOR ON  // Steuerung erneut einschalten
POSA 0
   // wieder den Motor benutzen (fahren),
   // der mit dem Slave-Encoder verbunden 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. Die Drehgeberauflösung finden Sie auf dem Typenschild oder im Datenblatt des Drehgebers.

- 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.
- CAN Drehgeber (32-00 = [3]):
  Inkrementalgeber: Pulse pro Umdrehung
  Absolutgeber: (Pulse/Umdrehung)/4

ACHTUNG!:
Es wird immer der Parameter für die inkrementelle Auflösung (32-01 oder 32-31) verwendet, auch wenn der CAN Drehgeber ein Absolutgeber ist. Allerdings muss für einen CAN Absolutgeber ein Viertel der Encoder-Auflösung eingestellt werden. Das liegt daran, dass intern immer mit dem Vierfachen der Anzahl Striche gerechnet wird, da ein inkrementeller Geber 4 Mal mehr Quadcounts liefert als er Striche hat. Ein Absolutgeber liefert aber dieser „nur" die tatsächliche Auflösung als Maximalwert.

Wenn *Motor Steuerung* [3] in Par. 32-50 *Rückführung Slave* ausgewählt ist, kann die Auflösung mit diesem Parameter gesetzt werden.

ACHTUNG!:
Wenn *Motor Steuerung* (SPI Encoder) benutzt wird, muss die Auflösung eine zweite Potenz betragen, andernfalls würden Rundungsfehler zu Positionsabweichungen führen (z.B. SYNCP).

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]
	Hiperface	[1]
	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 *Hiperface* [1], wenn solch ein Typ eines Absolutgebers angeschlossen ist. Diese Auswahl beinhaltet die Defaulteinstellungen Encoder-ID „1" und Encoderparity „even".

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: DEFORIGIN, HOME, INDEX, IPOS und WAITNDX.

ANMERKUNG: Die Auswahl [1] *Hiperface* ist ab Firmware Version 6.7.40 verfügbar.

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.

32-04 Absolutgeber Baudrate X55

ENCODERBAUD

Option

	600	[0]
	1200	[1]
	2400	[2]
	4800	[3]
*	9600	[4]
	19200	[5]
	38400	[6]

Funktion

Wählen Sie die Baudrate des angeschlossenen Encoders.

ANMERKUNG: Dieser Parameter ist ab Firmware Version 6.7.40 verfügbar.

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 Kanäle	[1]
	2 Kanäle	[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.

Ab MCO 5.00 ist der Faktor nicht mehr begrenzt auf kleine Werte, siehe auch Benutzereinheiten in Kapitel „Projektierungshandbuch lesen".

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.

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

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.

Ab MCO 5.00 ist der Faktor nicht mehr begrenzt auf kleine Werte, siehe auch Benutzereinheiten in Kapitel „Projektierungshandbuch lesen".

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

32-13 Enc.2 Steuerung

ENCCONTROL

Option

*	Kein Soft-Wechsel	[0]
	Encoder Soft-Wechsel aktiviert	[1]
	Soft Nullstellung aktiviert	[2]
	Encoder Soft-Wechsel und Nullstellung aktiviert	[3]

Funktion

Konfiguration, wie die Position nach einem Wechsel der Drehgeberquelle ausgewertet werden soll.

Ein „Soft Encoder Wechsel" wird benutzt, wenn die Encoder während des Betriebs gewechselt werden sollen. Würde dies ohne diesen Parameter durchgeführt, dann würde die Einstellung des neuen Encoders typischerweise einen Schleppfehler verursachen, weil die Encoderwerte nicht die gleichen sind.

Wählen Sie *Kein Soft-Wechsel* [0], um direkt zu den Positionsdaten der neuen Quelle zu wechseln.

Wählen Sie *Encoder Soft-Wechsel* [1], um nicht vollständig zum neuen Wert des neuen Encoders zu wechseln. Statt dessen wird der alte Wert behalten und die Differenz zum neuen Encoder addiert. Dadurch ist es möglich, die Encoder im laufenden Betrieb zu wechseln.

Wählen Sie *Soft Nullstellung* [2], wenn der Encoder von einer anderen Achse oder für andere Zwecke benutzt wird und es nicht gewünscht ist, die Encoderwerte bei einem DEFORIGIN tatsächlich zu wechseln. Wenn *Soft Nullstellung* aktiv ist, dann kann ein DEFORIGIN benutzt werden und trotzdem ist die neu gemeldete aktuelle Position nachher Null. Der Encoder hat jedoch noch seinen alten Wert (d.h. der Wert, den Sie sehen, wenn Sie ihn mit SYSVAR 0x0122022A oder ähnlich auslesen).

Die Auswahl *Encoder Soft-Wechsel und Nullstellung* [3] ermöglicht einen ruckfreien Wechsel des Feedback-Encoders während des Betriebs und das Setzen der Position auf Null ohne die aktuelle Position zu verlieren.

32-14 Enc.2 Teilnehmer ID

Encoder Teilnehmer ID

Bereich

1 – 127	* 127

Funktion

Geben Sie die Teilnehmer ID des Feedback-CAN-Encoders (Rückführung) ein.

32-15 Enc.2 CAN Überwachung

Encoder CAN Überwachung

Option

*	Aus	[0]
	Ein	[1]

Funktion

Die Überwachung des Feedback-CAN-Encoders kann ein- und ausgeschaltet werden.

Aus [0] ist die Default-Einstellung. Wählen Sie [1] wenn der Feedback CAN Encoder überwacht werden soll.

□ 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]
	CAN Drehgeber	[3]

Funktion

Legt den Typ des Inkrementalgebers fest, der mit der Drehgeber 1 Schnittstelle verbunden ist (X56 und X62, wenn ein CAN Drehgeber benutzt wird).

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.

Wählen Sie *CAN Drehgeber* [3], wenn ein Drehgeber mit CAN Schnittstelle angeschlossen ist.

Mit 2-stelligen Parameternummern kann der Encoder zugeordnet werden (statt des früheren Befehls SWAPMENC):

SET MENCODERTYPE 11 oder
SET MENCODERTYPE 21

Dieser Parameter kann nicht im Menü ausgewählt werden, sondern wird im APOSS Programm gesetzt. Siehe Beispiel in Par. 32-00.

Behandlung der automatisch erzeugten CAN-Objekte

Für die Informationsübertragung der Istposition über den Bus werden die benötigten CAN-Objekte (PDOs, GUARD-, SYNC-Objekt) automatisch angelegt und der Teilnehmer initialisiert (NMT0). Diese im Hintergrund ausgeführte Objekt-Erzeugung entspricht im wesentlichen dem CANINI-Befehl, bleiben die mit MENCODERTYPE automatisch erzeugten Objekte jedoch auch bei einem erneuten applikationsseitigen CANINI unverändert erhalten.

Ebenfalls werden die mit MENCODERTYPE automatisch erzeugten Objekte auch durch ein CANDEL –1 nicht gestoppt. Die automatisch erzeugten Objekte können innerhalb der Applikation nur durch einen erneuten, abweichenden SET MENCODERTYPE Befehl gelöscht bzw. umkonfiguriert werden.

Auch beim Programmabbruch (mit [Esc]) bleiben die Objekte noch bis zum nächsten Programmstart erhalten. Erst beim Neustart eines Programms werden die Objekte gelöscht und entsprechend der Einstellung im permanenten Parametersatz oder entsprechend der Definition im Programm neu angelegt.

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. Die Drehgeberauflösung finden Sie auf dem Typenschild oder im Datenblatt des Drehgebers.

– Digitaler Inkrementalgeber (32-30 = [1]): Pulse pro Umdrehung
– CAN Drehgeber (32-00 = [3]): Inkrementalgeber: Pulse pro Umdrehung Absolutgeber: (Pulse pro Umdrehung)/4

ACHTUNG!:
Es wird immer der Parameter für die inkrementelle Auflösung verwendet, auch wenn der CAN Drehgeber ein Absolutgeber ist. In diesem Fall muss aber ein Viertel der Encoder-Auflösung eingestellt werden

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]
	Hiperface	[1]
	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 *Hiperface* [1], wenn solch ein Typ eines Absolutgebers angeschlossen ist. Diese Auswahl beinhaltet die Defaulteinstellungen Encoder-ID „1" und Encoderparity „even".

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.

ANMERKUNG: Die Auswahl [1] *Hiperface* ist ab Firmware 6.7.40 verfügbar.

32-33 Absolutgeber Auflösung

MENCODERABSRES

Bereich [Unit]

1 – MLONG [PPR]	∗ 8192

Funktion

ANMERKUNG: Der Parameter wird nur angezeigt wenn Par. 32-32 ≠ 0.

32-34 Absolutgeber Baudrate X55

MENCODERBAUD

Option

	600	[0]
	1200	[1]
	2400	[2]
	4800	[3]
∗	9600	[4]
	19200	[5]
	38400	[6]

Funktion

Wählen Sie die Baudrate des angeschlossenen Encoders.

ANMERKUNG: Dieser Parameter ist ab Firmware Version 6.7.40 verfügbar.

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 Kanäle	[1]
	2 Kanäle	[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.

32-43 Enc.1 Steuerung

MENCCONTROL

Option

∗	Kein Soft-Wechsel	[0]
	Encoder Soft-Wechsel aktiviert	[1]
	Soft Nullstellung aktiviert	[2]
	Encoder Soft-Wechsel und Nullstellung aktiviert	[3]

Funktion

Konfiguration, wie die Master-Position nach einem Wechsel der Drehgeberquelle ausgewertet werden soll.

Ein „Soft Encoder Wechsel" wird benutzt, wenn die Encoder während des Betriebs gewechselt werden sollen. Würde dies ohne diesen Parameter durchgeführt, dann würde die Einstellung des neuen Encoders typischerweise einen Schleppfehler verursachen, weil die Encoderwerte nicht die gleichen sind.

Wählen Sie *Kein Soft-Wechsel* [0], um direkt zu den Positionsdaten der neuen Master-Quelle zu wechseln.

Wählen Sie *Encoder Soft-Wechsel* [1], um nicht vollständig zum neuen Wert des neuen Encoders zu wechseln. Stattdessen wird der alte Wert behalten und die Differenz zum neuen Encoder addiert. Dadurch ist es möglich, die Encoder im laufenden Betrieb zu wechseln.

Wählen Sie *Soft Nullstellung* [2], wenn der Encoder von einer anderen Achse oder für andere Zwecke benutzt wird und es nicht gewünscht ist, die Encoderwerte bei einem DEFMORIGIN tatsächlich zu wechseln. Wenn *Soft Nullstellung* aktiv ist, dann kann ein DEFMORIGIN benutzt werden und trotzdem ist die neu gemeldete aktuelle Position nachher Null. Der Encoder hat jedoch noch seinen alten Wert (d.h. der Wert, den Sie sehen, wenn Sie ihn mit SYSVAR 0x0122022A oder ähnlich auslesen).

Die Auswahl *Encoder Soft-Wechsel und Nullstellung* [3] ermöglicht einen ruckfreien Wechsel des Master-Encoders während des Betriebs und das Setzen der Position auf Null ohne die aktuelle Position zu verlieren.

32-44 Enc.1 Teilnehmer ID

Encoder 1 Teilnehmer ID

Bereich

1 – 127	∗ 127

Funktion

Geben Sie die Teilnehmer ID des Master-CAN-Encoders ein.

32-45 Enc.1 CAN Überwachung

Encoder 1 CAN Überwachung

Option

∗	Aus	[0]
	Ein	[1]

Funktion

Die Überwachung des Master-CAN-Encoders kann ein- und ausgeschaltet werden.

Aus [0] ist die Default-Einstellung. Wählen Sie [1] wenn der Master-CAN-Encoder überwacht werden soll.

□ 32-5* Rückführungsquelle

32-50 Rückführung Slave

Option

	Encoder 1 X56	[1]
∗	Encoder 2 X55	[2]
	Motor Steuerung	[3]

Funktion

Wählen Sie die Rückführungsquelle für MCO:

Wählen Sie [1] für Encoder 1 an Klemme X56.

Wählen Sie [2] für Encoder 2. 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.

Mit Par. 32-01 *Inkrementalgeber Auflösung* kann die Auflösung für die *Motor Steuerung* bestimmt werden.

ANMERKUNG: Parameter [1] ist ab Firmware-Version 6.7.40 verfügbar.

32-52 Rückführung Master

Option

∗	Encoder 1 X56	[1]
	Encoder 2 X55	[2]
	Motor Steuerung	[3]

Funktion

Wählen Sie die Rückführungsquelle für MCO:

Wählen Sie [1] für Encoder 1 an Klemme X56.

Wählen Sie [2] für Encoder 2. 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 ein MCO 305 Feedback von der Rückführungsquelle wie in Par. 102 festgelegt. Dies kann ein interner 24 V Encoder, eine Encoder Option oder eine Drehgeber Option sein.

ANMERKUNG: Der Parameter ist ab Firmware-Version 6.7.40 verfügbar.

□ 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-Feedforward* 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-Feedforward 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 einer Testfahrt mit dem Oszilloskop.

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 [Einheit]

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)

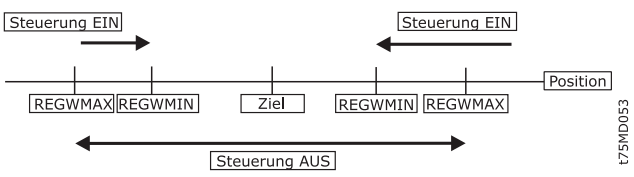
REGWMAX

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.

32-73 Integralgrenzwert Filterzeit

KILIMTIME

Range [ms]

-10000 … 10000	* 0

Option

*	Integralanteil der PID-Positionsregelung ist permanent aktiv.	[0]
	< 0 = Integralanteil ist nur bei Motorstillstand aktiv	[1]
	> 0 = Integralanteil ist nur während der Motorbewegung aktiv	[2]

Funktion

Zeitspanne, über welche der Integralgrenzwert des Positionsreglers auf den definierten KILIM-Wert erhöht bzw. auf Null reduziert wird.

Der Integralanteil der PID Positionsregelung kann permanent aktiv sein oder nur während einer Fahrt oder nur im Stillstand. Der Wert des Parameters KILIMTIME legt das gewünschte Verhalten fest.

Die Werkseinstellung 0 für KILIMTIME bedeutet, dass der Integralanteil der PID Positionsregelung permanent aktiv ist, entsprechend den Einstellungen der Parameter 32-62 KINT und 32-63 KILIM.

Ein Wert größer als 0 für KILIM bedeutet, dass der Integralanteil der PID Positionsregelung nur während der Motorbewegung aktiv ist. Beim Motorstillstand wird der Integralanteil auf Null reduziert. Beim Start der Bewegung wird das zulässige Integralmaximum innerhalb der durch KILIMTIME definierten Zeitspanne von 0 auf den definierten Grenzwert KILIM angehoben. Sobald der Motor wieder stoppt, wird das Integralmaximum innerhalb der Zeitspanne KILIMTIME auf Null reduziert und der Integralanteil ist somit abgeschaltet.

Ein solches Verhalten des Integralanteils erlaubt bei Synchronisationsanwendungen während der Bewegung einen sehr kleinen Synchronisationsfehler, bei gleichzeitig hoher Elastizität in der Ruhelage. Insbesondere bei Förderketten kann dies gewünscht sein.

Ein Wert von größer als 0 für KILIM bedeutet, dass der Integralanteil der PID Positionsregelung nur im Stillstand aktiviert ist. Der Absolutwert von KILIMTIME definiert hierbei die Zeitspanne, in der beim Motorstart das zulässige Integralmaximum auf Null reduziert wird bzw. in der beim erneuten Stopp das zulässige Integralmaximum auf den konfigurierten Grenzwert KILIM angehoben wird. Dieses Verhalten des Integralanteils ist insbesondere hilfreich um ein „nervöses" Motorverhalten oder Aufschaukeln während der Bewegung zu verhindern und gleichzeitig eine sehr exakte Endpositionierung zu ermöglichen.

Siehe auch Par. 32-60 KPROP und 32-61 KDER.

32-74 Schleppfehler Filterzeit

POSERRTIME

Bereich [ms]

0 … 10000	∗ 0

0 = es ist kein Zeitfenster aktiviert, d.h. der Fehlerstatus wird sofort ausgelöst.

Funktion

Zeitfenster [ms] für das Auslösen eines Positionsfehlerstatus.

Zu große Schleppfehler (POSERR) lösen den Fehlerstatus aus, wenn sie länger existieren als POSERRTIME.

Der Default dieses Parameters ist 0. Wenn dieser Parameter nicht 0 ist, wird ein Schleppfehler nur dann erzeugt, wenn der Wert des Schleppfehlers (Par. 32-67) für eine Zeit länger als POSERRTIME erreicht wird. Intern wird der Schleppfehler alle 20 ms geprüft. Das heißt, das Erkennen eines Schleppfehlers kann im schlimmsten Fall bis zu NoOfAxes * 20 ms dauern. Wenn zum Beispiel POSERRTIM auf 300 ms gesetzt ist, wird der Schleppfehler mehr als (300 / 1 * 20) = 15 Mal geprüft.

Wenn das erfüllt ist, wird ein Fehler ausgegeben.

□ 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

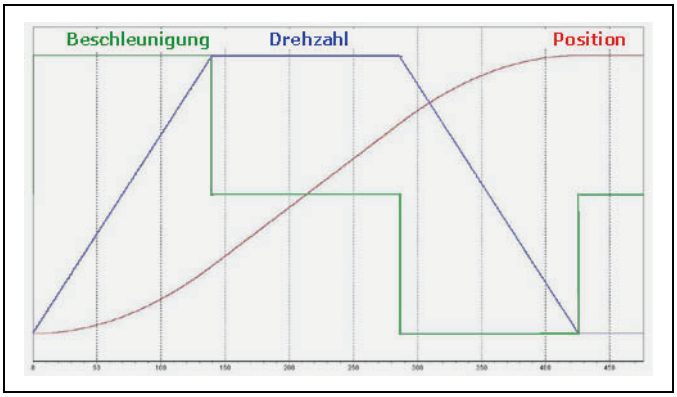
RAMPTYPE

Option

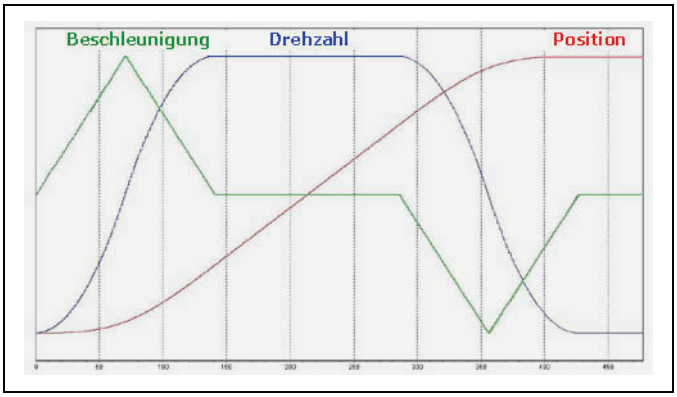
∗	Linear	[0]
	S-Rampe	[1]
	Bewegungsprofil mit Ruckbegrenzung	[2]

Funktion

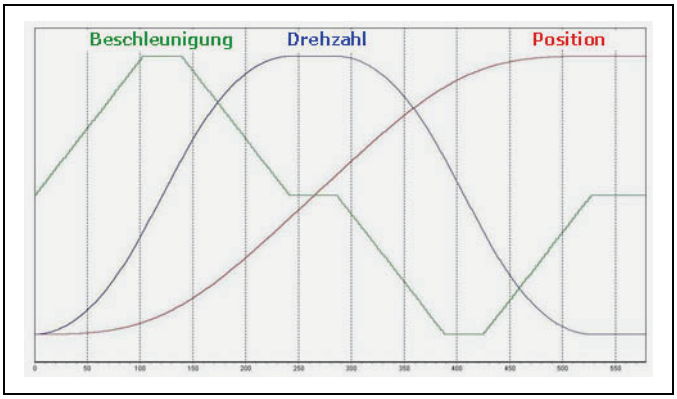
RAMPTYPE bestimmt die Rampenform: Trapez, sinusförmig oder mit Ruckbegrenzung. Diese Rampentypen sind für alle Fahrbewegungen (POSA, POSR, CVEL und MOTOR STOP) relevant, nicht aber bei SYNCx-Befehlen.



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 Par. 32-81 *Kürzeste Rampe* 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 Parameter *Ruckdauer* JERKMIN berechnet.

Es gibt vier verschiedene Parameter für RAMPTYPE = 2 mit Ruckbegrenzung, siehe „Ruckbegrenzung" im Kapitel „Funktionen und Beispiele".

Siehe auch *Datei* → *Beispiel* JerkMinTest.m.

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*.

32-86 Beschl.rampe für Ruckbegrenzung

JERKMIN

Range

0 … MLONG [ms]	∗ 100

Funktion

Die Beschleunigungsrampe-Konstante für ruckbegrenzte Fahrbewegungen definiert die Zeitspanne [ms], die zum Erreichen der Beschleunigung von 0 bis zur Maximalbeschleunigung notwendig ist.

Es gibt vier verschiedene JERKMIN Optionen, siehe „Ruckbegrenzung" im Kapitel „Funktionen und Beispiele".

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.

ACHTUNG!:
Eine geänderte JERKMIN Einstellung wird nach dem nächsten Fahrbefehl (POSA, POSR, CVEL oder MOTOR STOP) ausgeführt und bei der Berechnung der Beschleunigung 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.

NB!:
Ab MCO 5.00 steht der Parameter JERKMIN im Parameterfeld „Grundeinstellungen" zur Verfügung.

Beispiel

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

VELMAX = 3000 * 500 * $\frac{4}{60}$ = 100,000 qc/s
= 100 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

32-87 Beschl.dauer für Ruckbegrenzung

JERKMIN2

Bereich

0 … MLONG [ms] ∗ 0

Funktion

Die Beschleunigungsdauer für die Ruckbegrenzung definiert die Zeitspanne [ms], die von der Maximalbeschleunigung bis zum Erreichen der Beschleunigung 0, d.h. z.B. bis zum Erreichen der konstanten Maximalgeschwindigkeit notwendig ist.

ACHTUNG!:
Eine geänderte JERKMIN2 Einstellung wird nach dem nächsten Fahrbefehl (POSA, POSR, CVEL oder MOTOR STOP) ausgeführt und bei der Berechnung der Beschleunigung 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 JERKMIN2 auf „0" gesetzt ist, wird der gleiche Wert wie in Par. 32-86 benutzt.

Es gibt vier verschiedene JERKMIN Optionen, siehe „Ruckbegrenzung" im Kapitel „Funktionen und Beispiele" und Par. 32-86.

32-88 Bremsrampe für Ruckbegrenzung

JERKMIN3

Range

0 … MLONG [ms] ∗ 0

Funktion

Die Konstante Bremsrampe definiert die Zeitspanne [ms], die zum Erreichen der definierten Maximalverzögerung benötigt wird.

ACHTUNG!:
Eine geänderte JERKMIN3 Einstellung wird nach dem nächsten Fahrbefehl (POSA, POSR, CVEL oder MOTOR STOP) ausgeführt und in der Berechnung der Beschleunigung 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 JERKMIN3 auf „0" gesetzt ist, wird der gleiche Wert wie in Par. 32-86 benutzt.

Es gibt vier verschiedene JERKMIN Optionen, siehe „Ruckbegrenzung" im Kapitel „Funktionen und Beispiele" und Par. 32-86.

32-89 Bremsdauer für Ruckbegrenzung

JERKMIN4

Range

0 … MLONG [ms] ∗ 0

Funktion

Die Bremsdauer ist die Zeitspanne [ms], die benötigt wird, um von der Maximalverzögerung die Verzögerung „0" zu erreichen, also bis zum Stillstand.

ACHTUNG!:
Eine geänderte JERKMIN4 Einstellung wird nach dem nächsten Fahrbefehl (POSA, POSR, CVEL oder MOTOR STOP) ausgeführt und in der Berechnung der Beschleunigung 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 JERKMIN3 auf „0" gesetzt ist, wird der gleiche Wert wie in Par. 32-86 benutzt.

Es gibt vier verschiedene JERKMIN Optionen, siehe „Ruckbegrenzung" im Kapitel „Funktionen und Beispiele" und Par. 32-86.

□ MCO weitere Einstellungen

33-0*	Homefahrt	Seite 220
33-1*	Synchronisation	Seite 221
33-4*	Grenzwertbehandlung	Seite 231
33-5*	I/O Konfiguration	Seite 235
33-8*	Globale Parameter	Seite 239
33-9*	MCO Port Einstellungen	Seite 241

□ 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:

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.

$$\text{P. 33 - 03 Homefahrt - Geschwindigkeit [U/Min]} = \text{Homefahrt - Geschwindigkeit} * \frac{\text{P. 32 - 80 Max.Geschwindigkeit}}{\text{P. 32 - 83 Geschwindigkeitsteiler}}$$

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/Positions-synchronisation, 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.

$$\text{Slave Geschwindigkeit} = \text{Master Geschwindigkeit} * \frac{\text{Par. 33 - 11 Syncfaktor Slave}}{\text{Par. 33 - 10 Syncfaktor Master}}$$

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.

Ab MCO 5.00 ist der Faktor nicht mehr auf kleine Werte begrenzt, siehe auch Benutzereinheiten in Kapitel „Projektierungshandbuch lesen".

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.

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

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

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*.

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

Ab MCO 5.00 ist der Faktor nicht mehr auf kleine Werte begrenzt, siehe auch Benutzereinheiten in Kapitel „Projektierungshandbuch lesen".

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.

Ab MCO 5.00 ist der Faktor nicht mehr auf kleine Werte begrenzt, siehe auch Benutzereinheiten in Kapitel „Projektierungshandbuch lesen".

33-13 Genauigkeitsfenster für Positionssync.

SYNCACCURACY

Bereich [Unit]

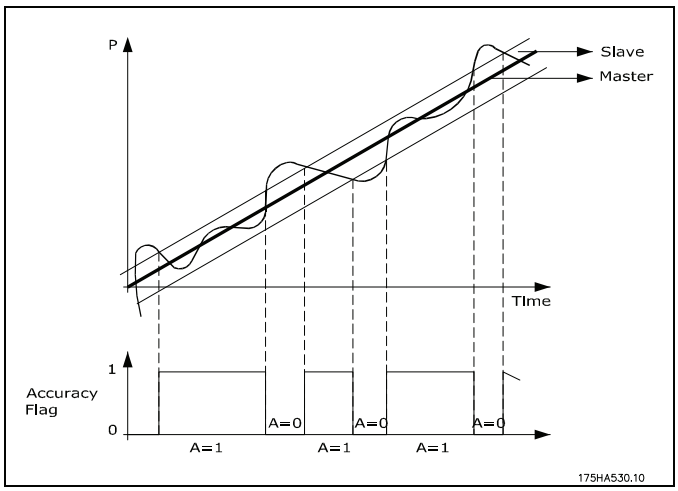
-MLONG – MLONG [qc]	* 1000
0 – MLONG =	Ein positives Vorzeichen liefert den absoluten Wert an SYNCERR.
–MLONG to –1 =	Ein negatives Vorzeichen liefert den Synchronisationsfehler an SYNCERR mit Vorzeichen. Daraus lässt sich dann erkennen, ob die Synchronisation voraus- oder nachläuft.

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.

ACHTUNG!:
Die neue Größe des Sync-Fensters wird erst nach einem neuen SYNC-Befehl aktiviert. Dies betrifft nicht nur die Größenänderung sondern auch die Aktivierung selbst.

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 [qc] or in CAM mode [BE]	∗ 4096

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.

Ein erweiterter Markertyp realisiert ein logisches UND für den Markereingang und einen zweiten Eingang, z.B. um jeden zweiten Marker mit einer SPS zu synchronisieren. Dies beeinflusst den Befehl MIPOS. MIPOS liefert nur dann einen neuen Wert, wenn die Bedingung erfüllt ist.

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

SYNCMTYPS unterstützt die folgenden Werte, wenn ENCODERTYPE == 9

0– virtuelle Marker werden mit einem Abstand von ENCODERFREQ erzeugt.

n2– Bei positiver Flanke von Eingang n wird die aktuelle Position als Markerposition genommen.

n3– Bei negativer Flanke von Eingang n wird die aktuelle Position als Markerposition genommen.

Die Genauigkeit ist natürlich auf 1 ms begrenzt, weil dies die interne Update-Rate ist.

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]
	SYNCMSTART 7	[7]
	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]
	SYNCMSTART 1007	[1007]
	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.

7 = Start mit einem Poly5 um den Master exakt auf einer Markerposition zu erreichen.

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.

1007 = wie [7], 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).

Parameter 7 and 1007 werden ab MCO 5.00 unterstützt: In der Online-Hilfe ist genau beschrieben, wie diese Parameter funktionieren.

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:

$$\text{Schrittweite} = \frac{\text{P.33 - 17 Markerabstand Master}}{\text{P.33 - 27 Offset Filterzeit (Int. Anteil)}}$$

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*).

Die Offset-Filterzeit definiert die Zeit die benutzt werden soll, um einen Markerabstand auszugleichen.

Ab MCO 5.00 wird nicht nur der Offset über die gegebene Zeit verteilt, sondern es wird auch eine trapezförmige Bewegung berechnet und zur normalen Synchronisationsfahrt addiert.

Damit werden begrenzte Beschleunigungen und Geschwindigkeiten berechnet, mit der Annahme dass normalerweise 20 % für Beschleunigung, 60 % mit konstanter Geschwindigkeit und weitere 20 % der Zeit zum Bremsen benutzt werden. Basierend

auf dieser Auslegung wird berechnet welche Beschleunigungs- und Verzögerungsrampen benutzt werden sollen und wie die maximale Geschwindigkeit begrenzt werden muss, um dies zu erreichen.

Diese neue Verteilung wird nicht nur für Offsets benutzt, sondern auch für Startkorrekturen (erste Markerkorrektur), normale Markerkorrekturen und kumulierte Fehler, die durch eine REVERS Einstellung verursacht werden.

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.

Wenn Par. 33-29 SYNCMFTIME gesetzt ist wird G_MMarkerDist ein wenig anders berechnet. Wenn die gesetzte Zeit zu einer Anzahl von Markern führt, die kleiner ist als 100 ist, werden mindestens 100 genommen um den durchschnittliche Markerabstand (Filter) zu berechnen. Dies betrifft nur die Berechnung des Markerabstands. Für die anderen Filterberechnungen (Abweichung) wird weiterhin die Anzahl der Marker verwendet, die durch Par. 33-29 SYNCMFTIME und aktueller Master-Geschwindigkeit berechnet wurden.

Siehe auch Abbildung Marker Correction im Kapitel „Technische Referenz".

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 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 die Markerfilter-Konstante als Zeitkonstante zum Filtern benutzt wird. Dann sollte die benötigte Zeit für eine Ausgabe entsprechend eines konstanten Eingangswerts 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:

$$\text{Getriebekorrektur} = \frac{\text{SYNCMPULSM - gefilterten Markerabstand}}{\text{gefilterten Markerabstand}}$$

Filter Master-Geschwindigkeit und Getriebekorrektur

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 Korrekturfilterzeit entsprechend dem Par. 33-27 *Offset Filterzeit* zum Filtern gesetzt. (abhängig von der Master-Geschwindigkeit; 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 sind, 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.

Berechnung Markerabstand

Wenn SYNCMFTIME gesetzt ist, wird G_MMarkerDist etwas anders berechnet. Wenn die angesetzte Zeit zu einer Anzahl von Markern führt, die kleiner ist als 100 ist, dann werden mindestens 100 genommen um den durchschnittliche Markerabstand (Filter) zu berechnen. Dies betrifft nur die Berechnung des Markerabstands. Für die anderen Filterberechnungen (Abweichung) wird weiterhin die Anzahl der Marker verwendet, die durch SYNCMFTIME und aktueller Master-Geschwindigkeit berechnet wurden.

Siehe auch Abbildung Marker Correctioni in der „Technischen Referenz" der „MCO 305 Befehlsreferenz".

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 Befehl arbeitet mit SYNCM und SYNCC. Der Wert wird durch den Anwender mit SET SYNCMMAXCORR in qc (Slave) eingegeben.

Ein Wert > 0 begrenzt die Markerkorrektur durch den vorgegeben Wert. Sollte also die Korrektur größer werden, ist sie auf diesen Wert begrenzt.

Ein Wert < 0 setzt den Parameter so, dass überhaupt keine Korrektur gemacht wird. Der Anwender kann so die Markerkorrektur ausschalten.

Unterstützung des Wertes < 0 ab MCO 5.00.

ACHTUNG!:
Wenn Par. 33-29 *Filterzeit für Markerkorrektur* oder Par. 33-26 *Geschwindigkeitsfilter* (negativ) gesetzt isat, 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.

33-32 Geschw. Feedforward Anpassung

SYNCFFVEL

Range

0 … MLONG [per mill von VCMD] ∗ 0

0 = nicht aktiviert

Funktion

Der Parameter unterstützt geschwindigkeitsabhängiges Feedforward in Synchronmodi (SYNCP/ SYNCM/ SYNCC).

Dieser Parameter ist entweder 0 (deaktiviert = Default) oder hat den Wert, der den Feedforward in 1/1000 der Sollgeschwindigkeit angibt. Das heißt, ein Wert von 1000 addiert einen Feedforward von der VCMD (Sollgeschwindigkeit) zur MPCMD (Sollposition der Synchronisation).

Um den Wert der benutzt werden soll zu bestimmen, kann die SYSVAR NORMTRACKERR benutzt werden. Wenn Sie herausfinden, dass der normale NORMTRACKERR 100 Prozent ist, dann sollten Sie die SYNCFFVEL auf 1000 setzen.

Dadurch kann SYNCERR minimiert werden. Mit so einem Feedforward kann SYNCERR fast ausgeschlossen werden, ohne die Nachteile durch Nutzen eines Interrupt-Teils im PID.

Zurzeit gibt es noch den Nachteil eines Rucks, wenn Sie SYNCP während der Fahrt starten. Dies tritt auf, wenn die Geschwindigkeit durch ein SYNCC oder ein CVEL erreicht ist und dann SYNCP gestartet wird.

Neue SYSVAR REG_NORMTRACKERR (4124), see also SDO dictionary, axis process data.

Diese SYSVAR liefert den Fahrwegfehler im Verhältnis zur Sollgeschwindigkeit in Prozent. Zum Beispiel sagt ein Wert von 120 aus, dass der Fahrwegfehler 1,2-mal VCMD ist. Dieser Wert ist relativ konstant, wenn sich die Bedingungen nicht ändern (Last, Reibung, …) und typischerweise unabhängig von der Geschwindigkeit.

Der Parameter ist ab MCO 5.00 verfügbar.

33-33 Synchronisationsfehler-Fenster

SYNCVFLIMIT

Bereich [qc]

0 … MLONG ∗ 0

0 = nicht aktiviert

Funktion

Synchronisationsfehler-Fenster [qc] für die automatische Deaktivierung des Par. 33-26 *Geschwindigkeitsfilter* SYNCVFTIME.

Der *Geschwindigkeitsfilter* wird nicht aktiviert, wenn der Synchronisationsfehler SYNCERR den Wert erreicht, der durch SYNCVFLIMIT definiert ist. Der *Geschwindigkeitsfilter* wird wieder aktiviert, wenn SYNCERR kleiner als 1/5 des Wertes von SYNCVFLIMIT wird.

Dieser Parameter hilft große Synchronisationsfehler zu vermeiden, wenn der Master beschleunigt oder bremst und ein hoher Wert für SYNCVFTIME benutzt wird.

Dieser Parameter erlaubt das Ausschalten des Par. 33-26 *Geschwindigkeitsfilter* SYNCVFTIME, wenn der Fehler größer als SYNCVFLIMIT wird. Sobald der Filterfehler (PFG_G_MFILTERROR) den Wert von SYNCVFLIMIT [qc] erreicht, wird der Filter langsam deaktiviert. Wenn der Fehler kleiner wird als SYNCVFLIMIT/5, wird der Filter wieder aktiviert. Bei SYNCM wird der interne Filter bei jedem SYNCVFTIME auf Null zurückgesetzt, weil er nur ansteigt und nie abnimmt. In diesem Fall wird nach SYNCVFTIME wieder geprüft ob der Fehler klein genug ist, um den Filter wieder zu aktivieren.

Der Filter wird nicht sofort deaktiviert oder aktiviert, sondern er wird langsam 1 ms pro ms erhöht oder verringert. Und er wird nicht vollständig deaktiviert, sondern es werden mindestens 5 ms übrig gelassen. Bei SYNCP wird der ursprüngliche Filterwert noch für die Berechnung des aktuellen Filterfehlers (PFG_G_MFILTERROR) benutzt, um zu entscheiden, ob der Filter wieder aktiviert werden sollte. Bei SYNCM (keine Filterkorrektur) wird der Filterfehler auf Null zurückgesetzt und mindestens SYNCVFTIME gewartet, um zu entscheiden, ob der Filterfehler unter dem Limit bleibt oder nicht. Falls er unter dem Limit bleibt wird der Filter wieder aktiviert.

Der Filterfehler kann mit der SYSVAR PFG_G_MFILTERROR beobachtet werden, siehe SDO Dictionary, axis process data.

Der Parameter ist ab MCO 5.00 verfügbar.

□ 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 Verwendung des Befehls DEFORIGIN wird die Wegbegrenzung automatisch angepasst, so dass 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 Verwendung des Befehls DEFORIGIN 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 des Endschalters erkannt werden, wodurch sich der Antrieb beim Auftreten der Fehlermeldung bereits außerhalb des zulässigen Verfahrbereichs befindet.

Es ist möglich, den Software-Endschalterfehler zu löschen und dann in die entgegengesetzte Richtung zu fahren. Wenn dann aber erneut versucht wird, in die falsche Richtung zu fahren, tritt der Fehler 198 F_LIMIT_VIOLATION auf. 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.

Verbessertes Verhalten der Endschalter ab MCO 5.00.

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 des Endschalters erkannt werden, wodurch der Antrieb beim Auftreten der Fehlermeldung bereits außerhalb des zulässigen Verfahrbereichs ist.

Es ist möglich, den Software-Endschalterfehler zu löschen und dann in die entgegengesetzte Richtung zu fahren. Wenn dann aber erneut versucht wird, in die falsche Richtung zu fahren, tritt der Fehler 198 F_LIMIT_VIOLATION auf. 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.

Verbessertes Verhalten der Endschalter ab MCO 5.00.

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.

ACHTUNG!:
Modifizierte Behandlung von TESTWIN um den Gebrauch von CANopen anzupassen (ab MCO 5.00): Wenn TESTTIM gesetzt ist, aber nicht TESTVAL, wird automatisch CANopen angenommen. In diesem Fall wird geprüft, ob die Zeit innerhalb von TESTWIN ist länger als TESTTIM. Falls dem so ist, ist die Position erreicht. Andernfalls ist die Position nicht erreicht worden.

□ 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 N*C [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.

ANMERKUNG: End- und Referenzschalter erlauben den Einsatz jeden beliebigen Eingangs, es werden also auch höhere Nummern unterstützt.

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

Das Verhalten bei aktivierten Endschaltern ist ab MCO 5.00 wie folgt:

Wenn Hardware- oder Software-Endschalter aktiviert sind, kann ein Software-Endschalterfehler gelöscht und danach in die entgegengesetzte Richtung gefahren werden. Wenn danach aber erneut versucht wird, in die falsche Richtung zu fahren, wird ein neuer Fehler 198 erzeugt.

Hardware-Endschalter werden genauso behandelt wie Software-Endschalter.

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.

So kann verhindert werden, dass beim Datenaustausch mit CAN-Terminals ein gestörter CAN-Verkehr bereits zu einem Stoppen des Reglers führt. (Z.B. wenn nur reine Informationsdaten an ein Terminal übertragen werden, deren korrekte Anzeige nicht sicherheitsrelevant ist oder die ohnehin zyklisch immer wieder aktualisiert werden.)

In der Error-Routine kann dann zuerst überprüft werden, ob es sich um einen CAN-Fehler (189) handelt, der sofort wieder gelöscht werden kann. In allen anderen Fällen kann in der Error-Routine die Motorregelung bei Bedarf abgeschaltet und der laufende Prozess entsprechend den Notwendigkeiten der Applikation gestoppt 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.

□ 33-9* MCO Port Einstellungen

33-90 X62 MCO CAN Teilnehmer ID

CANNR

Range

0 … 127 N/A	∗ 127

Funktion

Definiert die *CAN-Teilnehmer ID* für den MCO-Bus auf der Optionskarte. Sie wird während der Interface-Einstellungen festgelegt.

Wenn die *CAN-Teilnehmer ID* auf 9999 gesetzt wird, werden keine Standard CAN-Objekte erzeugt. Standard-CAN-Objekte werden zur Kommunikation mit dem APOSS-Programm bei den Befehlen OUTMSG, INMSG und INGLB benötigt.

33-91 X62 MCO CAN Baudrate

CANBAUD

Option

	10 Kbps	[16]
	20 Kbps	[17]
	50 Kbps	[18]
	100 Kbps	[19]
∗	125 Kbps	[20]
	250 Kbps	[21]
	500 Kbps	[22]
	1000 Kbps	[24]

Funktion

CANBAUD definiert die Baudrate für CAN für den MCO Bus.

ie Baudrate auch mit dem APOSS Befehl SET und dem Parameter CANBAUD gesetzt werden.

Beispiel

```
SET CANBAUD 22  // Setze Baudrate auf 500 kBaud
SAVE GLBPARS    // Globale Parameter speichern
                // Anlage neu starten
```

33-94 X60 MCO RS485 Serieller Abschluss

RSTERMINATION

Option

∗	Aus	[0]
	Ein	[1]

Funktion

Wählen Sie den Abschluss für die RS485 Verbindung an Klemme X60.

33-95 X60 MCO RS485 Serielle Baudrate

RSBAUDRATE

Option

	2400 Baud	[0]
	4800 Baud	[1]
∗	9600 Baud	[2]
	19200 Baud	[3]
	38400 Baud	[4]
	57600 Baud	[5]
	76800 Baud	[6]
	115200 Baud	[7]

Funktion

Definieren Sie die Baudrate für den MCO RS485 seriellen Adapter.

□ 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 zurückgesetzt
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

□ MCO Grundeinstellungen, Parameterliste

Par. Nr. #	Parametername	Parameterbeschreibung	Werkseinstellung	Ändern während des Betriebs	4-Par.-sätze	Umwandlungsindex	Typ
32-0*	**Drehgeber 2 - Slave**						
32-00	ENCODERTYPE	Inkrementalgeber Signaltyp	[1] RS422	TRUE	2-Set-ups		Uint8
32-01	ENCODER	Inkrementalgeber Auflösung	1024 PPR	TRUE	2-Set-ups		Uint32
32-02	ENCODER ABSTYPE	Absolutgeber Protokoll	[0] Keiner	TRUE	2-Set-ups		Uint8
32-03	ENCODER ABSRES	Absolutgeber Auflösung	8192 Pulse/U	TRUE	2-Set-ups		Uint32
32-04	ENCODERBAUD	Absolutgeber Baudrate X55	[4] 9600	FALSE	2 set-ups		Uint8
32-05	ENCODER ABSTYPE	Absolutgeber Datenlänge	25 Bit	TRUE	2-Set-ups		Uint8
32-06	ENCODERFREQ	Absolutgeber Taktfrequenz	262.000 kHz	TRUE	2-Set-ups		Uint32
32-07	ENCODER CLOCK	Absolutgeber Takterzeugung	[1] On	TRUE	2-Set-ups		Uint8
32-08	ENCODER DELAY	Absolutgeber Kabellänge	0	TRUE	2-Set-ups		Uint16
32-09	ENCODER MONITORING	Drehgeber-Überwachung	[0] Aus	TRUE	2-Set-ups		Uint8
32-10	POSDRCT	Drehrichtung	[1] Keine Funktion	TRUE	2-Set-ups		Uint8
32-11	POSFACT_N	Benutzerfaktor Nenner	1	TRUE	2-Set-ups		Uint32
32-12	POSFACT_Z	Benutzerfaktor Zähler	1	TRUE	2-Set-ups		Uint32
32-13	ENCCONTROL	Enc.2 Steuerung	0	TRUE	2-Set-ups		Uint8
32-14		Enc.2 Teilnehmer ID	127	TRUE	2-Set-ups		Uint8
32-15		Enc.2 CAN Überwachung	[0] Aus	TRUE	2-Set-ups		Uint8
32-3*	**Drehgeber 1 - Master**						
32-30	MENCODER TYPE	Inkrementalgeber Signaltyp	[1] RS422	TRUE	2-Set-ups		Uint8
32-31	MENCODER	Inkrementalgeber Auflösung	1024 PPR	TRUE	2-Set-ups		Uint32
32-32	MENCODER ABSTYPE	Absolutgeber Protokoll	[0] Keiner	TRUE	2-Set-ups		Uint8
32-33	MENCODER ABSRES	Absolutgeber Auflösung	8192 Pulse/U	TRUE	2-Set-ups		Uint32
32-34	MENCODERBAUD	Absolutgeber Baudrate X55	[4] 9600	FALSE	2 set-ups		Uint8
32-35	MENCODER DATLEN	Absolutgeber Datenlänge	25 Bit	TRUE	2-Set-ups		Uint8
32-36	MENCODER FREQ	Absolutgeber Taktfrequenz	262.000 kHz	TRUE	2-Set-ups		Uint32
32-37	MENCODER CLOCK	Absolutgeber Takterzeugung	[1] Ein	TRUE	2-Set-ups		Uint8
32-38	MENCODER DELAY	Absolutgeber Kabellänge	0	TRUE	2-Set-ups		Uint16
32-39	MENCODER MONITORING	Drehgeber-Überwachung	[0] Aus	TRUE	2-Set-ups		Uint8
32-40	MENCODER TERM	Drehgeber-Abschluss-widerstand	[1] Ein	TRUE	2-Set-ups		Uint8
32-43	MENCCONTROL	Enc.1 Steuerung	0	TRUE	2 set-ups		Uint8
32-44		Enc.1 Teilnehmer ID	127	TRUE	2 set-ups		Uint8
32-45		Enc.1 CAN Überwachung	[0] Off	TRUE	2 set-ups		Uint8

∗ Werkseinstellung **[ ] bei Kommunikation über serielle Schnittstelle benutzter Wert**

Par. Nr. #	Parametername	Parameterbeschreibung	Werkseinstellung	Ändern während des Betriebs	4-Par.-sätze	Umwandlungsindex	Typ
32-5*	**Rückführungsquelle**						
32-50		Rückführung Slave	[2] Enc2	TRUE	2-Set-ups		Uint8
32-52		Rückführung Master	[1] Enc1 X56	TRUE	2-Set-ups		Uint8
32-6*	**PID-Regelung**						
32-60	KPROP	Proportionalfaktor	30	TRUE	2-Set-ups	0	Uint32
32-61	KDER	Differentialwert für PID-Regelung	0	TRUE	2-Set-ups	0	Uint32
32-62	KINT	Integralfaktor	0	TRUE	2-Set-ups	0	Uint32
32-63	KILIM	Grenzwert für die Integralsumme	1000	TRUE	2-Set-ups	0	Uint16
32-64	BANDWIDTH	PID Bandbreite	1000	TRUE	2-Set-ups	0	Uint16
32-65	FFVEL	Geschwindigkeits-Feedforward	0	TRUE	2-Set-ups	0	Uint32
32-66	FFACC	Beschleunigungs-Feedforward	0 %	TRUE	2-Set-ups	0	Uint32
32-67	POSERR	Maximal tolerierter Positionsfehler	20000 qc	TRUE	2-Set-ups		Uint32
32-68	REVERS	Reversierungsverhalten Slave	[0] Reversieren	TRUE	2-Set-ups		Uint8
32-69	TIMER	Abtastzeit für PID-Regelung	1 ms	TRUE	2-Set-ups		Uint16
32-70	PROFTIME	Abtastzeit für Profilgenerator	[1] 1 ms	TRUE	2-Set-ups		Uint8
32-71	REGWMAX	Größe des Regelfensters (Aktivierung)	0 qc	TRUE	2-Set-ups		Uint32
32-72	REGWMIN	Größe des Regelfensters (Deaktivierung)	0 qc	TRUE	2-Set-ups		Uint32
32-73	KILIMTIME	Integralgrenzwert Filterzeit	0 ms	TRUE	2-Set-ups		int16
32-74	POSERRTIME	Schleppfehler Filterzeit	0 ms	TRUE	2-Set-ups		int16
32-8*	**Geschwindigkeit & Beschleunigung**						
32-80	VELMAX	Maximalgeschwindigkeit (Encoder)	1500 RPM	TRUE	2-Set-ups		Uint32
32-81	RAMPMIN	Kürzeste Rampe	1 s	TRUE	2-Set-ups		Uint32
32-82	RAMPTYPE	Rampenform	0	TRUE	2-Set-ups		Uint8
32-83	VELRES	Geschwindigkeitsteiler	100	TRUE	2-Set-ups		Uint16
32-84	DFLTVEL	Default-Geschwindigkeit	50	TRUE	2-Set-ups		Uint16
32-85	DFLTACC	Default-Beschleunigung	50	TRUE	2-Set-ups		Uint16
32-86	JERKMIN	Beschleunigungsrampe für Ruckbegrenzung	100 ms	TRUE	2 set-ups		Uint32
32-87	JERKMIN2	Beschleunigungsdauer für Ruckbegrenzung	0 ms	TRUE	2 set-ups		Uint32
32-88	JERKMIN3	Bremsrampe für Ruckbegrenzung	0 ms	TRUE	2 set-ups		Uint32
32-89	JERKMIN4	Bremsdauer für Ruckbegrenzung	0 ms	TRUE	2 set-ups		Uint32

∗ Werkseinstellung **[ ] bei Kommunikation über serielle Schnittstelle benutzter Wert**

□ MCO weitere Einstellungen, Parameterliste

Par. Nr. #	Parametername	Parameterbeschreibung	Werkseinstellung	Ändern während des Betriebs	4-Par.-sätze	Umwandlungsindex	Typ
33-0*	**Homefahrt**						
33-00	HOME_FORCE	Homefahrt erzwingen?	[0] nein	TRUE	2-Set-ups		Uint8
33-01	HOME_OFFSET	Nullpunkt-Offset bezüglich Home-Position	0 qc	TRUE	2-Set-ups		Int32
33-02	HOME_RAMP	Homefahrt-Rampe	10	TRUE	2-Set-ups		Uint16
33-03	HOME_VEL	Homefahrt-Geschwindigkeit	10	TRUE	2-Set-ups		Int16
33-04	HOME_TYPE	Homefahrt-Verhalten	[0] Reverse + Index	TRUE	2-Set-ups		Uint8
33-1*	**Synchronisation**						
33-10	SYNCFACTM	Synchronisationsfaktor Master (M:S)	1	TRUE	2-Set-ups		Int32
33-11	SYNCFACTS	Synchronisationsfaktor Slave (M:S)	1	TRUE	2-Set-ups		Int32
33-12	SYNCPOSOFFS	Positionsoffset für Synchronisation	0 qc	TRUE	2-Set-ups		Int32
33-13	SYNC ACCURACY	Genauigkeitsfenster für Positionssynchronisation	1000 qc	TRUE	2-Set-ups		Int32
33-14	SYNCVELREL	Relative Geschwindigkeits-begrenzung Slave	0 %	TRUE	2-Set-ups		Uint8
33-15	SYNCMARKM	Markeranzahl Master	1	TRUE	2-Set-ups		Uint16
33-16	SYNCMARKS	Markeranzahl Slave	1	TRUE	2-Set-ups		Uint16
33-17	SYNCMPULSM	Markerabstand Master	4096	TRUE	2-Set-ups		Uint32
33-18	SYNCMPULSS	Markerabstand Slave	4096	TRUE	2-Set-ups		Uint32
33-19	SYNCMTYPM	Markertyp Master	[0] Enc. Z pos.	TRUE	2-Set-ups		Uint8
33-20	SYNCMTYPS	Markertyp Slave	[0] Enc. Z pos.	TRUE	2-Set-ups		Uint8
33-21	SYNCMWINM	Master-Marker Toleranzfenster	0	TRUE	2-Set-ups		Uint32
33-22	SYNCMWINS	Slave-Marker Toleranzfenster	0	TRUE	2-Set-ups		Uint32
33-23	SYNCMSTART	Startverhalten für Marker-synchronisation	[0] Start Funkt. 1	TRUE	2-Set-ups		Uint16
33-24	SYNCFAULT	Markeranzahl für Fault	10	TRUE	2-Set-ups		Uint16
33-25	SYNCREADY	Markeranzahl für Ready	1	TRUE	2-Set-ups		Uint16
33-26	SYNCVFTIME	Geschwindigkeitsfilter	0 µs	TRUE	2-Set-ups	0	Int32
33-27	SYNCOFFTIME	Offset Filterzeit	0 ms	TRUE	2-Set-ups		Uint32
33-28	SYNCMFPAR	Markerfilter Konfiguration	[0] Marker Filter 1	TRUE	2-Set-ups		Uint8
33-29	SYNCMFTIME	Filterzeit für Markerkorrektur	0 ms	TRUE	2-Set-ups		Int32
33-30	SYNCM MAXCORR	Maximale Markerkorrektur	[0] Off	TRUE	2-Set-ups		Uint32
33-31	SYNCTYPE	Synchronisationstyp	[0] Standard	TRUE	2-Set-ups		Uint8

∗ Werkseinstellung **[ ] bei Kommunikation über serielle Schnittstelle benutzter Wert**

Par. Nr. #	Parametername	Parameterbeschreibung	Werkseinstellung	Ändern während des Betriebs	4-Par.-sätze	Umwandlungsindex	Typ
33-32	SYNCFFVEL	Geschw. Feedforward Anpassung	0	TRUE	2 set-ups		Uint32
33-33	SYNCVFLIMIT	Synchronisationsfehler Fenster	0 qc	TRUE	2 set-ups		Uint32
33-4*	**Grenzwertbehandlung**						
33-40	ENDSWMOD	Verhalten bei Endschalter	[0] Fehlerunterprogramm aufrufen	TRUE	2-Set-ups		Uint8
33-41	NEGLIMIT	Negative Software-Wegbegrenzung	-500000 qc	TRUE	2-Set-ups		Int32
33-42	POSLIMIT	Positive Software-Wegbegrenzung	500000 qc	TRUE	2-Set-ups		Int32
33-43	SWNEGLIMACT	Negative Software-Wegbegrenzung aktiv	[0] Inaktiv	TRUE	2-Set-ups		Uint8
33-44	SWPOSLIMACT	Positive Software-Wegbegrenzung aktiv	[0] Inaktiv	TRUE	2-Set-ups		Uint8
33-45	TESTTIM	Messzeit im Zielfenster	0 ms	TRUE	2-Set-ups		Uint8
33-46	TESTVAL	Zielfenster-Grenzwert	1 qc	TRUE	2-Set-ups		Uint16
33-47	TESTWIN	Zielfenster-Größe	0 qc	TRUE	2-Set-ups		Uint16
33-5*	**I/O Konfiguration**						
33-50	I_FUNCTION_1	Klemme X57/1 Digitaler Eingang	[0] keine Funktion	TRUE	2-Set-ups		Uint8
33-51	I_FUNCTION_2	Klemme X57/2 Digitaler Eingang	[0] keine Funktion	TRUE	2-Set-ups		Uint8
33-52	I_FUNCTION_3	Klemme X57/3 Digitaler Eingang	[0] keine Funktion	TRUE	2-Set-ups		Uint8
33-53	I_FUNCTION_4	Klemme X57/4 Digitaler Eingang	[0] keine Funktion	TRUE	2-Set-ups		Uint8
33-54	I_FUNCTION_5	Klemme X57/5 Digitaler Eingang	[0] keine Funktion	TRUE	2-Set-ups		Uint8
33-55	I_FUNCTION_6	Klemme X57/6 Digitaler Eingang	[0] keine Funktion	TRUE	2-Set-ups		Uint8
33-56	I_FUNCTION_7	Klemme X57/7 Digitaler Eingang	[0] keine Funktion	TRUE	2-Set-ups		Uint8
33-57	I_FUNCTION_8	Klemme X57/8 Digitaler Eingang	[0] keine Funktion	TRUE	2-Set-ups		Uint8
33-58	I_FUNCTION_9	Klemme X57/9 Digitaler Eingang	[0] keine Funktion	TRUE	2-Set-ups		Uint8
33-59	I_FUNCTION_10	Klemme X57/10 Digitaler Eingang	[0] keine Funktion	TRUE	2-Set-ups		Uint8
33-60	IOMODE	Klemme X59/1 und X59/2 Modus	[0] Ausgang	FALSE	2-Set-ups		Uint8
33-61	I_FUNCTION_11	Klemme X57/11 Digitaler Eingang	[0] keine Funktion	TRUE	2-Set-ups		Uint8
33-62	I_FUNCTION_12	Klemme X57/12 Digitaler Eingang	[0] keine Funktion	TRUE	2-Set-ups		Uint8

∗ Werkseinstellung **[ ] bei Kommunikation über serielle Schnittstelle benutzter Wert**

Par. Nr. #	Parametername	Parameterbeschreibung	Werkseinstellung	Ändern während des Betriebs	4-Par.-sätze	Umwandlungsindex	Typ
33-63	O_FUNCTION_1	Klemme X59/1 Digitaler Ausgang	[0] keine Funktion	TRUE	2-Set-ups		Uint8
33-64	O_FUNCTION_2	Klemme X59/2 Digitaler Ausgang	[0] keine Funktion	TRUE	2-Set-ups		Uint8
33-65	O_FUNCTION_3	Klemme X59/3 Digitaler Ausgang	[0] keine Funktion	TRUE	2-Set-ups		Uint8
33-66	O_FUNCTION_4	Klemme X59/4 Digitaler Ausgang	[0] keine Funktion	TRUE	2-Set-ups		Uint8
33-67	O_FUNCTION_5	Klemme X59/5 Digitaler Ausgang	[0] keine Funktion	TRUE	2-Set-ups		Uint8
33-68	O_FUNCTION_6	Klemme X59/6 Digitaler Ausgang	[0] keine Funktion	TRUE	2-Set-ups		Uint8
33-69	O_FUNCTION_7	Klemme X59/7 Digitaler Ausgang	[0] keine Funktion	TRUE	2-Set-ups		Uint8
33-70	O_FUNCTION_8	Klemme X59/8 Digitaler Ausgang	[0] keine Funktion	TRUE	2-Set-ups		Uint8
33-8*	**Globale Parameter**						
33-80	PRGPAR	Aktivierte Programmnummer	-1	TRUE	2-Set-ups	0	Uint8
33-81	Power-up State	Einschaltstatus	[1] Motor ein	TRUE	2-Set-ups		Uint8
33-82	STATUS MONITORING	Statusüberwachung Antrieb	[1] Ein	TRUE	2-Set-ups	0	Uint8
33-83	ERRCOND	Verhalten im Fehlerfall	[0] Leerlauf	TRUE	2-Set-ups		Uint8
33-84	ESCCOND	Verhalten bei Programmabbruch	[0] Geregelter Stopp	TRUE	2-Set-ups		Uint8
33-85	EXTERNAL24V	Externe 24VDC MCO Versorgung	[0] Nein	TRUE	2-Set-ups		Uint8
33-9*	**MCO Port Einstellungen**						
33-90		X62 MCO CAN Teilnehmer ID	127 N/A	TRUE	2 set-ups		Uint8
33-91		X62 MCO CAN Baudrate	[20] 125 kBaud	TRUE	2 set-ups		Uint8
33-94		X62 MCO RS485 serieller Abschluss	[0] Aus	TRUE	2 set-ups		Uint8
33-95		X62 MCO RS485 serielle Baudrate	[2] 9600 Baud	TRUE	2 set-ups		Uint8

□ MCO Datenanzeigen, Parameterliste

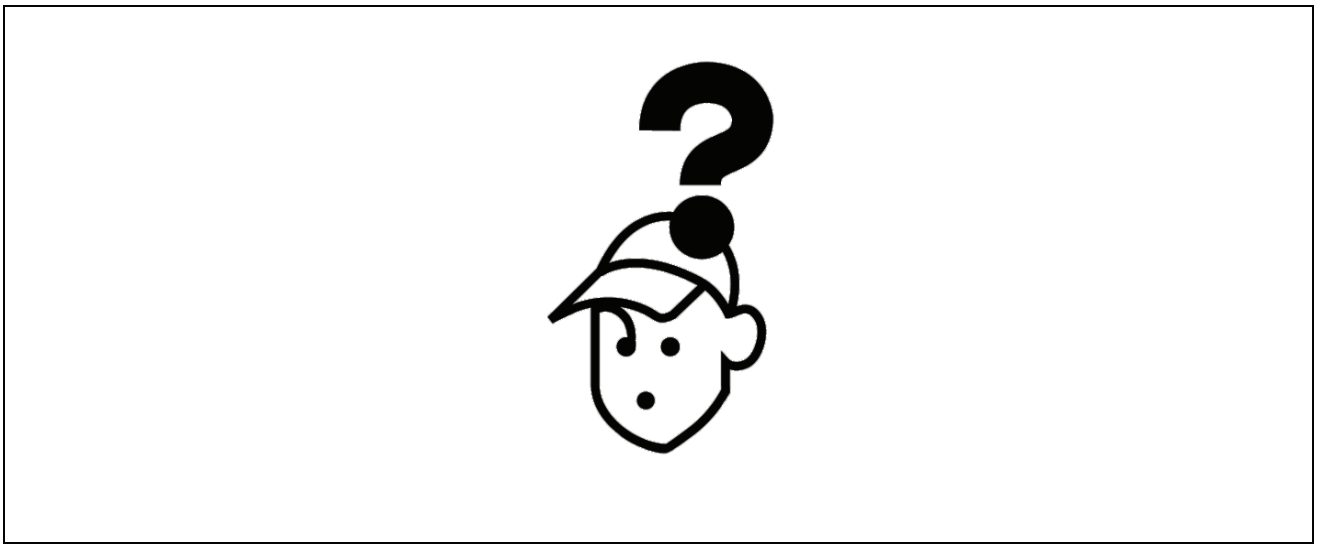
Par. Nr. #	Parametername	Parameterbeschreibung	Werks-ein-stellung	Ändern während des Betriebs	4-Par.-sätze	Umwand-lungs-index	Daten-typ
34-0*	**PCD Schreib-Parameter**						
34-01		PCD 1 nach MCO schreiben			All set-ups		Uint16
34-02		PCD 2 nach MCO schreiben			All set-ups		Uint16
34-03		PCD 3 nach MCO schreiben			All set-ups		Uint16
34-04		PCD 4 nach MCO schreiben			All set-ups		Uint16
34-05		PCD 5 nach MCO schreiben			All set-ups		Uint16
34-06		PCD 6 nach MCO schreiben			All set-ups		Uint16
34-07		PCD 7 nach MCO schreiben			All set-ups		Uint16
34-08		PCD 8 nach MCO schreiben			All set-ups		Uint16
34-09		PCD 9 nach MCO schreiben			All set-ups		Uint16
34-10		PCD 10 nach MCO schreiben			All set-ups		Uint16
34-2*	**PCD Lese-Parameter**						
34-21		PCD 1 von MCO lesen			All set-ups		Uint16
34-22		PCD 2 von MCO lesen			All set-ups		Uint16
34-23		PCD 3 von MCO lesen			All set-ups		Uint16
34-24		PCD 4 von MCO lesen			All set-ups		Uint16
34-26		PCD 6 von MCO lesen			All set-ups		Uint16
34-27		PCD 7 von MCO lesen			All set-ups		Uint16
34-28		PCD 8 von MCO lesen			All set-ups		Uint16
34-29		PCD 9 von MCO lesen			All set-ups		Uint16
34-30		PCD 10 von MCO lesen			All set-ups		Uint16
34-4*	**Eingänge & Ausgänge**						
34-40		Digitale Eingänge			All set-ups		Uint16
34-41		Digitale Ausgänge			All set-ups		Uint16

34-5*	**Prozessdaten**		**Unit**				
34-50		Istposition	BE		All set-ups		Int32
34-51		Sollposition	BE		All set-ups		Int32
34-52		Istposition Master	qc		All set-ups		Int32
34-53		Indexposition Slave	BE		All set-ups		Int32
34-54		Indexposition Master	qc		All set-ups		Int32
34-55		Kurvenposition			All set-ups		Int32
34-56		Schleppfehler	BE		All set-ups		Int32
34-57		Synchronisationsfehler	BE		All set-ups		Int32
34-58		Aktuelle Geschwindigkeit	BE/s		All set-ups		Int32
34-59		Aktuelle Master-Geschwindigkeit	qc/s		All set-ups		Int32

∗ Werkseinstellung **[ ] bei Kommunikation über serielle Schnittstelle benutzter Wert**

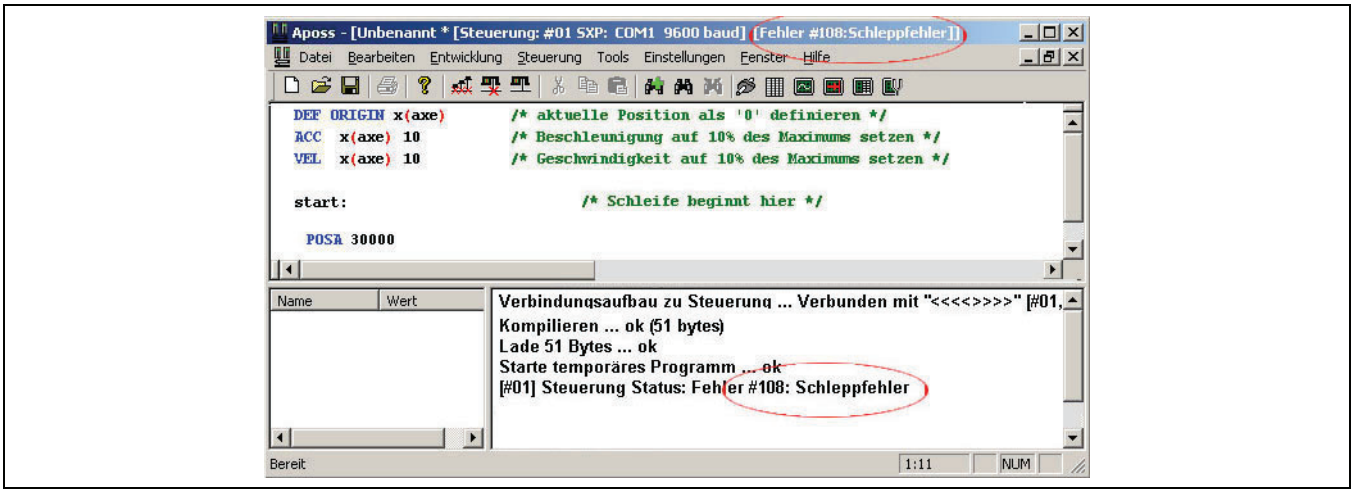
Par. Nr. #	Parametername	Parameterbeschreibung	Werks-ein-stellung	Ändern während des Betriebs	4-Par.-sätze	Umwand-lungs-index	Daten-typ
34-60		Synchronisationsstatus			All set-ups		Int32
34-61		Achsstatus			All set-ups		Int32
34-62		Programmstatus			All set-ups		Int32
34-7*	**Diagnoseanzeigen**						
34-70		MCO Alarmwort 1		FALSE	All set-ups		Uint32
34-71		MCO Alarmwort 2		FALSE	All set-ups		Uint32

Fehlersuche und -behebung



□ Warnungen und Fehlermeldungen

Alle Meldungen werden im LCP-Display des FC 300 in Kurzform und in der APOSS-Software mit der Fehlernummer und deren Bedeutung in der Titelleiste und im Kommunikationsfenster angezeigt:



Ab MCO 5.00 werden die letzten 50 Fehler seit dem Einschalten intern gespeichert. Diese Informationen werden nicht im Flash gespeichert. Sie können mit APOSS ausgelesen werden (siehe Fehler Historie) oder durch SDO-Zugriff. Zu jedem Fehler wird nicht nur die Zeit gespeichert, sondern auch zusätzliche Fehlerinformationen. Diese Information ist fehlerspezifisch.

Fehler Nr.	Fehlertext LCP	Fehlertext APOSS
102	Zu viele CAN Objekte	Keine weiteren CAN-Objekte verfügbar (CANINI).
103	Ung. Achsennr.	Achse nicht vorhanden.
105	Fehler n. quit.	Fehler nicht beseitigt
106	Ref.pkt n.err.	HOME noch nicht angefahren
107	Rf.pkt.geschw. 0	HOME Geschwindigkeit 0
108	Positionsfehler	Schleppabstand überschritten
109	Index n. gefunden	Indeximpuls (Encoder) nicht gefunden.
110	Undef. Befehl	Unbekannter Befehl
111	SW-Endbegren.	Software-Endschalter überschritten.
112	Undef. Param.	Falsche Parameternummer.
113	FC n. aktiviert	VLT Error Status
114	Zu viele Schliefen	Zu viele verschachtelte LOOP Befehle.
115	Par. Speich.-F.	INLONG Befehl erhielt einen ungültigen String.
116	Param. speicher	Parameter im Speicher defekt.
117	Progr. speicher	Programme im Speicher defekt.
118	Reset du. CPU	Reset durch CPU ausgelöst.
119	Benutzerabbruch	Abbruch durch Benutzer.
121	Keine SDO Kanäle mehr	Anzahl der SDO Kanäle überschritten.
125	HW Endschalter	HW Endschalter aktiviert.
149	Zu viele Interrupts.	Zu viele Interrupt-Funktionen.
150	K. ext. 24V	Externe Stromversorgung fehlt.
151	Zu viele GOSUB	Zu viele verschachtelte Unterprogramme.
152	Rückk. Überschr.	Zu viele RETURN Befehle.
154	D.-Ausgang Überl.	Digitaler Ausgang überlastet.
155	Verknüpf. Fehl.	LINKGPAR Befehl fehlgeschlagen.
156	Ungültiges DOUBLE Arg.	Aufruf einer Gleitkommafunktion mit unzulässigem Argument.
160	Interner Interrupt Fehler	Interrupt, jedoch die Interrupt-Adresse ist nicht mehr gültig.
162	Speicherfehler	Fehler beim Verifizieren.
170	Array Größe (DIM)	Zu viele DIM Arrays definiert.
171	Array zu klein	Array zu klein
175	Zu wenig Array Speicher	Zu wenig Speicher für ein neues durch DIM definiertes Array.
176	Array Größe falsch	Array Größe passt nicht zum vorhandenen Array.
179	WAITNDX TO	Timeout beim Warten auf Index.
184	Zu viele ON TIME	Zu viele ONTIME oder ONPERIOD Interrupts.
187	N. genug Speich.	Kein Platz mehr für Variablen.
188	CAN Guarding Fehler	CAN-Guarding unterbrochen.
189	CAN Sende-Empfangsfehler.	CAN Sende- oder Empfangsfehler.

Fehler Nr.	Fehlertext LCP	Fehlertext APOSS
190	Speicher gesperrt	Programmspeicher ist schreibgeschützt.
191	Ung. Kurvennr.	Kurven-Array in DIM-Anweisung falsch.
192	Drehgeb.fehler	Encoder Fehler
193	Stack Überlauf	Stack zu klein (zu viele lokale Variablen oder verschachtelte Funktionsaufrufe)
194	Kein dynamischer Speicher	Kein dynamischer Speicher verfügbar.
195	Zu viele Testindizes	Zu viele Testindizes im Befehl TESTSETP.
196	Befehlscode zu alt	Befehlscode zu alt für die aktuelle Firmware.
198	SW Endschalter überschritten	Falsche Richtung nachdem Endschalter ausgelöst wurde.
199	Interner MCO Fehler	Interner MCO Fehler

Fehler 102

Keine weiteren CAN-Objekte verfügbar (CANINI).

CANINI Guarding meldet einen Fehler, da keine weiteren CAN-Objekte erzeugt werden können. Die zusätzliche Fehlerinformation (siehe Fehler Historie) wird wie folgt benutzt:

CN_TIMEOUT	-2	// Timeout eines CAN Befehls beim Senden oder Lesen // von Telegrammen
NO_HARDWARE	-6	// keine CAN Hardware vorhanden
NO_MEMORY	-7	// keine weiteren Einträge möglich (Mailboxen oder Listen)
NO_CANMEMORY	-10	// keine weiteren Mailboxen verfügbar für den define Befehl
NO_MOBJ	-11	// die angeforderte Mailbox ist nicht verfügbar
CN_CANERROR	-12	// a CAN Bus Fehler wurde erkannt (Low Level Bus Error)
CN_MOBJ_DIRERR	-13	// Richtung der Mailbox ist falsch (Versuch eine Write-Box // zu lesen oder umgekehrt)
NO_USER	-33	// liefert Wert für SDO Status (SDOSTATE).

SDO_ABORT	-50	// muss höher als die CN_ error Meldung sein
SDO_ID_NOT_IN_USE	-33	// liefert Wert für SDO Status (SDOSTATE)
SDO_SEG_ARRAY_TOO_SMALL	-51	
SDO_SEG_TOGGLE_ERROR	-52	
SDO_SEG_TOO_MUCH_DATA	-53	
SDO_SEG_NOT_ENOUGH_DATA	-54	
SDO_SEG_ARRAY_WRITE_ERROR	-55	

GUARD_ERROR_NOT_OPERATIONAL	-101
GUARD_ERROR_TOGGLE	-102
GUARD_ERROR_MODULE_NOT_RESPONSE	-103
GUARD_ERROR_NO_MODULE	-104

Fehler 103

Ungültige Achsennummer.

Ein Bewegungsbefehl adressiert eine Achsennummer größer 1.

Fehler 105

Fehler nicht quittiert

Es wurde versucht einen Bewegungsbefehl auszuführen, obwohl eine aktuell bestehende Fehlermeldung noch nicht gelöscht wurde.

Quittieren Sie erst die Fehlermeldung und führen dann den Bewegungsbefehl aus.

Fehler 106

Referenzpunkt noch nicht erreicht

„Referenzfahrt erzwingen" par. 33-00 *Homefahrt erzwingen?* ist aktiviert und es wurde vor der Referenzfahrt ein Bewegungsbefehl gegeben.

Führen Sie zuerst die Referenzfahrt aus und geben dann erst den Bewegungsbefehl.

Fehler 107

Referenzpunkt-Geschwindigkeit 0

Es wurde versucht, eine Referenzfahrt (HOME) auszuführen, aber die Referenzpunkt-Geschwindigkeit ist in Par. 33-03 *Homefahrt-Geschwindigkeit* auf [0] gesetzt.

Fehler 108

Positionsfehler

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

Undefinierter Befehl

Ein Kommunikations- oder Programmfehler. Das Programm muss neu kompiliert und neu geladen werden.

Fehler 111

SW-Endbegrenzung

Ein Fahrbefehl wird die Software-Endschalter aktivieren oder hat sie bereits aktiviert. Diese Begrenzungen werden aktiviert, wenn die Ist- oder Sollposition außerhalb der in Par. 33-43 und 33-44 *Negative* und *Positive Software-Wegbegrenzung* festgelegten Softwaregrenzen liegt.

Bei einer Bewegung im Drehzahlmodus wird das Überschreiten der Wegbegrenzung erst erkannt, nachdem die aktuelle Position mit dem Software-Endschalter identisch ist.

Ab MCO 5.00 ist es möglich einen Software-Endschalterfehler zu löschen und dann in die entgegengesetzte Richtung zu fahren. Wenn allerdings versucht wird, erneut in die falsche Richtung zu fahren, wird der Fehler 198 erzeugt.

Hardware-Endschalter werden genauso behandelt wie Software-Endschalter.

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.

End- und Referenzschalter erlauben die Verwendung jedes Eingangs; es werden also nicht nur 1..8 unterstützt, sondern auch größere Nummern.

Behandlung eines Software-Endschalterfehlers bis MCO 5.00: Ein Software-Endschalterfehler kann nicht gelöscht werden: 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.

Fehler 112

Undefinierter Parameter

Es wurde versucht, mit einem SET oder SETVLT Befehl einen Parameter zu verändern, den es nicht gibt.

Fehler 113

FC nicht aktiviert

FC 300 ist nicht eingeschaltet, aber der Positionsregler MCO 305 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:

- FC 300 hat einen Alarm,
- FC 300 ist im Hand-Betrieb,
- Par. 8-02 Aktives Steuerwort steht nicht auf Option C0.

Fehler 114

Zu viele Schleifen

Im ausgeführten Programm sind zu viele ineinander geschachtelte LOOP Befehle.

Fehler 115

Parameter speichern Fehler

Fehler beim Speichern des Optionsparameters.

Der Befehl INLONG wurde benutzt, um einen Long Wert von der seriellen Leitung zu lesen. Wenn der ankommende String keine gültige Nummer repräsentiert, wird dieser Fehler ausgegeben.

Fehler 116

Param. im Speicher defekt

Die Parameter im EPROM sind nicht mehr gültig, weil

- EPROM 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 Benutzerparametern überschreiben.

Fahrprogramme, die Anwendungsparameter voraussetzen, würden sonst nicht mehr korrekt funktionieren.

Mit APOSS stand-alone können Sie dazu auch den Menü-Befehl *Steuerung* → *Parameter* → *Reset* benutzen.

Fehler 117

Programmspeicher

Die im EPROM gespeicherten Programmdaten können nicht mehr gefunden werden oder sind falsch, weil

- das EPROM defekt ist oder
- Stromausfall beim Speichern.

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.

Dies entspricht dem Befehl *Reset* → *Vollständig* im APOSS Menü *Steuerung*.

Mit APOSS stand-alone löschen Sie das EPROM mit *Steuerung* → *Speicher* → *EPROM löschen* und laden anschließend alle Programme und Parameter neu in die Steuerung.

Fehler 118

Reset durch CPU

Der Prozessor wurde angehalten und ein automatisches Reset (eine automatische Quittierung) ausgeführt (Watchdog).

Ursachen können sein:

- Kurzzeitiger Spannungsabfall,
- Spannungsspitze oder
- Kurzschluss.

Fehler 119

Benutzerabbruch

Das Autostart-Programm wurde durch den Benutzer abgebrochen.

Oder die [CANCEL]-Taste wurde während des Hochfahrens gedrückt und damit ein Master-Reset ausgelöst.

Fehler 121

Keine SDO Kanäle mehr

Die mögliche Anzahl der SDO Kanäle wurde überschritten.

Wenn ein Befehl SDOREAD oder SDOWRITE mit einem negativen Index benutzt wurde, wird das Ergebnis sofort zurückgeliefert und die laufenden SDO in einen Kanal gespeichert. Dieser wird freigegeben, sobald das Ergebnis gelesen wurde.

Es gibt maximal 5 Kanäle.

Fehler 125

HW Endbegren.

Ein Fahrbefehl hat den Hardware-Endschalter einer Achse aktiviert.

Durch diese Aktivierung wurde die Steuerung (abhängig von Par. 33-40 *Verhalten bei Endschalter*) automatisch abgeschaltet und der Antrieb muss (bis Versionen < MCO 5.00) manuell aus dieser Position herausgefahren werden, bevor die Fehlermeldung zurückgesetzt werden kann.

Das Verhalten im Fall von Hard- und Software-Endschalter wurde mit MCO 5.00 verbessert: Es ist nun möglich, einen Endschalterfehler zu löschen und dann den Antrieb in die entgegengesetzte Richtung zu fahren. Aber wenn versucht wird, erneut in die falsche Richtung zu fahren, wird der Fehler 198 ausgegeben.

End- und Referenzschalter erlauben den Einsatz eines belieben Eingangs; es werden also auch höhere Nummern unterstützt.

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 posint GOSUB: ON APOS, ON IPOS, ON MAPOS, ON MCPOS, ON MIPOS

Fehler 150

Keine externe 24V

Externe 24 V 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.

Vermeiden Sie zu viele gegenseitige (maximal 10!) und möglichst auch rekursive (sich selbst aufrufende) Subroutinen.

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

Digital-Ausgang überlastet

Digitaler Ausgang überlastet.

Fehler 155

Verknüpfungs-Fehler

LINKGPAR Befehl fehlgeschlagen.

Fehler 156

Ungültiges DOUBLE Argument

Mathematischer Fehler (Floating Point): Ungültige Datenargumente in einer der „double" Funktionen, d.h. eine Gleitkommafunktion wurde mit einem unzulässigen Argument aufgerufen. Zum Beispiel erhielt sqrt eine negative Zahl, oder asin oder acos wurden mit einem Argument größer 1 aufgerufen.

Double Funktion ist ab MCO 5.00 verfügbar.

Fehler 160

Interner Interrupt Fehler

Ein Interrupt trat auf, jedoch ist die Interrupt-Adresse nicht mehr gültig. (Interner Fehler, der nie vorkommen sollte.)

Fehler 162

Speicherfehler

Nach einem Speichervorgang ins EPROM (Programm oder Parameter) wurde beim Verifizieren ein Fehler festgestellt.

Löschen Sie das EPROM mit einem 3-Finger-Reset und versuchen Sie noch einmal das Programm oder die Parameter zu speichern.

In APOSS stand-alone können Sie stattdessen auch den Menübefehl *Steuerung → Speicher → EPROM löschen* benutzen.

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 im 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 mit *Steuerung → Speicher → EPROM löschen* oder benutzen Sie den Befehl *Steuerung → Reset → 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 175

Zu wenig Array Speicher

Es gibt keinen Speicherplatz mehr für ein neues durch eine DIM Anweisung definiertes Array.

Fehler 176

Array Größe falsch

Die Größe in einer DIM Anweisung entspricht nicht der Größe des vorhandenen Arrays.

Die Felder in der Steuerung könnten von älteren APOSS-Programmen stammen und das aktuelle Programm hat andere Definitionen.

Korrigieren Sie die DIM Anweisung oder löschen Sie die vorhandenen Arrays mit *Steuerung → Speicher → EPROM löschen* oder benutzen Sie den Befehl *Steuerung → Reset → Arrays.*

ACHTUNG!:
Beachten Sie aber die Ratschläge zur Sicherung der Programme und Parameter, bevor Sie das EPROM löschen.

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 183

Ungültiges Argument

Dieser Fehler zeigt an, dass ein TESTSTOP Befehl einen ungültiges Argument enthält.

Oder ein Compiler-Fehler in anderen Befehlen, die ungültige Parameterwerte, ein ungültiges Format oder einen ungültigen Bereich enthalten. (Interner Fehler, der normalerweise nicht auftritt.)

Fehler 184

Zu viele ON TIME

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

Nicht genug Speicher

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 mit *Steuerung* → *Programme* → *Alle löschen*

Oderlöschen Sie beides, die Programme und Arrays, d.h. durch Löschen des gesamten Speichers mit *Steuerung* → *Speicher* → *EPROM löschen* benutzen.

ACHTUNG!:
Beachten Sie aber die Ratschläge zur Sicherung der Programme und Parameter, bevor Sie das EPROM löschen.

Fehler 188

CAN Guarding Fehler

CAN-Guarding meldet einen Fehler. Dies passiert, entweder wenn Guarding-Meldungen vom Slave angefordert werden oder wenn Guarding durch den Master durchgeführt wurde. In beiden Fällen wird die Meldung durch ein Timeout verursacht. Die zusätzliche Fehlerinformation zeigt, ob der Fehler durch ein Master-Guarding verursacht wurde.

Diese zusätzlichen Fehlerinformationen (siehe Fehler Historie) werden genutzt wie in Fehler 102 gezeigt.

Fehler 189

CAN Sende-Empfangsfehler.

Dies ist eine Sende- oder ein Empfangsfehler verursacht durch einen SDOREAD oder SDOWRITE Befehl, durch einen CANIN oder CANOUT Befehl, oder durch einen IN oder OUT Befehl beim Einsatz von CAN-I/Os.

Die zusätzliche Fehlerinformation enthält entweder die CAN-ID, die den Fehler erzeugte (IN, OUT, SDO, …) oder die Objektnummer (handle), die benutzt worden war (CANIN, CANOUT).

Fehler 190

Speicher gesperrt

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 → *EPROM löschen* nicht ausgeführt.

Fehler 191

Ungültige Kurvennummer

In der DIM-Anweisung für SETCURVE ist ein falsches oder altes Array definiert.

Ein alters Array könnte existieren, wenn die zbc-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 (→ *öffnen* und → *speichern*).
- Oder die Reihenfolge eines Arrays in der DIM-Anweisung stimmt nicht mit der Reihenfolge in der zbc-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 193

Stack Überlauf

Interner Fehler: Stack Overflow verursacht durch zu viele lokale Variablen oder zu viele verschachtelte Funktionsaufrufe.

Erhöhen Sie die Stack-Größe in *Einstellungen → Compiler.*

Fehler 194

Kein dynamischer Speicher

Es gibt nicht genug dynamischen Speicher für das angeforderte Datenprotokoll (TESTSETP) Entweder weil TESTSTART einen zu großen dynamischen Speicher belegt oder wiederholend aufgerufen wird.

Fehler 195

Zu viele Testindizes

Der Befehl zum Protokollieren der Daten (TESTSETP) enthält zu viele Indizes. Diese sind derzeit begrenzt auf 20.

Fehler 196

Befehlscode zu alt

Der Compiler, der den Programmcode erzeugt hat, war zu alt für diese Firmware.

Bitte verwenden Sie eine neuere APOSS-Version.

Fehler 198

SW Endschalter überschritten

Nach dem Erreichen des Endschalters und Löschen des Fehlers wurde versucht, erneut in die falsche Richtung zu fahren.

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.

Anschluss-Pin nicht erlaubt

Anschluss-Pin ist nicht erlaubt in Zeile .. Spalte ..

Im OUT Befehl wurde eine ungültige Kombination oder Pin-Nummer verwendet, die so nicht gesetzt werden kann.

Fehler beim Compilieren

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 → Programme→ Dieses 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.

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.

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 .. aus!

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

Timeout: Keine Antwort vom FC

Der FC 300 antwortet nicht; überprüfen Sie die Verbindung.

Verbindung abgebrochen

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 ... besteht bereits

Verbindung zu .. besteht bereits – 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.

Index

[

[Esc]-Taste ... 59

1

19-** Anwendungsparameter ... 203
19-00 ...19-89 Anwendungsparameter ... 203
19-90 .. 19-99 Nur-Lesen Anwendungsparameter ... 203

3

32-0* Drehgeber 2 - Slave ... 204
32-3* Drehgeber 1 - Master ... 210
33-0* Homefahrt ... 220
33-1* Synchronisation ... 222
33-4* Grenzwertbehandlung ... 233
33-5* I/O Konfiguration ... 235
33-8* Globale Parameter ... 240
33-9* MCO Port Einstellungen ... 242
34-0* PCD Schreib-Parameter ... 243
34-2* PCD Lese-Parameter ... 243
34-4* Eingänge & Ausgänge ... 243
34-5* Prozessdaten ... 243
34-7* Diagnoseanzeigen ... 244

A

Abbrechen [Esc] und Fortsetzen ... 67
Abbruch alle ... 69
Achsenparameter ... 77
Anwendungsbeispiel
  Absolute Positionierung ... 19
  Geschwindigkeitssynchronisation ... 25
  Kartons bedrucken mit Markerkorrektur ... 39
  Kartons mit Datum stempeln ... 36
  Koffertransportband (SNCV) ... 25
  Mechanische Bremssteuerung ... 48
  Palettierer ... 19
  Relative Positionierung ... 21
  Touch-Probe Positionierung ... 22
  Verpacken mit festen Produktabständen ... 27
  Verpacken mit variierenden Abständen und Schlupf 31
Anwendungseinstellungen ... 203
Anwendungsparameter ... 199
APOSS Fenster ... 58
APOSS Zahlenformate ... 169
Arithmetik ... 175
Arrays ... 170
  Arrays versus Variablen ... 172
  schreiben und lesen ... 172
Assignment Operation ... 177
Ausführen [F5] ... 66
Autostart ... 74

B

BANDWIDTH ... 214
Befehle
  Befehle zum Initialisieren ... 181
  CAM-Befehle ... 189
  Drehzahlregelung ... 188
  Ein-/Ausgänge (I/O) ... 183
  Handhabung der Parameter ... 187
  Interrupt-Funktionen ... 185
  Kommunikationsoption ... 188
  Positionierbefehle ... 188
  Steuerungsbefehle ... 182
  Synchronisationsbefehle ... 189
Befehls-Ausführungszeiten ... 165
Befehlshilfe [F12] ... 71
  Einfügen ... 71
Befehlsstruktur ... 166
Beispiel ... 63
Benutzereinheiten [BE] ... 8
Binär Programm Sichern /Rücklesen ... 73
Bitoperatoren ... 176

C

CAM Editor
  Symbolleiste ... 95
CAM Kontextmenüs ... 97
CAM Registerkarten ... 98
CAM-Editor ... 93
  Fenster ... 95
  Zoomen und Verschieben ... 97
CAM-Editor starten
  APOSS stand-alone ... 94
  mit MCT 10 ... 94
CAM-Modus ... 35
CAN Basic Library ... 190
CANBAUD ... 242
CANNR ... 242
Closed-Loop ... 9

D

Debugging ... 167
DFLTACC ... 218
DFLTVEL ... 218
Digitale Ausgänge

Funktionen 238
DIM Anweisung 171
Download alle 70
Drehgeber 15
Drehgeber-Drehrichtung 6

E

Editierfenster 58, 60
Eingabebereich 204
Einstellungen für die Anwendung 203
Einstellungen Kurvenprofil 103
ENCCONTROL 209
ENCODER 205
Encoder 1 CAN Überwachung 213
Encoder 1 Teilnehmer ID 213
Encoder CAN Überwachung 209
Encoder Teilnehmer ID 209
ENCODERABSRES 206
ENCODERABSTYPE 206
ENCODERBAUD 206
ENCODERCLOCK 207
ENCODERDATLEN 206
ENCODERDELAY 207
ENCODERFREQ 206
ENCODERMONITORING 207
ENCODERTYPE 204
ENDSWMOD 233
ERRCOND 240
Error Handling 166
ESCCOND 241
EXTERNAL24V 241

F

FC 300 Parameter 199
FC 300 Parameterübersicht 201
Fehlerhandhabung 166
Fehlermeldungen 255
Feldbus-Schnittstelle 14
FFACC 215
FFVEL 214
Fixpunkte 104
bearbeiten 104
Typ 105
Funktionstasten 59

G

Genauigkeit 35
Gepufferte Nachrichtenübertragung 190

H

HOME_FORCE 220
HOME_OFFSET 220
HOME_RAMP 221
HOME_TYPE 221
HOME_VEL 221

I

I_FUNCTION_11 und 12 237
I_FUNCTION_n 236
Indizes 172
Initialisierung auf die Werkseinstellungen 200
Interpolation 35
Interrupts 167
Interrupt Schachtelung 169
ON PERIOD innerhalb von Interrupt-Prozeduren 168
Prioritäten 168
Reaktionszeiten 168
Variable innerhalb von Interrupt-Prozeduren 167
IOMODE 237

J

J Befehlshilfe [F12]
Jetzt ausführen 71
JERKMIN 218, 219, 220

K

KDER 214
KILIM 214
KILIMTIME 216
KINT 214
Kommunikationsfenster 58
Konfigurationsbeispiele 13
Konstanten 170
Kontextmenüs 58
KPROP 214
Kurvenprofil-Diagramm 96
Kurvenscheibensteuerung 35
Festlegung des Markerabstandes 41
Sensorabstand ist größer als 1 Masterzyklus L. 41
Slave-Synchronisation mit Marker 43
Synchronisation mit Marker 39

L

Lesezeichen 65
Literatur 4
logische Verknüpfungen 177

M

Makro Aufzeichnen / Abrufen 62
Makros löschen 65
Master Units [MU] 9

Maus-Funktionen 60
MCO 305 11
MCO Datenanzeigen 243
MCO Grundeinstellungen 204
MCO Parameter 199, 204
MCO weitere Einstellungen 220
Mechanische Bremssteuerung 48
Meldungen -> Log-Datei 67
MENCCONTROL 212
MENCODER 210
MENCODERABSRES 211
MENCODERABSTYPE 210
MENCODERBAUD 211
MENCODERCLOCK 211
MENCODERDATLEN 211
MENCODERDELAY 212
MENCODERFREQ 211
MENCODERMONITORING 212
MENCODERTERM 212
MENCODERTYPE 210
Menü Bearbeiten 64
Menü Datei 63
Menü Entwicklung 66
Menü Steuerung 72
MLONG 6

N

NEGLIMIT 233
Nockenschaltwerk 47
NOWAIT in Interrupts 169

O

O_FUNCTION_n 239
Online / Offline Parameter 6
Open-Loop 9
Operatoren 175

P

Parameter 76
  Achsen 77
  Allgemeine Information zu den Parameterwerten . 204
  ändern und speichern 201
  auf Parameter zugreifen 199
  Global 76
  lesen und schreiben 200
  Name 77
  Speichern in Datei 77
  Wiederherstellen aus Datei 78
Parameterliste 246
  Anwendungsparameter 246
  MCO Datenanzeigen 252
  MCO Grundeinstellungen 247
  weitere Einstellungen 249
PID-Regelung 14
POSDRCT 207
POSERR 215
POSERRTIME 217
POSFACT_N 208
POSFACT_Z 208
Positionierung 17
  absolute 17
  relative 18
  Touch-Probe 18
POSLIMIT 233
PRGPAR 240
Priorität der Operatoren und Operationen 177
PROFTIME 216
Programm
  Rücklesen 73
  sichern 73
  Sichern mit Quellcode 73
  Starten 74
Programmausführung 15
Programmbeispiel
  Geschwindigkeitssynchronisation 25
  Kartons bedrucken mit Markerkorrektur 40
  Kartons mit Datum stempeln 38
  Markersynchronisation 33
  Mechanische Bremssteuerung 48
  Nockenschaltwerk 47
  Palettierer 20
  Positionssynchronisation 28
  Relative Positionierung 21
  Slave-Synchronisation mit Marker 46
  Touch-Probe Positionierung 23
Programmbeispiele Überblick 195
Programme löschen 70, 76
Programmende 63
Programmlayout 163

Q

Quadcounts 6

R

RAMPMIN 217
RAMPTYPE 217
Raw Telegramm Modus 190
Registerkarte
  Kurvendaten 98
  Kurveninfo 101
  Sync Marker 102
  Synchronisation 102
REGWMAX 216
REGWMIN 216

Relais Option MCB 105 14
Reset
  Arrays 79
  Daten im LCP-EEPROM 79
  Parameter 79
  Vollständig 79
REVERS 215
RSBAUDRATE 242
RSTERMINATION 242
Ruckbegrenzung 50
  Beispiele 52
Rückführung Master 213
Rückführung Slave 213
Rückgängig 62

S

Schnittstelle
  Alle Schnittstellen schließen 71
  schließen 71
Schnittstellen 14
SDO2 191
Sequentielle Befehlsabarbeitung 165
Speicher
  Daten im LCP-EPROM sichern 79
  EPROM löschen 78
  RAM speichern 78
Sprachelemente 169
Start alle 70
Start-Stop-Punkte 107
  bearbeiten 107
STATUSMONITORING 240
Steuerung
  Programme 72
Steuerung auswählen 70
Suchen in Dateien 65
Suchen und Ersetzen 64
SWNEGLIMACT 233
SWPOSLIMACT 234
Symbole 5
Symbolleiste 58
SYNCACCURACY 223
SYNCFACTM 222
SYNCFACTS 222
SYNCFAULT 227
SYNCFFVEL 232
Synchronisation 24
  Geschwindigkeitssynchronisation (SYNCV) 24
  Markersynchronisation (SYNCM) 31
  Position/Winkel-Synchronisation (SYNCP) 27
SYNCMARKM 224
SYNCMARKS 224
SYNCMFPAR 229
SYNCMFTIME 229
SYNCMMAXCORR 231
SYNCMPULSM 224
SYNCMPULSS 225
SYNCMSTART (mit Markerkorrektur) 226
SYNCMTYPM 225
SYNCMTYPS 225
SYNCMWINM 226
SYNCMWINS 226
SYNCOFFTIME 228
SYNCPOSOFFS 223
SYNCREADY 227
SYNCTYPE 231
SYNCVELREL 224
SYNCVFLIMIT 232
SYNCVFTIME 228
Systemüberblick 12

T

Tab umwandeln 65
Tabulatoren 62
Tangentenpunkte für gerade Abschnitte 35
Tastatur-Funktionen 60
TESTTIM 234
TESTVAL 234
TESTWIN 235
TIMER 215
Titelleiste 58

U

Überwachungsfenster 58

V

Variablen 170
VELMAX 217
VELRES 218
Vergleichsoperationen 177
Virtueller Master 7
VLT5000 > MCO 305 Konvertierung 69

W

Warnungen 255
Werteeingaben 166

Z

Zeilennummer 62
Zum Antrieb schreiben 63
Zuweisung 177