

Inhaltsverzeichnis

■ MCO 305 Befehlsreferenz lesen	3
□ Befehlsreferenz lesen	3
□ Verfügbare Literatur für FC 300, MCO 305 und MCT 10 Motion Control Tool.....	4
□ Symbole und Konventionen	5
□ Abkürzungen	5
□ Definitionen	6
■ Befehlsreferenz	9
■ Anhang	149
□ Neues in der aktuellen Version ab MCO 5.00	149
□ Technische Referenz	154
□ Abbildungen	162
□ Stichwortverzeichnis.....	165

Copyright

© Danfoss A/S, 2010

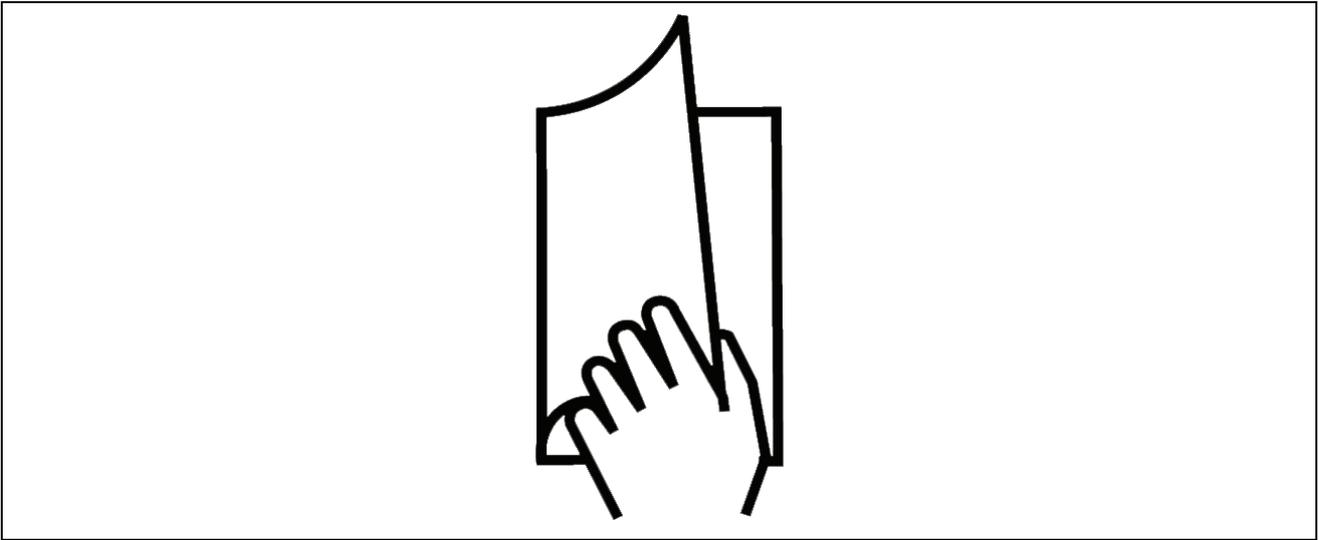
Warenzeichen

VLT ist ein eingetragenes Warenzeichen von Danfoss.

Hiperface® ist ein 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.

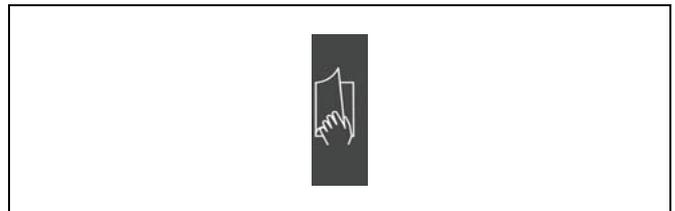
MCO 305 Befehlsreferenz lesen



□ Befehlsreferenz lesen

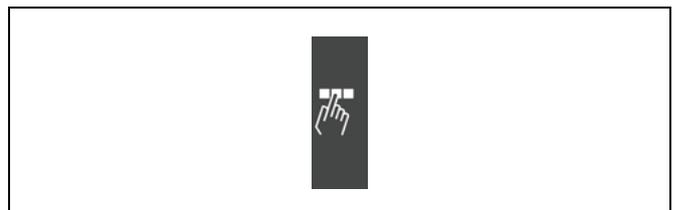
Die Befehlsreferenz ergänzt das MCO 305 Projektierungshandbuch mit der detaillierten Beschreibung aller Befehle. Bitte lesen Sie auch das Produkthandbuch, um sicher und professionell mit dem System zu arbeiten und beachten Sie vor allem auch die Sicherheitshinweise und allgemeinen Warnungen.

Das Kapitel **Befehlsreferenz lesen** informiert über die Symbole, Abkürzungen und Definitionen, die in diesem Handbuch benutzt werden.



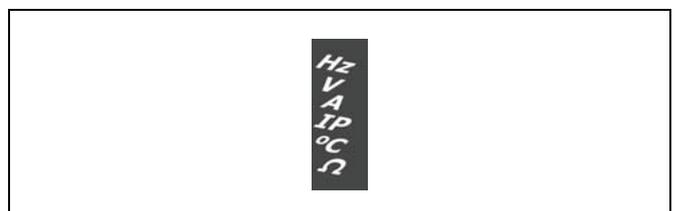
Seitenteiler für „Befehlsreferenz lesen“.

Das Kapitel **Befehlsreferenz** enthält die detaillierte Beschreibung aller Befehle mit deren Syntax sowie Programmbeispielen.



Seitenteiler für „Befehlsreferenz“.

Das Kapitel **Anhang** berichtet über „Neues in der aktuellen Version“. Erfahrene Anwender finden ausführliche Informationen in der Technischen Referenz zum Beispiel „Array Structure of CAM Profiles“. Das Handbuch schließt mit einem Stichwortverzeichnis.



Seitenteiler für „Anhang“.

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

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

- Das MCO 305 Produkthandbuch liefert die erforderlichen Informationen zum Einbau und für die Inbetriebnahme des MCO 305 sowie für die Optimierung der Steuerung.
- Das MCO 305 Projektierungshandbuch enthält alle technischen Informationen über die Optionskarte sowie Informationen für die Realisierung kundenspezifischer Designs und Anwendungen.
- Diese 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.

□ Symbole und Konventionen

In diesem Handbuch verwendete Symbole:



ACHTUNG!:

Kennzeichnet einen wichtigen Hinweis.



Kennzeichnet eine allgemeine Warnung.

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 Funktionstasten 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
Benutzereinheiten	BE
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

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

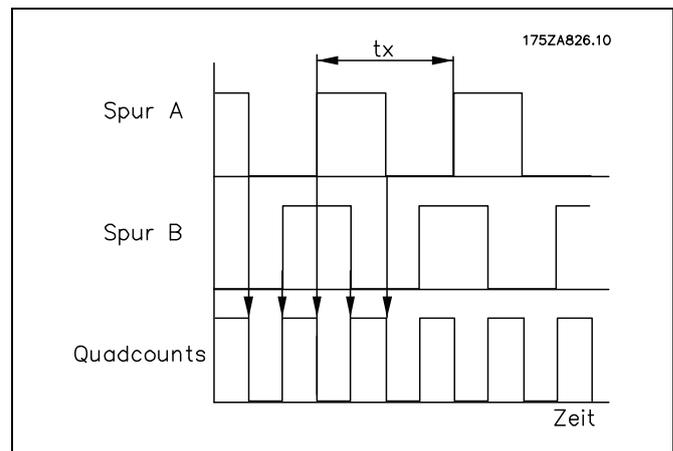
MLONG = 1.073.741.823

□ 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.

□ 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 $\text{delta_master} * 1/2^{32}$ 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.

Master Units [MU]

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

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

Par. 33-10 *Synchronisationsfaktor Master* SYNCFACTM

Par. 33-11 *Synchronisationsfaktor Slave* SYNCFACTS

Befehlsreferenz



Im folgenden Abschnitt finden Sie alle Befehle in alphabetischer Reihenfolge ausführlich beschrieben mit Syntax-Beispielen sowie kurzen Programmbeispielen. Bitte lesen Sie auch den Abschnitt „Grundlagen“ im Kapitel „Programmieren“ im MCO 305 Projektierungshandbuch.

□ ACC

Kurzinfo Beschleunigung für Fahrbefehle setzen

Syntax ACC a

Parameter a = Beschleunigung

Beschreibung Der Befehl ACC bestimmt die Beschleunigung für die nächsten Fahrbefehle (Drehzahl-, Positionier- oder Synchronisationsmodus). Der Wert bleibt solange gültig, bis mit einem weiteren ACC Befehl eine neue Beschleunigung gesetzt wird. Der Wert bezieht sich auf die Parameter 32-81 *Kürzeste Rampe* und 32-80 *Maximalgeschwindigkeit* sowie 32-83 *Geschwindigkeitsteiler*.

**ACHTUNG!:**

Wurde vor einem Fahrbefehl noch keine Beschleunigung definiert, wird mit dem Default-Wert aus Par. 32-85 *Default-Beschleunigung* beschleunigt.

ACHTUNG!:

Wenn die MCO 305 zum Steuern des FC 300 benutzt wird, sollten die Rampen immer über die Optionskarte eingestellt werden und nicht im FC 300. Die FC 300-Rampen müssen dabei immer auf Minimum stehen.

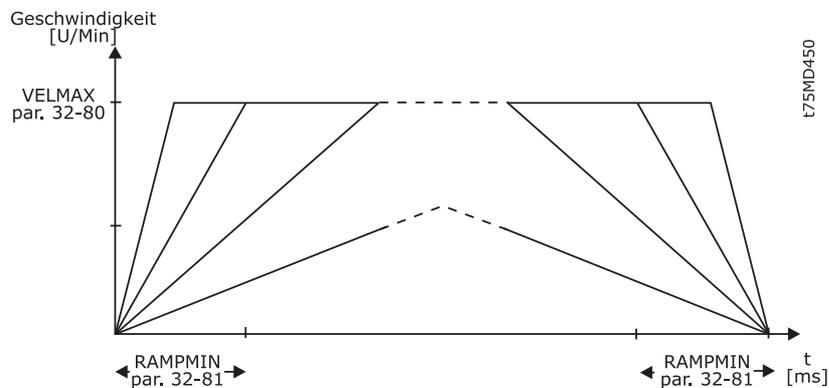
Befehlsgruppe REL, ABS

Querverweise DEC, VEL, POSA, POSR,
Parameter: 32-81 *Kürzeste Rampe*, 32-80 *Maximalgeschwindigkeit*, 32-83 *Geschwindigkeitsteiler*

Syntax-Beispiel ACC 10 /* Beschleunigung 10 */

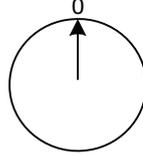
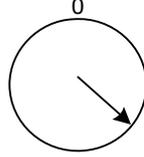
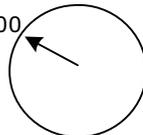
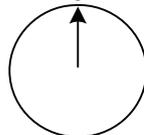
Beispiel

Minimale Beschleunigungszeit:	1000 ms
Maximale Geschwindigkeit:	1500 U/Min (25 U/s)
Geschwindigkeitsteiler:	100



Programmbeispiel ACC_01.M

□ APOS

Kurzinfo	Aktuelle Position einer Achse abfragen
Syntax	erg = APOS
Rückgabewert	erg = Istposition in Benutzereinheiten (BE) absolut zum aktuellen Nullpunkt Wegangaben in Fahrbefehlen erfolgen immer in Benutzereinheiten und werden intern in Quadcounts umgerechnet. (Siehe auch Benutzerfaktor Zähler und Nenner in Parameter 32-12 und 32-11.) Die Benutzereinheit (BE) entspricht in der Standardeinstellung der Anzahl Quadcounts: $\text{Parameter} = \frac{\text{Par. 32-12 Benutzereinheit Zähler}}{\text{Par. 32-11 Benutzereinheit Nenner}} = 1$
Beschreibung	Der Befehl APOS kann die Position der Achse absolut zum aktuellen Nullpunkt abfragen. ACHTUNG! Wenn ein mit SETORIGIN gesetzter und aktiver Temporärnullpunkt existiert, bezieht sich der Positionswert auf diesen Nullpunkt. ACHTUNG! Das Ergebnis muss nicht der Ziel- oder Sollposition entsprechen, wenn man mit APOS die Position abfragt. Es können sich Fehler oder Abweichungen durch die Mechanik und den abgerundeten Dezimalstellen in den Benutzereinheiten ergeben. APOS wird von den Parametern 32-12 und 32-11 sowie den Befehlen SETORIGIN p und DEFORIGIN beeinflusst. Beispiel: POSA 2000 PRINT "Istposition erreicht", APOS Ausgabe: Istposition erreicht 2000 (abhängig von den PID Einstellungen könnte eine kleine Abweichung auftreten) Beispiel mit SETORIGIN SETORIGIN 2000 POSA 2000 PRINT "Istposition", APOS Ausgabe: Istposition 2000
	<div style="display: flex; justify-content: space-around;"> <div style="text-align: center;"> <p>Initial</p>  </div> <div style="text-align: center;"> <p>Nach der Positionierung</p>  </div> </div> <div style="display: flex; justify-content: space-around; margin-top: 20px;"> <div style="text-align: center;"> <p>Nach der Ausführung</p>  </div> <div style="text-align: center;"> <p>Initial</p>  </div> </div>
	Beim Programmstart wird in diesem Beispiel die absolute Position 2000 qc als Startposition festgelegt; dann wird der Antrieb um 2000 qc entsprechend dem Positionierbefehl weiter gefahren.
Befehlsgruppe	SYS
Querverweise	CPOS, DEFORIGIN, SETORIGIN, POSA, POSR Parameter: 32-12 <i>Benutzerfaktor Zähler</i> , 32-11 <i>Benutzerfaktor Nenner</i>
Syntax-Beispiel	PRINT APOS /* Istposition der Achse am PC ausgeben */
Programmbeispiel	APOS_01.M, GOSUB_01.M, MOTOR_01.M

__ Befehlsreferenz __

□ APOSDIFF

Kurzinfo	Overflow-Handling von Inkrementalgebern in Anwendungen.
Syntax	erg = APOSDIFF oldpos
Parameter	n = Achsnummer oldpos = APOS zu einem früheren Zeitpunkt
Rückgabewert	Liefert die Differenz zwischen APOS und oldpos (erg = APOS – oldpos) in BE.
Beschreibung	Dieser Befehl vereinfacht die Behandlung des Überlaufs von Inkrementalgebern in Anwendungen. Wenn zum Beispiel der Anwender eine aktuelle Position in seinem Programm speichert und später die Differenz berechnen will, muss er normalerweise den Überlauf der Position berücksichtigen. Statt dessen kann dieser Befehl benutzt werden; siehe unten. Intern prüft diese Routine, ob die Differenz größer als POS_LIMIT (0x3FFFFFFF) ist. Falls ja, wird angenommen, dass ein Überlauf stattfand und es wird korrekt gehandhabt.
	ACHTUNG! Dies löst aber nicht das Überlaufproblem wenn in der Anwendung Benutzereinheiten BE benutzt werden.
Portabilität	Standardbefehl ab MCO 5.00
Befehlsgruppe	SYS
Querverweise	APOS
Syntax-Beispiel	oldpos = APOS diff = APOSDIFF oldpos // liefert die Differenz zwischen APOS und oldpos in BE // zur Handhabung von Overflow, falls notwendig (diff = APOS – oldpos)

□ AVEL

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

__ Befehlsreferenz __

□ CANDEL

Kurzinfo	Löscht alle oder einzelne CAN-Objekte
Syntax	CANDEL objnr
Parameter	objnr = Objektnummer, die beim Definition des Objektes zurück gegeben wurde = -1 löscht alle Objekte (außer den Standardobjekten)
Beschreibung	Mit dem Befehl CANDEL können CAN-Objekte wieder gelöscht werden, die zuvor mit DEFCANIN oder DEFCANOUT angelegt wurden.
	 <p>ACHTUNG! Standardobjekte für die gepufferte Ein-/Ausgabe (OUTMSG oder INMSG) können damit nicht gelöscht werden. Diese können nur beim Initialisieren nicht angelegt werden.</p>
Portabilität	Optionaler Befehl; Funktion „Einzelnes Objekt löschen“ ab MCO 5.00.
Befehlsgruppe	CAN
Syntax-Beispiel	CANDEL -1 /* es werden alle CAN-Objekte gelöscht */

□ CANIN

Kurzinfo	Liest ein Objekt über den CAN-Bus
Syntax	status = CANIN objnr timeout control varhi varlo
Parameter	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 <i>timeout</i> [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 <i>timeout</i> auf die Daten gewartet varhi Byte 0 bis 3 der CAN-Objektdaten varlo Byte 4 bis 7 der CAN-Objektdaten
Rückgabewert	status = -1 keine Daten angekommen = 0 ok
Beschreibung	<p>Der Befehl CANIN kopiert die Daten (falls vorhanden) des CAN-Objektes <i>objnr</i> in die Variablen <i>varhi</i> und <i>varlo</i>. Wenn <i>control</i> = 1 ist, werden die Daten zuerst angefordert.</p> <p>Es ist möglich alle Transmit-PDOs eines digitalen Eingangs-Moduls oder des CAN-Drive-Status unter Verwendung eines einzigen CAN-Telegramms zu sammeln. Diese Funktion ist auf den <u>Master-Bus</u> beschränkt. Die Funktion muss das CAN-Objekt 15 benutzen, das intern die Anzahl der verwendbaren CAN-Objekte auf dem Master-Bus um 1 reduziert.</p> <p>Zum Aktivieren der Funktion muss der Befehl CANINI 999 benutzt werden. Dieser Befehl aktiviert den Empfang der TxPDOs durch Interrupt und speichert jedes erhaltene PDO in einen Puffer mit der Tiefe von 1. Dies funktioniert für alle IDs von 1 bis 127. Das heißt, wenn irgendein I/O-Modul in diesem Bereich ein TxPDO sendet, wird es erfasst und im Puffer gespeichert. Das nächste TxPDO des gleichen Moduls überschreibt natürlich das erste. Der Puffer kann mit folgendem Befehl ausgelesen werden:</p> <p style="text-align: center;">erg = CANIN (id * 100) timeout 0 hi lo</p>

__ Befehlsreferenz __

Dieser Befehl liefert -1 wenn keine neue Information im Puffer war, andernfalls liefert er 0. Timeout kann normal benutzt werden:

- 1 wartet nicht,
- 0 wartet auf neue Information (ohne Abbruch),
- n wartet n ms.

Das Ergebnis wird in hi und lo geliefert, aber es wird gleich in Bytes angeordnet, so dass diese benutzt werden können, falls sie mit PDO[] gelesen werden. Das heißt, wenn lo eine 32-Bit Zahl enthält, kann es sofort benutzt werden, ohne die Bytes neu zu ordnen.

Wann immer ein neuer CANINI Befehl benutzt oder ein neues Programm gestartet wird, ist diese Funktion deaktiviert.

Natürlich beansprucht diese Funktion den Prozessor weil sie starken PDO Traffic auf dem Bus verursacht.



Es ist nicht empfehlenswert diese Funktion gemeinsam mit anderen CAN-IO Befehlen auf dem gleichen Bus zu verwenden. Dies kann zu unerwarteten Ergebnissen führen. Wenn zum Beispiel ein digitales CANopen-I/O-Modul mit ID 3 auf dem Master-Bus mit einem IN (3*256+1) oder CANINI 3 999 benutzt wird, wird dies zu einer Situation führen, in der der Befehl IN zwar arbeitet, aber es werden nicht die PDOs mit den oben beschriebenen CANIN Befehl geliefert. Der Grund hierfür ist, dass zwei CAN-Objekte mit der gleichen ID existieren. In diesem Fall bedient der Prozessor nur die erste. Wie oben erwähnt, wird Objekt 15 für den Empfang von allen PDOs benutzt, in diesem Fall also das letzte.

Portabilität

Optionaler Befehl;
erweiterte Befehle CANINI und CANIN für den Einsatz eines einzigen CAN-Telegramms ab MCO 5.0.

Befehlsgruppe

CAN

Querverweise

CANOUT, CANDEL, DEFCANOUT, DEFCANIN

Syntax-Beispiel

```
MSG = 0
temp = 0                /* Variablen definieren */
rx1 = DEFCANIN 42 8    /* es wird ein RX-Objekt angelegt */
STAT = CANIN rx1 1000 0 MSG temp /* 1s auf Daten warten */
```

Programmbeispiel

CAN_sample.M, CANIN.M



□ CANINI

Kurzinfo	Initialisiert die notwendigen Objekte (PDOs) für den Datenaustausch mit CANopen-Teilnehmern oder aktiviert die erweiterte CANINI, CANIN Funktion.
Syntax	<p>CANINI nr, nr, nr, ..., CANINI 999 CANINI nr, nr, 999, nr</p> <p>ist auch möglich, jedoch nur im gleichen Befehl. Der nächste neue Befehle würde sonst die vorhergehenden Parameter löschen.</p>
Parameter	<p>nr = guard * (busoffset * 1000 + id) guard = -1, +1 (ohne / mit Guarding) busoffset = 100000, 0 (Slave-Bus, Master-Bus)</p> <p>Beispiele für CANINI Werte:</p> <ul style="list-style-type: none"> 0 Löscht alle zuvor mit CANINI definierten Objekte 999 Aktiviert den Empfang von allen TPDO1s (0x181-0x1FF) durch Interrupt und speichert jeden erhaltenen PDO in einen Puffer mit der Tiefe 1. 1...127 CAN I/O mit Bus-ID 1...127 mit Guarding -1...-127 CAN I/O mit Bus-ID 1...127 ohne Guarding
Beschreibung	<p>Der Befehl CANINI nimmt Kontakt zu den CAN-Modulen auf und legt entsprechende permanente CAN-Objekte an, um mit diesen Modulen kommunizieren zu können (PDO). Das hat den Vorteil, dass diese Eingangsmodule auch für Interrupt-Funktionen verwendet werden können ohne den Bus permanent zu belasten.</p> <p>Wenn Sie keine Interrupts benötigen, können Sie mit CANINI die Abarbeitung der Befehle IN und INB beschleunigen, da die entsprechenden Module dann bei jeder Zustandsänderung eines Eingangs diese Information selbstständig schicken.</p> <p>Es wird dringend empfohlen Eingangsmodule die mit CANINI initialisiert wurden auch mit Guarding zu überwachen (d.h. CANINI > 0), um sicherzustellen, dass das Modul noch präsent ist und an der Bus-Kommunikation teilnimmt. Falls ein Teilnehmer nicht mehr vorhanden ist, wird durch die fehlende Rückantwort auf das GUARD-Objekt ein Fehler 188 ausgelöst. Auf diesen kann dann in der mit ON ERROR definierten Fehlerbehandlung reagiert werden.</p> <p>Wenn CANINI für Antriebe ausgeführt wird, wird ein entsprechendes PDO erzeugt und – falls notwendig – auch das SYNC Objekt. Wenn die Überwachung gestartet wird, wird alle 20 ms ein Guarding-Telegramm zu einem Modul gesendet. Sind zum Beispiel 4 Module vorhanden, dauert es 80 ms, um jedes Modul zu überprüfen. Wenn innerhalb von 100 ms keine Antwort kommt, wird der Fehler 188 (guarding error) angezeigt.</p> <p>Es können maximal 16 Module intern gespeichert werden.</p> <p>Jeder erneute CANINI Befehl initialisiert alle, früher mit CANINI angelegten Objekte, d.h. die Überwachung (GUARD) und auch die SYNC Telegramme werden gestoppt. Allerdings nicht, wenn es permanente Objekte von einem Parameter 32-00 oder 32-30 <i>Inkrementalgeber Signaltyp</i> gibt. Diese bleiben ebenso erhalten, wie die intern automatisch angelegten PDOs gemäß der Definition in den Parametern 32-00 und 32-30.</p> <p>Der CANINI Befehl startet gleichzeitig die Module, d.h. für jedes Modul wird eine NMT0 Meldung gesendet, um das Modul auf den Status „betriebsbereit“ zu setzen. Wenn CANINI fehlschlägt kann die SYSVAR 67 [0x01220244] GuardErrorId ausgelesen werden, um herauszufinden welche ID das Problem verursacht hat.</p>

__ Befehlsreferenz __

Portabilität	Befehl ab MCO 5.00 verfügbar.
	ACHTUNG! Falls der CANINI Befehl auf Steuerungen mit mehreren, getrennten CAN-Bus-Kreisen eingesetzt wird, wird die GUARD- und SYNC-Funktionalität nur auf dem so genannten Slave-Bus unterstützt.
Befehlsgruppe	SYS
Querverweise	IN, INAD, INB, OUT, OUTB, OUTDA, Par. 32-00 oder 32-30 <i>Inkrementalgeber Signaltyp</i>
Syntax-Beispiel	CANINI 1,2,3,4 /* Initialisieren der CAN-Module mit voreingestellter Teilnehmernummer */
Programmbeispiel	CAN_sample.M

□ CANOUT

Kurzinfo	Nachricht mit interner Nummer abschicken
Syntax	CANOUT nr werthi wertlo
Parameter	werthiByte 0 bis 3 der CAN-Objektdaten wertloByte 4 bis 7 der CAN-Objektdaten
Beschreibung	Mit diesem Befehl wird eine CAN-Nachricht von einem durch DEFCANOUT definierten Sendeobjekt abgeschickt. Die Werte <i>hi</i> und <i>lo</i> sind 4 Bytes lang.
Portabilität	optionaler Befehl
Befehlsgruppe	CAN
Querverweise	CANIN, CANDEL, DEFCANIN, DEFCANOUT
Syntax-Beispiel	wertlhi = 21 wertlo = 41 tx1 = DEFCANOUT 500 8 /* ein TX-Objekt mit Id=500 und Datenlänge = 8 Bytes wird definiert */ CANOUT tx1 valhi vallo /* eine CAN-Nachricht wird gesendet */

□ COMOPTGET

Kurzinfo	Liest ein Telegramm der Kommunikationsoption.													
Syntax	COMOPTGET anz array													
Parameter	array = der Name eines Arrays, das mindestens die Größe 'anz' haben muss anz = Anzahl Datenworte, die gelesen werden sollen													
Beschreibung	COMOPTGET liest aus dem Puffer der Kommunikationsoption 'anz' Datenworte aus und schreibt sie in das Array 'array' beim ersten Element beginnend.													
Kompatibilität	Mit eingebauter Kommunikationsoption.													
Kommunikationsoption	Funktion der Kommunikationsoption: Parameter: Die Lese- und Schreibparameter werden von der Optionskarte nicht verändert.													
	ACHTUNG!: Die Parameter 9-15 und 9-16 müssen zusätzlich mit den richtigen Werten gesetzt werden.													
Kontrolldaten	Die Funktion des Steuerwortes (STW) und der Hauptsollwertes (HSW) hängt davon ab, wie der Par. 33-82 <i>Statusüberwachung Antrieb gesetzt ist</i> ; das Zustandswort (ZSW) und der Hauptwert (HIW) sind immer aktiv.													
		<table border="1"> <thead> <tr> <th></th> <th>Parameter 33-82</th> <th>Parameter 33-82</th> </tr> </thead> <tbody> <tr> <td></td> <td>„MCO 305 EIN“</td> <td>„MCO 305 AUS“</td> </tr> <tr> <td>STW/HSW</td> <td>nicht aktiv</td> <td>aktiv</td> </tr> <tr> <td>ZSW/HIW</td> <td>aktiv</td> <td>aktiv</td> </tr> </tbody> </table>		Parameter 33-82	Parameter 33-82		„MCO 305 EIN“	„MCO 305 AUS“	STW/HSW	nicht aktiv	aktiv	ZSW/HIW	aktiv	aktiv
	Parameter 33-82	Parameter 33-82												
	„MCO 305 EIN“	„MCO 305 AUS“												
STW/HSW	nicht aktiv	aktiv												
ZSW/HIW	aktiv	aktiv												
Prozessdaten	PCD's 1 – 4 von PPO Typ 2/ 4 und PCD's 1 – 8 von PPO Typ 5 sind nicht mit einer Parameternummer 9-15 und 9-16 festgelegt, sondern können frei in einem APOSS-Programm benutzt werden. Der Befehl COMOPTGET kopiert die empfangenen Daten der Kommunikationsoption in ein Array, in dem jedes Array-Element ein Datenwort (16 Bit) enthält. Der Befehl COMOPTSEND kopiert die Daten von einem Array, in dem jedes Array Element ein Datenwort (16 Bit) enthält, in einen Sendepuffer der Kommunikationsoption, von dem es via Netzwerk zum Master gesendet wird.													
Befehlsgruppe	Kommunikationsoption													
Querverweis	COMOPTSEND													
Programmbeispiel	COM_OPT													

□ COMOPTSEND

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

□ CONTINUE

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

□ CPOS

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

__ Befehlsreferenz __

□ CPOSDIFF

Kurzinfo	Overflow-Handling von Inkrementalgebern in Anwendungen.
Syntax	erg = CPOSDIFF oldpos
Parameter	oldpos = CPOS zu einem früheren Zeitpunkt
Rückgabewert	Liefert die Differenz zwischen CPOS und oldpos (erg = CPOS – oldpos) in BE.
Beschreibung	<p>Dieser Befehl vereinfacht die Behandlung des Überlaufs von Inkrementalgebern in Anwendungen. Wenn zum Beispiel der Anwender eine aktuelle Position in seinem Programm speichert und später die Differenz berechnen will, muss er normalerweise den Überlauf der Position berücksichtigen. Statt dessen kann dieser Befehl benutzt werden; siehe unten.</p> <p>Intern prüft diese Routine, ob die Differenz größer als POS_LIMIT (0x3FFFFFFF) ist. Falls dies der Fall ist, wird angenommen, dass ein Überlauf stattfand und korrekt behandelt.</p> <p>ACHTUNG!: Dies löst aber nicht das Problem des Überlaufs, wenn in der Anwendung Benutzer-einheit BE verwendet werden.</p>
Portabilität	Standardbefehl ab MCO 5.00
Befehlsgruppe	SYS
Querverweise	CPOS
Syntax-Beispiel	<pre>oldpos = CPOS diff = CPOSDIFF oldpos // liefert die Differenz zwischen CPOS und oldpos in BE // Behandlung eines Overflows falls notwendig (diff = CPOS – oldpos)</pre>

□ CSTART

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

□ CSTOP

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



□ CURVEPOS

Kurzinfo Der aktuellen Master-Position entsprechende Slave-Kurvenposition abfragen.

Syntax erg = CURVEPOS

Rückgabewert erg = Slave-Position in CAM-Einheiten (BE) absolut zum aktuellen Nullpunkt.

Beschreibung Mit dem Befehl CURVEPOS kann die Slave-Kurvenposition, die der aktuellen Master-Position entspricht, abgefragt werden.

Die Position kann unabhängig vom Betriebszustand (Lageregelung im Stillstand, Positioniervorgang, Drehzahlregelung oder Synchronisation) abgefragt werden.

CMASTERCPOS (SYSVAR) und CURVEPOS werden aktualisiert, auch wenn SYNCC nicht weiter aktiv ist. Die Aktualisierung dieser Werte beginnt nach einem SETCURVE Befehl (wenn Par. 33-23 *Startverhalten für Sync* ist < 2000) oder nach SYNCC und den ersten Master-Marker (wenn Par. 33-23 = 2000).

Die Aktualisierung wird also auch nach dem Stoppen des SYNCC Befehls fortgeführt, wenn Par. 33-23 *Startverhalten für Sync* < 2000.

ACHTUNG!:

Die Position ist nur definiert, wenn zuvor ein SETCURVE gesetzt wurde.

ACHTUNG!:

Wenn ein mit SETORIGIN gesetzter und aktiver Temporärnullpunkt existiert, bezieht sich der Positionswert auf diesen Nullpunkt.

ACHTUNG!:

DEFMCPOS und DEFMORIGIN können diese Position noch verändern.

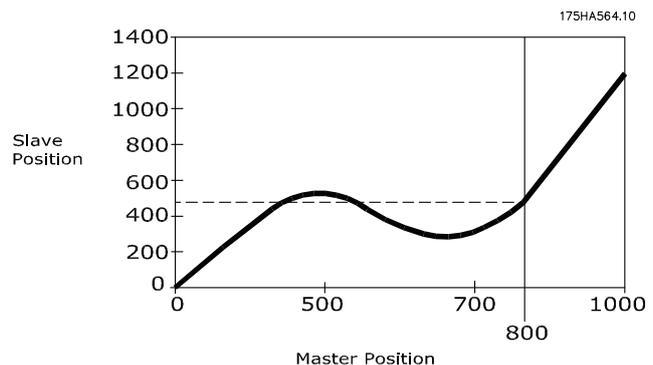
Befehlsgruppe CAM

Querverweise APOS, DEFORIGIN, SETORIGIN, POSA, POSR, DEFMCPOS,
Parameter: 33-10 *Sync-Faktor Master*, 33-11 *Sync-Faktor Slave*

Syntax-Beispiel PRINT CURVEPOS // aktuelle Slave-Position der Kurve ausgeben

Beispiel Fixpunkte einer Kurve:

Master	Slave
0	0
500	500
700	300
1000	1200



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

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

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

Also ist CURVEPOS unabhängig von der Slave-Position.

□ CVEL

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



□ DEC

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

□ DEFCANIN

Kurzinfo	Definiert ein Empfangsobjekt
Syntax	<code>objnr = DEFCANIN id dlen</code>
Parameter	<p><code>id = CAN-Identifikationsnummer</code></p> <p><code>dlen = Datenlänge des Objektes in Bytes (max. 8 Bytes)</code></p>
Rückgabewert	<p><code>objnr</code></p> <p>Ein positiver Wert bedeutet, dass ein Objekt erfolgreich angelegt wurde. Dieser Wert ist eine interne Nummer des Objektes und wird von anderen APOSS-CAN-Befehlen benutzt. Ein negativer Wert meldet einen Fehler.</p>
Beschreibung	<p>Mit diesem Befehl wird im CAN-Controller ein Empfangs-Kommunikations-Objekt definiert.</p> <p>Dieser Befehl kann auch mit dem Offset für den Slave-Bus genutzt werden (Telegramm ID 100000).</p> <p>Diese Objekte können nun einzeln mit dem Befehl CANDEL <code>objnr</code> gelöscht werden, wobei <i>objnr</i> die Nummer ist, die von DEFCANIN oder DEFCANOUT zurückgegeben werden.</p>
Portabilität	Befehl ist ab MCO 5.00 verfügbar.
Befehlsgruppe	CAN
Syntax-Beispiel	<pre>var1 = 0 /* Variablen deklarieren */ var2 = 0 rx1= DEFCANIN 500 8 /* es wird ein RX-Objekt mit Id=500 und Datenlänge = 8 Bytes definiert */ CANIN rx1 0 0 var1 var /* es wird eine CAN-Nachricht gelesen */</pre>



□ DEFCANOUT

Kurzinfo	Definiert ein Sendeobjekt im CAN-Controller
Syntax	<code>objnr = DEFCANOUT id dlen</code>
Parameter	<p><code>id</code> = CAN-Identifikationsnummer</p> <p><code>dlen</code> = Datenlänge des Objektes in Bytes (max. 8 Bytes)</p>
Rückgabewert	<p><code>objnr</code></p> <p>Ein positiver Wert bedeutet, dass ein Objekt erfolgreich angelegt wurde. Dieser Wert ist eine interne Nummer des Objektes und wird von anderen APOSS-CAN-Befehlen benutzt. Ein negativer Wert meldet einen Fehler.</p>
Beschreibung	<p>Mit diesem Befehl wird im CAN-Controller ein Objekt definiert. Das Objekt ist ein Sendeobjekt mit Länge in n-Bytes und der CAN-Identifizierung (<code>id</code>). <code>Objnr</code> ist dann eine interne Nummer des Objektes <code>Id</code> und wird vom Befehl CANOUT benutzt.</p> <p>Dieser Befehl kann mit dem Offset für Slave-Bus auch genutzt werden (Telegramm ID 100000).</p> <p>Diese Objekte können mit dem Befehl CANDEL <code>objnr</code> nun einzeln gelöscht werden, wobei <code>objnr</code> die Nummer ist, den von DEFCANIN oder DEFCANOUT zurückgegeben wird.</p>
Portabilität	Der Befehl ist ab MCO 5.00 verfügbar.
Befehlsgruppe	CAN
Querverweise	DEFCANIN, CANOUT, CANIN
Syntax-Beispiel	<pre>nr = DEFCANOUT 500 8 // wird ein Objekt mit id =500 und Länge 8 Bytes definiert, // die Funktion liefert eine 1 zurück, // eine Nachricht mit id =500 und langen 8 Bytes kann jetzt mit // dem Befehl CANOUT 1 wert1 wert2 abgeschickt werden */ nr = DEFCANOUT id len</pre>

□ DEFCORIGIN

Kurzinfo	Sollposition als Nullpunkt setzen
Syntax	DEFCORIGIN
Beschreibung	<p>Mit dem DEFCORIGIN Befehl wird die Sollposition als Nullpunkt gesetzt. Alle absoluten Positionierbefehle (POSA etc.) beziehen sich fortan auf diesen Nullpunkt.</p> <p>Auf diese Weise wird CPOS auf Null gesetzt und APOS so, dass die Differenz erhalten bleibt.</p>
Portabilität	Der Befehl ist ab MCO 5.00 verfügbar.
Befehlsgruppe	INI
Querverweise	POSA, DEFCORIGIN, CPOS
Syntax-Beispiel	<pre>POSA 80000 /* Achse absolut positionieren */ DEFCORIGIN /* Sollposition als Nullpunkt definieren */</pre>

__ Befehlsreferenz __

□ DEFMCPPOS

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

MAPOS(qc)

0

1000

2000

3000

4000

5000

▶ Position

Master
Curve
position

500

1000

1500

2000

500

1000



Master's physical position

175HA560.10

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

MAPOS(qc)

0

1000

2000

3000

4000

5000

▶ Position

Master
Curve
position

500

1000

1500

2000

500



Master's physical position

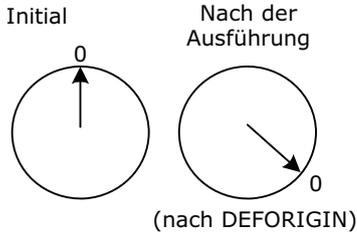
175HA561.10

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

□ DEFMORIGIN

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

□ DEFORIGIN

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

□ DEFSYNCORIGIN

Kurzinfo	Definiert das Verhältnis Master:Slave für den nächsten SYNCP oder SYNCM Befehl oder die Startwerte für Standardkurven mit SYNCC Befehl.	
Syntax	DEFSYNCORIGIN mcpos spos	
Parameter	mcpos	= Master Sollposition in qc, oder Master Kurvenposition in qc
	spos	= Slave Sollposition oder
	spos < SlaveCurveLenght	= Slave Kurvenposition
	spos > SlaveCurveLenght	= die aktuelle Kurvenposition wird als korrekte Position innerhalb der Kurve entsprechend zur Masterposition <i>mcpos</i> betrachtet
Beschreibung	<p>Dieser Befehl definiert, wie viel Abstand vor oder nach dem Slave im Verhältnis zur Masterposition sein soll. Damit kann das Verhältnis zwischen Master und Slave für den nächsten SYNCP oder SYNCM Befehl definiert werden. Er setzt die interne Slave-Sollposition auf den Wert des Slaves.</p> <p>Der Wert des Masters wird für einen internen MOVE SYNCORIGIN benutzt. Dafür wird ein MOVESYNCORIGIN durch diesen Befehl überschrieben. Beide Aktionen werden in dem Moment ausgeführt, wenn der SYNC Befehl aktiviert wird. Das garantiert, dass Master und Slave auf die o.g. Master-Slave-Position synchronisiert werden.</p> <p>Mit SYNCC kann der Befehl DEFSYNCORIGIN benutzt werden um die Startwerte für Standardkurven wie folgt zu definieren:</p>	

__ Befehlsreferenz __

Dabei meldet *mcpos* der Steuerung, dass die aktuelle Masterposition (MAPOS) einer Master-Kurvenposition von *mcpos* entspricht.

spos hat zwei verschiedene Bedeutungen:

spos < SlaveCurveLength = definiert die aktuelle Position des Slaves als die Slave-Kurvenposition *spos*.

spos > SlaveCurveLength = die aktuelle Kurvenposition wird als die korrekte Position innerhalb der Kurve betrachtet, die der Masterposition *mcpos* entspricht (*spos* selbst hat keine Bedeutung, sie muss nur größer als die Slave-Kurvenlänge sein).

Beispiel: Angenommen die Kurve ist eine gerade Linie, die von 0,0 nach 2000,4000 mittels (master,slave) geht.

Weiterhin sei angenommen, dass der Slave auf der Position 11000 qc ist und dass der Master eine aktuelle Position von 15000 qc hat. Um es einfach zu machen, sei angenommen, dass die Getriebefaktoren 1 : 1 sind.

Wenn nun ein SETCURVE ohne irgendwelche weiteren Befehle durchgeführt wird, passiert Folgendes: Die Steuerung versucht zu herauszufinden, wo sich der Slave innerhalb einer Kurve befindet. Daher berechnet sie den Rest von 11000 / 4000, welcher 3000 ist und den Rest von 15000 / 2000, welcher 1000 ist. Das heißt, das Ergebnis wäre eine aktuelle MCPOS (Master-Kurvenposition) von 1000. Daher sollte die entsprechende Slave-Kurvenposition 2000 sein. Tatsächlich ist der Slave aber in diesem Moment auf einer Kurvenposition von 3000. (Die csstart Position ist die Startposition der letzten Kurve, welche in unserem Fall 8000 sein würde.) Wenn diese Kurve gestartet wird, würde der Slave versuchen die Position zu korrigieren und auf 2000 fahren.

Wenn nun der Befehl

```
DEFSYNCORIGIN 500 100000
```

benutzt wird, passiert Folgendes:

Die aktuelle Masterposition ist als Master-Kurvenposition 500 definiert. Und da 100000 größer ist als die Slave-Kurvenlänge 4000, entspricht die aktuelle Slave-Position (von 11000 qc) der Slave-Kurvenposition 1000, die zur Master-Kurvenposition 500 gehört. Daher wird csstart auf 10000 gesetzt. Wenn nun die Kurve gestartet wird, wird sie korrekt berechnet und folgt der Kurve, sobald der Master die Fahrbewegung startet.

DEFSYNCORIGIN in Verbindung mit CU_GRAD Kurven (Typ 3)

Im Fall einer CU_GRAD Kurve, kann DEFSYNCORIGIN benutzt werden, um die absolute Endposition einer Kurve in qc zu definieren. In diesem Fall wird die Kurve sofort gestartet und in so berechnet, dass die Endpositionen am Ende der Kurve erreicht werden.



ACHTUNG!

Um degenerierte Polynome zu vermeiden, muss der Abstand in einer korrekten Relation stehen. Wenn mit der Geschwindigkeit Null gestartet wird, die in einer Geschwindigkeit 1 enden soll (Slave hat die gleiche Geschwindigkeit wie der Master), dann sollte die Masterdistanz geringer sein, als die zweifache Slave-Distanz. Andernfalls wird das Polynom innerhalb des Intervalls Schwankungen aufweisen. In anderen Fällen ist dies schwieriger vorzuberechnen (Startgeschwindigkeit nicht 0 oder Endgeschwindigkeit nicht 1). Deshalb kann man das PG_FLAG_CURVE_ERR prüfen, um zu sehen, ob die letzte SETCURVE eine Kurve mit Schwankungen produzierte (siehe STAT). Dann kann man die SYSVAR PFG_LASTERROR auslesen (siehe SDO Dictionary) um zu entscheiden, was die Ursache war.

__ Befehlsreferenz __

Portabilität Definition der Startwerte für Kurven ist ab MCO 5.00 möglich.

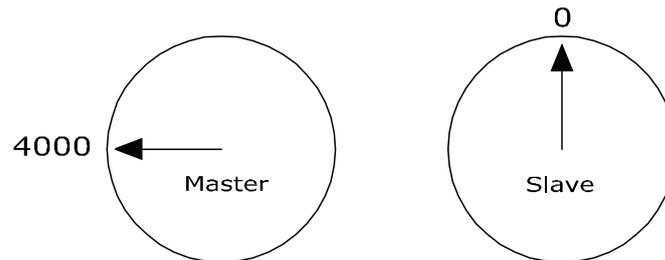
Befehlsgruppe SYN

Querverweise MOVESYNCORIGIN

Beispiel In diesem Beispiel soll der Slave auf Position 4000 qc sein, wenn der Master auf 2000 qc ist, d.h. der Slave sollte einen Vorsprung von 2000 qc auf den Master haben.

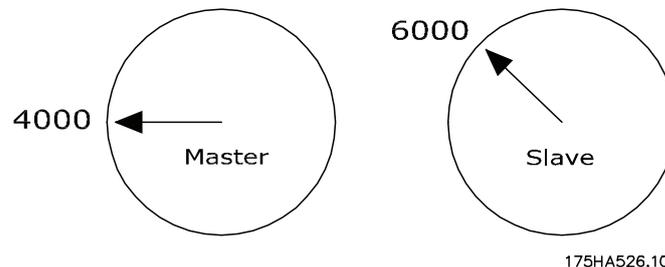
Ebenso soll der Slave auf Position 5000 qc sein, wenn der Master auf 3000 qc ist.

Ausgangsposition



Befehl; DEFSYNCORIGIN 2000 4000

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



175HA526.10

□ DELAY

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

□ DELETE ARRAYS

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

□ DIM

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

□ DISABLE .. interrupts

Kurzinfo	Sperrt die Ausführung von Interrupts.
Syntax	DISABLE inttyp
Parameter	inttyp = ALL CANMSG COMBIT INT KEYPRESSED PARAM PERIOD position interrupts: ON APOS, ON IPOS, ON MAPOS, ON MCPOS, ON MIPOS STATBIT TIME
	<p>ACHTUNG! Die Ausführung der Fehlerbehandlung (ON ERROR) kann mit DISABLE nicht gesperrt werden. Der Fehler-Interrupt hat höchste Priorität und unterbricht auch andere aktive Interrupts.</p>
Beschreibung	<p>DISABLE schaltet alle oder explizit genannte Interrupts – außer ON ERROR – ab. Wenn die Funktion DISABLE .. im Hauptprogramm verwendet wird, kann sie Interrupts der entsprechenden Art verhindern.</p> <p>Dies ist insbesondere nützlich, wenn eine Variable, die in einer Interrupt-Prozedur gesetzt ist, im Hauptprogramm verwendet wird. Dazu sollten Sie im Hauptprogramm zunächst die entsprechenden (oder alle) Interrupts mit DISABLE .. abschalten, die Variable ändern und anschließend die entsprechenden (oder alle) Interrupts mit ENABLE .. wieder einschalten.</p>
	<p>ACHTUNG! Wird ein Interrupt disabled (d.h. gesperrt) existiert er weiterhin, wird aber nicht mehr ausgeführt. (Ausnahme: DISABLE ALL).</p> <p>Die Erkennung läuft weiter im Hintergrund und die Interrupt-Anforderung wird im Fall eines nicht (!) flankengetriggerten oder nachrichtenorientierten Interrupts (ON PERIOD, ON APOS, ON PARAM, etc.) gespeichert. Wenn der Interrupt dann wieder enabled (d.h. freigegeben) wird und es zuvor einen noch nicht ausgeführten, gespeicherten (nicht flankengetriggerten) Interrupt gab, wird dieser Interrupt sofort ausgeführt.</p> <p>Im Fall eines flankengetriggerten Interrupts (z.B. ON INT, ON COMBIT, ON STATBIT), werden alle Interrupts, die während der DISABLE-Phase stattgefunden haben, nicht ausgeführt, auch dann nicht, wenn wieder auf ENABLE umgeschaltet wird. Diese Interrupts werden im Status DISABLE nicht gespeichert. Flankengetriggerte Interrupts, die nach dem erneuten ENABLE stattfinden, werden weiterhin wieder ausgeführt.</p>
	<p>ACHTUNG! Ausnahme: DISABLE ALL</p> <p>Während bei dem selektiven Sperren flankengetriggerten Interrupts (z.B. DISABLE INT) diese Interrupts, wie beschrieben, ignoriert und auch nach der Freigabe nicht mehr ausgeführt werden, wird bei DISABLE ALL die Anforderung (auch von flankengetriggerten Interrupts) gespeichert und der Interrupt nach der Freigabe (ENABLE ALL) noch ausgeführt!</p>



__ Befehlsreferenz __



DISABLE ALL in Kombination mit selektivem DISABLE

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



Interrupt-Behandlung im Interrupt

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

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

Befehlsgruppe INT

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

Syntax-Beispiel DISABLE ALL /* Alle Interrupts abschalten */
DISABLE STATBIT /* Interrupt für Statusbit abschalten */

□ ENABLE .. interrupts

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




ENCPOSOFFS

Kurzinfo	Synchronisiert den inkrementalen Positionszähler mit dem absoluten im Encoder.	
Syntax	erg = ENCPOSOFFS offset	
Parameter	offset = Liefert die Differenz zwischen der absoluten und inkrementalen Position (absolute – inkremental)	
Rückgabewerte	OK 0	Der Befehl war erfolgreich.
	TIMEOUT -1	Keine Antwort innerhalb von 300 ms erhalten.
	BADFRAME -2	Der erhaltene Frame ist nicht gültig.
	OVERFLOW -4	Mehr Bytes erhalten, als der Empfangspuffer aufnehmen kann.
Beschreibung	<p>Es wird die Differenz zwischen der absoluten Encoder-Position und dem inkrementalen Zähler bestimmt und zurückgeliefert.</p> <p>Hierfür wird der inkrementale Zähler im DSP exakt in dem Moment gelatcht, in dem auch der Hiperface-Encoder die absolute Position latcht und diese via RS485 liefert. Mit Hilfe dieser Differenz kann der Anwender z.B. die Position innerhalb APOSS mit SETORIGIN auf den absoluten Wert setzen.</p> <p>Wird der Hiperface-Encoder als Master-Signal statt eines Slave-Signals benutzt, verwenden Sie den Befehl MENCPOSOFFS (siehe Parameter 32-50).</p> <p><u>Motor Rückführung:</u></p> <p>Das Feedbacksignal des Motors wird durch das inkrementale Signal erzeugt. Häufig werden die Hiperface-Encoder mit einem PM-Motor eingesetzt. Für solche PM-Motoren muss man die absolute Position des Motorläufers wissen. Die Läuferposition relativ zur absoluten Encoder-Position muss einmal während der Inbetriebnahme des Systems (oder mehrmals; sie wird im Encoder gesichert) ermittelt werden. Der Offset wird dann in Par. 1-41 (Parameter der Steuerungskarte) gesichert.</p> <p>Nach dem Aus- und Wiedereinschalten muss das inkrementale Signal (das für die Motorrückführung benutzt wird) wieder auf die absolute Position synchronisiert werden (siehe Programmbeispiel).</p>	
Befehlsgruppe	SYS	
Querverweis	MENCPOSOFFS	
Programmbeispiel	<pre> MOTOR OFF SET ENCODERTYPE 0 // kein inkrementaler Encoder angeschlossen SET ENCODERABSTYPE 1 // Hiperface Encoder SET ENCODEABSRES 4096 // Hiperface Auflösung DELAY 1000 pos = 0 RSTORIGIN offset = 0 PRINT "apos vorher: ", apos retval = ENCPOSOFFS offset PRINT "encposoffs erhalten: ", retval, " offset ist: ", offset SETORIGIN -offset PRINT "apos nachher: ", apos WHILE(1) DO PRINT apos // inkrementale Position ausgeben DELAY 500 ENDWHILE </pre>	

□ ENCTGREAD

Kurzinfo	Liest ein RS485 Telegramm vom Encoder.															
Syntax	erg = ENCTGREAD array															
Parameter	array = Benutzer-Array, in das die erhaltene Datenmenge geschrieben werden soll.															
Rückgabewerte	<table> <tr> <td>OK</td> <td>x (>0)</td> <td>TG mit x Byte Anwenderdaten angekommen</td> </tr> <tr> <td>ACTIVE</td> <td>0</td> <td>Die Übertragung läuft noch.</td> </tr> <tr> <td>TIMEOUT</td> <td>-1</td> <td>Keine Antwort innerhalb von 300 ms erhalten.</td> </tr> <tr> <td>BADFRAME</td> <td>-2</td> <td>Der erhaltene Frame ist nicht gültig.</td> </tr> <tr> <td>OVERFLOW</td> <td>-4</td> <td>Mehr Bytes erhalten, als der Empfangspuffer aufnehmen kann.</td> </tr> </table>	OK	x (>0)	TG mit x Byte Anwenderdaten angekommen	ACTIVE	0	Die Übertragung läuft noch.	TIMEOUT	-1	Keine Antwort innerhalb von 300 ms erhalten.	BADFRAME	-2	Der erhaltene Frame ist nicht gültig.	OVERFLOW	-4	Mehr Bytes erhalten, als der Empfangspuffer aufnehmen kann.
OK	x (>0)	TG mit x Byte Anwenderdaten angekommen														
ACTIVE	0	Die Übertragung läuft noch.														
TIMEOUT	-1	Keine Antwort innerhalb von 300 ms erhalten.														
BADFRAME	-2	Der erhaltene Frame ist nicht gültig.														
OVERFLOW	-4	Mehr Bytes erhalten, als der Empfangspuffer aufnehmen kann.														
Beschreibung	<p>Nachdem ein Telegramm mit ENCTGWRITE gesendet wurde, kann die Antwort mit diesem Befehl gepollt werden. Der Rückgabewert wird ausgegeben, entweder wenn er schon angekommen ist oder wenn ein Timeout aufgetreten ist.</p> <p>Wird der Hiperface-Encoder als Master-Signal statt eines Slave-Signals benutzt, verwenden Sie den Befehl MENCTGREAD (siehe Parameter 32-50).</p>															
Befehlsgruppe	SYS															
Querverweise	MENCTGREAD, ENCTGWRITE															
Programmbeispiel	<pre>// Programmbeispiel zum Empfangen der absoluten Position DIM sendbuffer[20] #define HIPER_READ_POS 0x42 SET ENCODERTYPE 0 // kein inkrementaler Encoder angeschlossen SET ENCODERABSTYPE 1 // Hiperface Encoder SET ENCODEABSRES 4096 // Hiperface Auflösung DELAY 1000 pos = 0 WHILE(1) DO sendbuffer[1] = HIPER_READ_POS retval = ENCTGWRITE 1 sendbuffer // Telegramm senden DELAY 1000 retval = ENCTGREAD sendbuffer // Antwort erhalten // prüfen, ob die korrekte Anzahl von Bytes erhalten wurde IF(retval == 7) then // 0x40 0x42 pos0 pos1 pos2 pos3 crc pos.b4 = sendbuffer[3] pos.b3 = sendbuffer[4] pos.b2 = sendbuffer[5] pos.b1 = sendbuffer[6] PRINT "Pos = ", pos ELSE PRINT "----- Übertragungsfehler -----: ", retval PRINT "1: ", sendbuffer[1] PRINT "2: ", sendbuffer[2] PRINT "3: ", sendbuffer[3] PRINT "4: ", sendbuffer[4] EXIT ENDIF DELAY 500 ENDWHILE</pre>															



__ Befehlsreferenz __

□ ENCTGWRITE

Summary	Sendet ein RS485 Telegramm zum Encoder.
Syntax	erg = ENCTGWRITE länge array
Parameter	länge = Anzahl der zu sendenden Bytes (im Benutzer-Array). array = Benutzer-Array, das die zum Encoder zu sendende Datenmenge enthält.
Rückgabewerte	OK 0 Telegramm wurde gesendet BUSY -3 Es läuft noch eine andere Übertragung; bis jetzt noch kein Timeout.
Beschreibung	Dieser Befehl sendet ein RS485 Telegramm zum Encoder mit der ID „ENCODERID“. Der Anwender muss zuvor die Datenmenge in das Array eintragen. Der Befehl füllt dann diese Daten in einen normalen RS485 Datenübertragungsblock und fügt einen CRC Wert hinzu. Der Befehl wartet nicht bis die Daten gesendet wurden oder eine Antwort erhalten wurde; der Rückgabewert kommt sofort. Das Antworttelegramm muss mit ENCTGREAD gepollt werden. Wird der Hiperface-Encoder als Master-Signal statt eines Slave-Signals benutzt, verwenden Sie den Befehl MENCTGWRITE (siehe Parameter 32-50).
Befehlsgruppe	SYS
Querverweise	ENCTGREAD, MENCTGWRITE
Programmbeispiel	See program sample ENCTGREAD command.

□ ERRCLR

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

□ ERRNO

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

□ EXIT

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

□ GET

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

□ GETVLT

Kurzinfo	Liest einen VLT-Parameter.
Syntax	erg = GETVLT par
Parameter	par = Parameternummer
Rückgabewert	erg = Parameterwert
Beschreibung	<p>GETVLT liest einen VLT-Parameter und liefert den entsprechenden Wert zurück. Mit GETVLT haben Sie somit Zugriff auf Betriebsdaten (z.B. Motorstrom 1-24) oder auf Konfigurationen (z.B. max. Sollwert Par. 3-03) des FC 300.</p> <p>Da ausschließlich Ganzzahlenwerte übertragen werden, muss bei der Auswertung des Rückgabewertes der Umwandlungsindex beachtet werden. So ist ein LCP Wert von 50,0 Hz (Par. 16-13 Umwandlungsindex = -1) gleichbedeutend mit einem Rückgabewert von 500.</p> <p>Die Liste der FC 300 Parameter mit dem zugehörigen Umwandlungsindex finden Sie im FC 300 Produkthandbuch.</p>
	<p>ACHTUNG! Benutzen Sie GETVLTSUB um Parameter mit Indexnummern zu lesen, z.B. den FC 300 Parameter 5-40.</p>
Befehlsgruppe	PAR
Querverweise	SETVLT
Syntax-Beispiel	PRINT GETVLT 413 /* Lese Par. 4-13 Motordrehzahl-Obergrenze */

□ GETVLTSUB

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

□ GOSUB

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

□ GOTO

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

□ HOME

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

□ IF ..THEN .., ELSEIF .. THEN .. ELSE .. ENDIF

Kurzinfo	Bedingte ein- oder mehrfache Programmverzweigung; (wenn Bedingung erfüllt, dann führe aus ..., sonst ...)
Syntax	IF Bedingung THEN Befehl ELSEIF Bedingung THEN Befehl ELSE Befehl ENDIF
Parameter	Bedingung = Verzweigungskriterium Befehl = ein oder mehrere Programmbefehle
Beschreibung	<p>Mit der IF..ENDIF Konstruktion können bedingte Programmverzweigungen realisiert werden.</p> <p>Ist die hinter IF bzw. ELSEIF stehende Bedingung erfüllt, werden die Befehle bis zur nächsten ELSEIF, ELSE oder ENDIF Anweisung ausgeführt und dann mit den nach der ENDIF Anweisung stehenden Befehlen das Programm fortgesetzt.</p> <p>Ist die Bedingung nicht erfüllt, werden die nachfolgenden ELSEIF Verzweigungen überprüft und es wird, sofern die Bedingung erfüllt ist, der entsprechende Programmteil ausgeführt und das Programm nach ENDIF fortgesetzt.</p> <p>Die Verzweigungsbedingung, die nach einer IF oder ELSEIF Anweisung überprüft wird, kann sich aus einer oder mehreren Vergleichsoperationen zusammensetzen.</p> <p>Innerhalb der IF..ENDIF Konstruktion können beliebig viele ELSEIF Verzweigungen auftreten, es darf jedoch nur eine ELSE Anweisung vorhanden sein. Hinter der ELSE Anweisung steht der Programmteil, der abgearbeitet wird, sofern keine der Bedingungen erfüllt wurde.</p> <p>Die Anweisungen ELSEIF und ELSE können, müssen aber nicht innerhalb einer IF ... ENDIF Konstruktion enthalten sein.</p>
	<p>ACHTUNG!:</p> <p>Nachdem eine Bedingung erfüllt wurde, wird der zugehörige Programmteil ausgeführt und das Programm nach der ENDIF Anweisung fortgesetzt. Weitere Bedingungen werden nicht mehr überprüft.</p>
Befehlsgruppe	CON
Querverweise	REPEAT .. UNTIL, WHILE .. ENDWHILE
Syntax-Beispiel	<pre> /**/ Einfachverzweigung /**/ IF (a == 1) THEN /* Variable a = 1, dann */ Befehlszeile 1 Befehlszeile n ENDIF /**/ Mehrfachverzweigung /**/ IF (a == 1 AND b != 1) THEN Befehlszeilen ELSEIF (a == 2 AND b != 1) THEN Befehlszeilen ELSEIF (a == 3) THEN Befehlszeilen ELSE Befehlszeilen ENDIF </pre>
Programmbeispiel	IF_01.M, ERROR_01.M, EXIT_01.M, HOME_01.M, IN_01.M, ...

□ IN

Kurzinfo	Zustand eines digitalen Eingangs abfragen.
Syntax	erg = IN n
Parameter	n = Nummer des Eingangs 1 – 10 oder 1 – 12 (Optionale Eingänge) 18, 19, 27, 29, 32, 33 bzw. für CANopen I/O-Module: CAN-Bus + (Modul-CAN-ID * 256) + Eingangsnummer (bzw. Eingangsbyte)
Rückgabewert	erg = Zustand des Eingangs 0 = Low-Pegel oder undefiniert 1 = High-Pegel
Beschreibung	<p>Mit dem IN Befehl können Sie den Zustand eines digitalen Eingangs abfragen. Es wird abhängig vom anliegenden Signalpegel eine 0 oder 1 zurückgeliefert.</p> <p>Der Modus Eingang 11,12 wird in Par. 33-60 IOMODE ausgewählt.</p> <p>Die Definition des High- und Low-Pegels sowie die Eingangsbeschaltung sind in den Produkthandbüchern MCO 305 und FC 300 beschrieben.</p> <p>Die Eingänge 5 und 6 werden auch als Marker-Eingänge für die Master- und Slave-Drehgeber benutzt.</p> <p>CAN-Module, die die CANopen Spezifikationen erfüllen, können Sie ebenfalls mit dem IN Befehl ansprechen, und zwar über die entsprechende Nummer, die wie folgt definiert ist:</p> <p style="padding-left: 40px;">CAN-Bus + (Modul-CAN-ID * 256) + Eingangsnummer (bzw. Eingangsbyte)</p> <p>Beim Ausführen eines solchen Befehls werden temporär die entsprechenden CAN-Objekte angelegt, ausgewertet und anschließend wieder freigegeben. Daher können Sie beliebig viele Module ansprechen, aber es sind zunächst keine Objekte da, die von sich aus empfangsbereit sind, zum Beispiel für Interrupt-Funktionen. Um Interrupt-Funktionen durchzuführen, müssen Sie die entsprechenden Module vorher mit CANINI initialisieren.</p>
Portabilität	Die Parameter für CANopen sind ab MCO 5.00 verfügbar.
Befehlsgruppe	I/O
Querverweise	INB, OUT, OUTB, CANINI Parameter: 33-60 <i>Klemme X59/1 und X59/2 Modus</i> , IOMODE, 33-50...59,61,62 <i>Klemme X57/n Digitale Eingänge</i> , I_FUNCTION_n
Syntax-Beispiel	<pre>in4 = IN 4 /* Zustand Eingang 4 in Variable ein4 speichern */ IF (IN 2) THEN /* Bei High-Pegel an Klemme 2, Ausgang 1 setzen */ OUT 1 1 ELSE OUT 1 0 ENDIF</pre>
Programmbeispiel	IN_01.M



□ INAD

Kurzinfo	Analogen Eingang lesen oder Prozess Daten Objekte (PDO) eines CAN Objekts.
Syntax	erg = INAD n
Parameter	n = a) Nummer des Analogeingangs: 53,54 b) für CAN-Anwendungen: Modulnummer * 256 + Nummer des Eingangs
Rückgabewert	erg = Analogwert a) Klemme 53/54: -1000 – 1000 = -10 V – 10 V Klemme 53/54: 0 – 10 V erg = 0 – 100 b) Der Wertebereich hängt vom eingesetzten Analogeingang ab.
Beschreibung	Der INAD Befehl liest den Analogwert des entsprechenden Eingangs. CAN-Module, die die CANopen Spezifikationen erfüllen, können Sie mit dem INAD Befehl ansprechen, und zwar über die entsprechende Modulnummer * 256 + I/O-Nummer. Beim Ausführen eines solchen Befehls werden temporär die entsprechenden CAN-Objekte angelegt, ausgewertet und anschließend wieder freigegeben; daher können Sie beliebig viele Module ansprechen.
	ACHTUNG!: Der Befehl arbeitet mit den vordefinierten PDOs von CANopen. Ändern Sie auf keinen Fall diese Default-Einstellung (minimum capability device), denn dann funktioniert der Befehl nicht mehr.
Portabilität	Die Funktion PDO lesen ist ab MCO 5.00 verfügbar.
Befehlsgruppe	I/O
Querverweise	CANINI, Produkthandbücher MCO 305 und FC 300
Syntax-Beispiel	an1 = INAD 53 PRINT "Analogeingang 53 " ,an1

□ INB

Kurzinfo	Zustand der digitalen Eingänge byteweise abfragen.
Syntax	erg = INB n
Parameter	<p>n = Eingangsbyte:</p> <p>0 = Eingang 1 (LSB) - 8 (MSB)</p> <p>1 = Eingang 33 (LSB) - 18 (MSB)</p> <p>2 = Eingang 9 - 10 (12)</p> <p>bzw. für CANopen I/O-Module:</p> <p>CAN-Bus + (Modul-CAN-ID * 256) + Eingangsnummer (bzw. Eingangsbyte)</p>
	<p>ACHTUNG!</p> <p>Die Nummerierung der Bytes beginnt bei 0; im Gegensatz zu den einzelnen Eingängen, deren Nummerierung bei 1 beginnt.</p>
Rückgabewert	<p>erg = Wert des Eingangsbytes (0 - 255)</p> <p>Das niederwertigste Bit entspricht dabei dem Zustand des Eingangs 1/33.</p>
Beschreibung	<p>Mit dem INB Befehl kann der Zustand der digitalen Eingänge byteweise abgefragt werden. Der zurückgelieferte Wert spiegelt den Zustand der einzelnen Eingänge wieder.</p> <p>Die Definition des High- und Low-Pegels sowie die Eingangsbeschaltung ist im FC 300 Produkthandbuch beschrieben.</p> <p>CAN-Module, welche die CANopen Spezifikationen erfüllen, können Sie ebenfalls mit dem INB Befehl ansprechen, und zwar über die entsprechende Nummer, die wie folgt definiert ist:</p> <p style="padding-left: 40px;">CAN-Bus + (Modul-CAN-ID * 256) + Eingangsnummer (bzw. Eingangsbyte)</p> <p>Beim Ausführen eines solchen Befehls werden temporär die entsprechenden CAN-Objekte angelegt, ausgewertet und anschließend wieder freigegeben. Daher können Sie beliebig viele Module ansprechen, aber es sind zunächst keine Objekte da, die von sich aus empfangsbereit sind, zum Beispiel für Interrupt-Funktionen. Um Interrupt-Funktionen durchzuführen, müssen Sie die entsprechenden Module vorher mit CANINI initialisieren.</p>
Befehlsgruppe	I/O
Querverweise	IN, OUT, OUTB, CANINI, FC 300 Produkthandbuch
Syntax-Beispiel	in = INB 0 /* Zustand der ersten 8 Eingänge speichern */
Beispiel	<p>IN1 = low, IN2 = high, IN3 = high,</p> <p>alle anderen Eingänge low</p> <p>erg = $2^1 + 2^2 = 6$</p>
Programmbeispiel	INB_01.M, INB_02.M, OUTB_01.M

□ INDEX

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

□ INGLB

Kurzinfo	Liest eine globale CAN-Nachricht über CAN-Bus
Syntax	erg = INGLB (p)
Parameter	p = maximale Wartezeit, definiert in ... p = 0 es wird gewartet bis eine Nachricht kommt p > 0 es wird maximal p Millisekunden gewartet p < 0 es wird nicht auf eine Nachricht gewartet
Rückgabewert	erg = -1, falls keine Nachricht gekommen ist, bzw. Bytes 2 und 3 der CAN-Nachricht, wenn eine Nachricht gekommen ist. Die globale Variable MSGVAL enthält dann die obersten Bytes 4 bis 7 der CAN-Nachricht.
Beschreibung	Mit diesem Befehl wird eine globale CAN-Nachricht gelesen, also eine Nachricht, die an alle Teilnehmer gesandt wurde. Diese Nachrichten haben den Identifier 0 und damit höchste Priorität.
	ACHTUNG! Diese Nachrichten werden nicht gepuffert und somit beim Eintreffen der nächsten Nachricht überschrieben
Portabilität	Standardbefehl
Befehlsgruppe	CAN
Querverweise	INMSG, OUTMSG

□ INKEY

Kurzinfo	Einlesen eines Zeichens der Tastatur.																																		
Syntax	INKEY (p)																																		
Parameter	p maximale Wartezeit, definiert in ... p = 0 es wird gewartet bis Zeichen kommt p > 0 es wird maximal p Millisekunden gewartet p < 0 es wird nicht auf Zeichen gewartet (ein negativer Parameter muss in Klammern angegeben werden)																																		
Rückgabewert	ASCII-Code des empfangenen Zeichens bzw. -1 falls kein Zeichen vorhanden ist. Folgende Tasten-Codes werden zurückgesendet, solange die Taste gedrückt wird. Werden mehr als eine Taste gleichzeitig gedrückt, wird die entsprechende Summe der Werte zurückgesendet: <table border="1" data-bbox="438 689 837 1288"> <thead> <tr> <th>Taste:</th> <th>Wert:</th> </tr> </thead> <tbody> <tr><td>[Main Menu]</td><td>1</td></tr> <tr><td>[Quick Menu]</td><td>2</td></tr> <tr><td>[Alarm Log]</td><td>4</td></tr> <tr><td>[Status]</td><td>8</td></tr> <tr><td>[OK]</td><td>16</td></tr> <tr><td>[Cancel]</td><td>32</td></tr> <tr><td>[Info]</td><td>64</td></tr> <tr><td>[Back]</td><td>128</td></tr> <tr><td>[→]-Taste / rechts</td><td>256</td></tr> <tr><td>[↑]-Taste / nach oben</td><td>512</td></tr> <tr><td>[↓]-Taste / nach unten</td><td>1024</td></tr> <tr><td>[←]-Taste / links</td><td>2048</td></tr> <tr><td>[Auto on]</td><td>4096</td></tr> <tr><td>[Reset]</td><td>8192</td></tr> <tr><td>[Hand on]</td><td>16384</td></tr> <tr><td>[Off]</td><td>32768</td></tr> </tbody> </table> Kombinationen senden folgende Werte: [OK] und [Cancel] 48 [Auto on] und [↑]-Taste 4608	Taste:	Wert:	[Main Menu]	1	[Quick Menu]	2	[Alarm Log]	4	[Status]	8	[OK]	16	[Cancel]	32	[Info]	64	[Back]	128	[→]-Taste / rechts	256	[↑]-Taste / nach oben	512	[↓]-Taste / nach unten	1024	[←]-Taste / links	2048	[Auto on]	4096	[Reset]	8192	[Hand on]	16384	[Off]	32768
Taste:	Wert:																																		
[Main Menu]	1																																		
[Quick Menu]	2																																		
[Alarm Log]	4																																		
[Status]	8																																		
[OK]	16																																		
[Cancel]	32																																		
[Info]	64																																		
[Back]	128																																		
[→]-Taste / rechts	256																																		
[↑]-Taste / nach oben	512																																		
[↓]-Taste / nach unten	1024																																		
[←]-Taste / links	2048																																		
[Auto on]	4096																																		
[Reset]	8192																																		
[Hand on]	16384																																		
[Off]	32768																																		
	 ACHTUNG!: Die Tasten behalten ihre FC 300-Funktionen, wenn sie nicht in Parameter 0-4* deaktiviert werden. ACHTUNG!: NLCP (LCP 101 Numerical Local Control Panel) ist derzeit nicht enthalten.																																		
Beschreibung	Mit dem INKEY Befehl kann ein Tastensignal vom FC 300 LCP-Tastenfeld eingelesen werden. Der mit INKEY übergebene Parameter bestimmt dabei, ob auf ein Tastensignal ohne Bedingung, eine gewisse Zeitspanne oder gar nicht gewartet wird. Pro erfolgreichen INKEY Befehl wird jeweils ein Tastensignal eingelesen. Für die Eingabe von Zeichenketten muss der INKEY Befehl (p<>0) in einer Schleife so oft wiederholt werden, bis keine weiteren Tastensignale mehr vorliegen.																																		
Befehlsgruppe	I/O																																		
Querverweise	PRINT																																		

__ Befehlsreferenz __

Syntax-Beispiel	<pre>input = INKEY 0 /* Warten bis Tastensignal gelesen wird */ character = INKEY 5000 /* max. 5 Sek. auf Eingabe warten */ character = INKEY (-1) /* nicht auf Eingabe warten */</pre>
Programmbeispiel	INKEY_01.M, EXIT_01.M, WHILE_01.M

□ INMSGG

Kurzinfo	CAN-Nachricht aus dem Puffer lesen
Syntax	intval = INMSG timeout
Parameter	<pre>timeout < 0 es wird nicht auf die Daten gewartet = 0 es wird gewartet, bis die Daten kommen > 0 es wird <i>timeout</i> [ms] auf die Daten gewartet</pre>
Rückgabewert	<p>INMSG liefert -1, falls keine Nachricht gekommen ist, bzw. Bytes 2 und 3 der CAN-Nachricht, wenn eine Nachricht gekommen ist.</p> <p>Die globale Variable MSGVAL enthält dann die obersten Bytes 4 bis 7 der CAN-Nachricht.</p>
Beschreibung	<p>Mit diesem Befehl wird eine Nachricht aus dem Puffer gelesen, wobei <i>timeout</i> eine analoge Bedeutung hat wie bei dem INKEY Befehl. Die Nachricht hat eine Id, die mit dem Befehl „\$N_slavenr_baudrate“ definiert wurde.</p> <p>Die CAN-Identifikationsnummer der Nachricht wird durch \$N-Befehl bestimmt.</p> <p>INMSG liest immer Objekte, die 8 Bytes lang sind. Für den Benutzer sind nur die Bytes von 2 bis 7 vorgesehen; Byte 0 und 1 sind reserviert.</p>
Portabilität	Standardbefehl
Befehlsgruppe	CAN
Querverweise	OUTMSG, ON CANMSG
Syntax-Beispiel	<pre>a = INMSG -1 IF (a > -1) THEN b = MSGVAL ENDIF</pre>

□ IPOS

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



__ Befehlsreferenz __

Beispiel

```

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

```

□ IPOSDIFF

Kurzinfo Overflow-Handling von Inkrementalgebern in Anwendungen.

Syntax erg = IPOSDIFF oldpos

Parameter oldpos = IPOS zu einem früheren Zeitpunkt

Rückgabewert Liefert die Differenz zwischen IPOS und oldpos (erg = IPOS – oldpos) in BE.

Beschreibung Dieser Befehl vereinfacht die Behandlung des Überlaufs von Inkrementalgebern in Anwendungen. Wenn zum Beispiel der Anwender eine aktuelle Position in seinem Programm speichert und später die Differenz berechnen will, muss er normalerweise den Überlauf der Position berücksichtigen. Stattdessen kann dieser Befehl benutzt werden; siehe unten.

Intern prüft diese Routine, ob die Differenz größer als POS_LIMIT (0x3FFFFFFF) ist. Falls dies der Fall ist, wird angenommen, dass ein Überlauf stattfand und korrekt behandelt.



ACHTUNG!

Dies löst aber nicht das Problem des Überlaufs, wenn in der Anwendung Benutzer-einheiten [BE] verwendet werden.

Portabilität Der Befehl ist ab MCO 5.00 verfügbar.

Befehlsgruppe SYS

Querverweise IPOS

Syntax-Beispiel oldpos = IPOS

```

..
diff = IPOSDIFF oldpos
// liefert die Differenz zwischen IPOS und oldpos in BE
// Behandlung eines Overflows falls notwendig (diff = IPOS – oldpos)

```

□ JERKFINVEL

Kurzinfo	Berechnet die Endgeschwindigkeit für einen ruckbegrenzten Stopp mit maximaler Beschleunigung/Verzögerung.
Syntax	erg = JERKFINVEL
Parameter	-
Rückgabewert	erg = in Prozent (oder VELRES Einheiten)
Beschreibung	Der Befehl berechnet die Endgeschwindigkeit, die erreicht werden würde, wenn die aktuelle Beschleunigung (oder Verzögerung) unter Berücksichtigung von vorgegebenen JERKMIN Werten stoppt. Dies funktioniert auch, wenn etwa eine Fahrbewegung anders als mit RAMPTYPE 2 aktiv ist. Daher kann auch die Endgeschwindigkeit berechnet werden, wenn zum Beispiel die aktuelle SYNCP verlassen wird. Das Ergebnis ist in Prozent (oder VELRES Einheiten) und kann daher ohne Konvertierung für einen VEL oder CVEL Befehl verwendet werden.
	Das Ergebnis könnte auch negativ sein (z.B. beim Rückwärtsfahren oder nahe an der Geschwindigkeit Null mit einer hohen Verzögerung).
Portabilität	Der Befehl ist ab MCO 5.00 verfügbar.
Befehlsgruppe	SYS
Querverweise	Par. 32-82 <i>Rampenform</i> RAMPTYPE, Par. 32-86 <i>Beschleunigungsrampe für Ruckbegrenzung</i> JERKMIN,

□ JERKSTOPDIST

Kurzinfo	Berechnet die notwendige Distanz für einen ruckbegrenzten Stopp mit maximaler Verzögerung.
Syntax	erg = JERKSTOPDIST dec
Parameter	dec = setzt die Verzögerung in % oder VELRES Einheiten
Rückgabewert	erg = Distanz in BE
Beschreibung	Der Befehl berechnet den Abstand, den die Achse n benötigt um zu stoppen, wenn eine maximale Verzögerung von DEC erlaubt ist und der RAMPTYPE 2 mit JERKMIN2 und JERKMIN4 (oder JERKMIN) benutzt wird. In dem Befehl ist die Verzögerung DEC in Prozent (oder VELRES Einheiten) und das Ergebnis in Benutzereinheiten gegeben. Dieser Befehl funktioniert auch wenn keine RAMPTYPE 2 Fahrbewegung aktiv ist. So kann man zum Beispiel in einer SYNCP Positionierung eine RAMPTYPE 2 Stopprampe berechnen.
Portabilität	Der Befehl ist ab MCO 5.00 verfügbar.
Befehlsgruppe	SYS
Querverweise	DEC, Par. 32-83 VELRES, Par. 32-86 <i>Beschleunigungsrampe für Ruckbegrenzung</i> JERKMIN, Par. 32-87 <i>Beschleunigungsdauer für Ruckbegrenzung</i> , Par. 32-89 <i>Bremsdauer für Ruckbegrenzung</i>

□ LINKGPAR

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

□ LINKPDO

Kurzinfo Mapping der RxPDOs: Inhalte eines RxPDO mit Elementen des internen Systemvariablen-Pseudoarrays SYSVAR verknüpfen. Jede Änderung des RxPDOs wird in der Folge automatisch in das konfigurierte SYSVAR-Element kopiert.

Syntax LINKPDO nr len indx pdo

Parameter

nr = Reihenfolge im RxPDO beginnend bei 1

len = Anzahl der Bits, die übernommen werden sollen;
Bedingung: Länge = Vielfaches von 8 (bitweise);
(z.B. 128 um 4 long Werte in das PDO zu kopieren, falls PDO mehr als 8 Bytes hat.)

indx = Index des Systemvariablen-Pseudoarrays SYSVAR

pdo = 1 .. 4 oder 5 ("serielles" PDO 5);
Um Abwärtskompatibilität zu gewährleisten werden alle anderen Werte (z.B. auch der frühere Defaultwert 0) als 1 (= PDO 1) interpretiert.



ACHTUNG!

Die Reihenfolgen-Nummer (nr) muss – beginnend mit 1 – aufsteigen.

Multiple PDOs

Der Befehl kann für beliebige PDOs verwendet werden, d.h. PDO 2 .. PDO 5 werden bei aktuellen Firmware- und Compiler-Versionen ebenfalls unterstützt. Die PDO-Nummer wird mittels dem letzten Parameter (der in älteren Firmware-Versionen reserviert war) des LINKPDO-Befehls übergeben. Um Abwärtskompatibilität zu garantieren, wird jede Zahl abweichend von dem zulässigen Wertebereich als 1 interpretiert, d.h. das PDO 1 ist Default. Dies ermöglicht die Abwärtskompatibilität mit älteren Versionen, bei denen keine PDO-Auswahl möglich war oder der letzte Parameter auf 0 gesetzt wurde.



CANopen PDO Größe

Ein PDO ist immer 8 Byte lang; es kann daher maximal 8 Objekte enthalten.

PDO 5 (= „Serielles PDO“) Größe

Die Mailbox-Größe des PDO 5 kann eine Länge von maximal circa 250 Bytes besitzen. Das PDO 5 wird ebenfalls von dem Oszilloskop-Tool der APOSS Entwicklungsumgebung genutzt. Es wird deshalb empfohlen dieses PDO nicht in Applikationsprogrammen zu verwenden, die später mit dem Oszilloskop-Tool gedebugged werden sollen.

Beschreibung

Der Befehl LINKPDO verknüpft Inhalte von RxPDOs mit beliebigen Elementen des Systemvariablen-Pseudoarrays SYSVAR. Dies wird als PDO-Mapping bezeichnet. Die Inhalte des RxPDOs werden in der Folge automatisch, entsprechend der mit LINKPDO erstellten Mapping-Konfiguration in die zugewiesenen Elemente des Systemvariablen-Pseudoarrays kopiert.

Das Systemvariablen-Pseudoarray SYSVAR umfasst sämtliche SDOs (gemäß SDO Object Dictionary) und somit auch jeweils die ersten 250 Elemente von Anwendungsarrays. Es ist nahezu beliebig möglich den Inhalt von RxPDOs in interne Variablen, Parameter oder Arrays zu leiten. Der benötigte SYSVAR-Index eines SDOs kann wie folgt berechnet werden:

$$0x01000000 + ("SDO-Index" \ll 8) + "SDO-Subindex"$$

Beispiel 1: SDO 0x2300 / 12 (= SDO mit dem Achsparameter KPROP der Achse 1)
=> SYSVAR-Index: 0x0123000C

Beispiel 2: SDO 0x2100 / 5 (= SDO mit 1.Element des Anwendungsarrays 1)

__ Befehlsreferenz __

=> SYSVAR-Index: 0x01210005

Das RxPDO ist gleichzeitig in dem PDO-Array abgebildet. In diesem stehen die identischen Dateninhalt ebenfalls zur Verfügung.

Automatische PDO Aktivierung

Gemäß CANopen-Spezifikation ist nur das PDO 1 1 (RxPDO = 0x200 + Node-ID / TxPDO = 0x180 + Node-ID) standardmäßig enabled, d.h. das „Valid“ Bit (0x1400 / Subindex 1) gesetzt. Bei der Mapping-Konfiguration weiterer PDOs mit LINKPDO (oder LINKSDO) wird automatisch das „Valid“-Bit (0x1401 - 0x1404, Subindex 1) dieser RxPDOs ebenfalls gesetzt.

CANopen und APOSS Mapping

Bezieht sich der SYSVAR-Index auf ein SDO aus dem SDO Object Dictionary der Steuerung, (d.h. SYSVAR-Index beginnt mit 0x01....) wird intern ein reines CAN-Mapping ausgeführt. Wenn ein entsprechendes Objekt über einen SDO-Befehl geändert wird, wird sofort auch das PDO neu geschrieben. Das Mapping für das entsprechende CAN-Objekt kann von einer übergeordneten Steuerung mit den Standard CANopen Mapping-Objekten ausgelesen werden.

Werden andere Indizes verwendet, wird ein APOSS-Mapping durchgeführt. Dieses kann zwar gleichzeitig zum CAN-Mapping stattfinden, ist aber nicht CANopen konform, da die Einträge nicht mit den CANopen Mapping-Objekten lesbar sind.



ACHTUNG!

Die Verknüpfung mit internen Systemvariablen ist sorgfältig durchzuführen und sollte nur von Anwendern, die mit APOSS Erfahrung haben, vorgenommen werden. Es sind gründliche Kenntnisse über die Benutzung und Bedeutung der internen Systemvariablen notwendig, um kein fehlerhaftes Systemverhalten zu verursachen.

Portabilität Der Befehl ist ab MCO 5.00 verfügbar.

Befehlsgruppe PAR

Querverweise LINKSDO, SYSVAR, Parameter-Referenz, PDO

Syntax-Beispiel 1 // Das RxPDO 1 mit dem Anwendungsparameter (= SDO 0x2201/01) verknüpfen
LINKPDO 1 32 0x01220101 1

Syntax-Beispiel 2 // 8 Bits des RxPDOs 1 mit den digitalen Ausgängen 1...8
// (= SDO 2202/10) verknüpfen
LINKPDO 1 8 0x0122020A 1
// Nächste 8 Bits des RxPDOs 1 mit den digitalen Ausgängen 9...16
// (= SDO 0x2202/11) verknüpfen
LINKPDO 2 8 0x0122020B 1

Syntax-Beispiel 3 // 16 Bits des RxPDOs 1 mit dem DS402 ControlWord
// (= SDO 0x6040/0) verknüpfen
LINKPDO 1 16 0x01604000 1

□ LINKSDO

Kurzinfo Mapping der TxPDOs: Elemente des internen Systemvariablen-Pseudoarrays SYSVAR mit einem TxPDO verknüpfen. Jede Änderung des entsprechenden SYSVAR-Elements wird in der Folge automatisch in das TxPDO übernommen.

Syntax LINKSDO indx len nr "text" pdo

Parameter

indx = Index der Systemvariable SYSVAR

len = Länge der Bits, die übernommen werden sollen;
Bedingung: Länge = Vielfaches von 8 (bitweise),
(z.B. 128 um 4 Long Werte in das PDO zu kopieren, falls PDO mehr als 8 Bytes hat.)

nr = Reihenfolge im PDO

text = " " (wird noch nicht ausgewertet, kann aber als Kommentar genutzt werden)

pdo = Werte zwischen 1 .. 4 oder 5 („serielles“ PDO 5)



ACHTUNG!

Die Reihenfolgen-Nummer (nr) muss – beginnend mit 1 – aufsteigen.

Multiple PDOs

Der Befehl kann für beliebige PDOs verwendet werden, d.h. PDO 2 - PDO 5 werden bei aktuellen Firmware- und Compilerversionen ebenfalls unterstützt. Die PDO-Nummer wird mittels dem letzten Parameter (der in älteren Firmware-Versionen reserviert war) des LINKSDO-Befehls übergeben. Um Abwärtskompatibilität zu garantieren, wird jede Zahl abweichend von dem zulässigen Wertebereich als 1 interpretiert, d.h. das PDO 1 ist Default. Dies ermöglicht die Abwärtskompatibilität mit älteren Versionen, bei denen keine PDO-Auswahl möglich war oder der letzte Parameter auf 0 gesetzt wurde



CANopen PDO Größe

Ein PDO ist immer 8 Byte lang; es kann daher maximal 8 Objekte enthalten.



PDO 5 (= „Seriellles PDO“) Größe

Die Mailbox-Größe des PDO 5 kann eine Länge von maximal circa 250 Bytes besitzen. Das PDO 5 wird ebenfalls von dem Oszilloskop-Tool der APOSS Entwicklungsumgebung genutzt. Es wird deshalb empfohlen dieses PDO nicht in Applikationsprogrammen zu verwenden, welche später mit dem Oszilloskop-Tool gedebugged werden sollen.

Beschreibung Der Befehl LINKSDO verknüpft beliebige Elementen des Systemvariablen-Pseudoarrays SYSVAR mit einem TxPDO. Dies wird als PDO-Mapping bezeichnet. Jede Änderung in dem verknüpften SYSVAR-Element wird in der Folge automatisch, entsprechend der mit LINKSDO erstellten Mapping-Konfiguration, in das TxPDO kopiert.

Das Systemvariablen-Pseudoarray SYSVAR umfasst sämtliche SDOs (gemäß SDO Object Dictionary) und somit auch jeweils die ersten 250 Elemente von Anwendungsarrays. Der Inhalt von Variablen, Arrays und Parametern kann durch die Konfiguration (= PDO-Mapping) mit LINKSDO in TxPDOs geleitet werden. Der benötigte SYSVAR-Index eines SDOs kann wie folgt berechnet werden:
 $0x01000000 + ("SDO-Index" << 8) + "SDO-Subindex"$

Beispiel 1: SDO 0x2500 / 1 (= SDO mit der aktuellen Position der Achse 1)
=> SYSVAR-Index: 0x01250001

Beispiel 2: SDO 0x2100 / 5 (= SDO mit 1.Element des Anwendungsarrays 1)

__ Befehlsreferenz __

=> SYSVAR-Index: 0x01210005

Das TxPDO ist gleichzeitig in dem PDO-Array abgebildet. In diesem stehen die identischen Dateninhalt zur Verfügung.



ACHTUNG!

Standardgemäß wird ein verändertes TxPDO automatisch verschickt (Asynchron Modus). Wenn dies nicht gewünscht ist, kann über das SDO 0x1800 - 0x1804 Subindex 2 der Übertragungstyp auf eine anderen Einstellung konfiguriert werden (z.B. 254, statt Standard 255). Mit dieser Umkonfiguration wird nicht mehr aktiv verschickt, sondern das PDO muss mittels Remote-Frame abgeholt werden.

Automatische PDO Aktivierung

Gemäß CANopen-Spezifikation ist nur das PDO 1 (RxPDO = 0x200 + Node-ID / TxPDO = 0x180 + Node-ID) standardmäßig enabled, d.h. das „Valid“ Bit (0x1800 / Subindex 1) gesetzt. Bei der Mapping-Konfiguration weiterer PDOs mit LINKSDO (oder LINKPDO) wird automatisch das „Valid“-Bit (0x1801 - 0x1804, Subindex 1) dieser TxPDOs ebenfalls gesetzt.

CANopen und APOSS mapping

Bezieht sich der SYSVAR-Index auf ein SDO aus dem SDO Object Dictionary der Steuerung, (d.h. SYSVAR-Index beginnt mit 0x01....) wird intern ein reines CAN-Mapping ausgeführt. Wenn ein entsprechendes Objekt über einen SDO-Befehl geändert wird, wird sofort auch das PDO neu geschrieben. Das Mapping für das entsprechende CAN-Objekt kann von einer übergeordneten Steuerung mit den Standard CANopen Mapping-Objekten ausgelesen werden.

Werden andere Indizes verwendet, wird ein APOSS-Mapping durchgeführt. Dieses kann zwar gleichzeitig zum CAN-Mapping stattfinden, ist aber nicht CANopen konform, da die Einträge nicht mit den CANopen Mapping-Objekten lesbar sind.



ACHTUNG!

Die Verknüpfung mit internen Systemvariablen ist sorgfältig durchzuführen und sollte nur von in APOSS erfahrenen Anwendern vorgenommen werden. Es sind gründliche Kenntnisse über die Benutzung und Bedeutung der internen Systemvariablen notwendig, um kein fehlerhaftes Systemverhalten zu verursachen.

Portabilität Der Befehl ist ab MCO 5.00 verfügbar. Mit dieser Version wurde auch der Befehl #DEBUG durch den Debug Modus ersetzt.

Befehlsgruppe PAR

Querverweise LINKPDO, SYSVAR, Parameter-Referenz,, PDO, Debug-Befehle

Syntax-Beispiel 1 // Den aktuellen Schleppfehler (= SDO 0x0x2500/6) mit dem TxPDO 1 verknüpfen
LINKSDO 0x01250006 32 1 " " 1

Syntax-Beispiel 2 // Digitale Eingänge 1...8 (= SDO 2202/02) mit TxPDO 1 verknüpfen
LINKSDO 0x01220202 8 1 " " 1
// Digitale Eingänge 9...16 (= SDO 2202/02) mit nächsten 8 Bits des TxPDO 1 verknüpfen
LINKSDO 0x01220203 8 2 " " 1

Syntax-Beispiel 3 // Das DS402 Statuswort (= SDO 0x6041/0) mit dem TxPDO 1 verknüpfen
LINKSDO 0x01604100 16 1 " " 1
// Die DS402 "Betriebsartanzeige" (= SDO 0x6061/0) mit dem TxPDO 1 verknüpfen
LINKSDO 0x01606100 8 2 " " 1

□ LINKSYSVAR

Kurzinfo	Systemvariable mit LCP-Display verknüpfen.
Syntax	LINKSYSVAR indx parnr "text"
Parameter	indx = Index der Systemvariable SYSVAR parnr = LCP-Parameternummer 1900 bis 1999 text = beschreibender Text für Display
Beschreibung	Der Befehl LINKSYSVAR verknüpft die Systemvariable SYSVAR[indx] mit dem FC 300 Parameter (19-00 to 19-99) und dem Anzeigetext "text". Auf diese Weise können Sie interne Werte ohne Umweg über LINKGPAR auf dem Display darstellen.
	ACHTUNG!: Alle 40 ms wird der Parameter aktualisiert. Wenn also auf diese Weise fünf Parameter verknüpft werden, dauert es mindestens 200 ms, bis derselbe Parameter wieder erneuert wird.
Befehlsgruppe	PAR
Querverweise	LINKGPAR, SYSVAR, Anwendungsparameter, Parameter-Referenz
Syntax-Beispiel	LINKSYSVAR 33 1990 "interne Zeilennummer" LINKSYSVAR 30 1991 "Motorspannung"

□ LOOP

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

□ MAPOS

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

□ MAPOSDIFF

Kurzinfo	Overflow-Handling von Inkrementalgebern in Anwendungen.
Syntax	erg = MAPOSDIFF oldpos
Parameter	n = Achsnummer oldpos = MAPOS zu einem früheren Zeitpunkt
Rückgabewert	Liefert die Differenz zwischen MAPOS und oldpos (erg = MAPOS – oldpos) in BE.
Beschreibung	Dieser Befehl vereinfacht die Behandlung des Überlaufs von Inkrementalgebern in Anwendungen. Wenn zum Beispiel der Anwender eine aktuelle Position in seinem Programm speichert und später die Differenz berechnen will, muss er normalerweise den Überlauf der Position berücksichtigen. Statt dessen kann dieser Befehl benutzt werden; siehe unten. Intern prüft diese Routine, ob die Differenz größer als POS_LIMIT (0x3FFFFFFF) ist. Falls dies der Fall ist, wird angenommen, dass ein Überlauf stattfand und korrekt behandelt.
	ACHTUNG! Dies löst aber nicht das Problem des Überlaufs, wenn in der Anwendung Benutzereinheit BE verwendet werden.
Portabilität	Der Befehl ist ab MCO 5.00 verfügbar.
Befehlsgruppe	SYS
Querverweise	MAPOS
Syntax-Beispiel	oldpos = MAPOS .. diff = MAPOSDIFF oldpos // liefert die Differenz zwischen MAPOS und oldpos in BE // Behandlung eines Overflows falls notwendig (diff = MAPOS – oldpos)

□ MAVEL

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



□ MENCPOSOFFS

Kurzinfo	Synchronisiert den inkrementalen Positionszähler mit dem absoluten im Encoder.	
Syntax	erg = MENCPOSOFFS offset	
Parameter	offset = Liefert die Differenz zwischen der absoluten und inkrementalen Position (absolute – inkremental)	
Rückgabewerte	OK 0	Der Befehl war erfolgreich.
	TIMEOUT -1	Keine Antwort innerhalb von 300 ms erhalten.
	BADFRAME -2	Der erhaltene Frame ist nicht gültig.
	OVERFLOW -4	Mehr Bytes erhalten, als der Empfangspuffer aufnehmen kann.
Beschreibung	<p>Es wird die Differenz zwischen der absoluten Encoder-Position und dem inkrementalen Zähler bestimmt und zurückgeliefert.</p> <p>Hierfür wird der inkrementale Zähler im DSP exakt in dem Moment gelatcht, in dem auch der Hiperface-Encoder die absolute Position latcht und diese via RS485 liefert.</p> <p>Mit Hilfe dieser Differenz kann der Anwender z.B. die Position innerhalb APOSS mit SETMORIGIN auf den absoluten Wert setzen.</p> <p>Wird der Hiperface-Encoder als Slave-Signal statt eines Master-Signals benutzt, verwenden Sie den Befehl ENCPOSOFFS (siehe Parameter 32-52).</p>	
Befehlsgruppe	SYS	
Querverweis	ENCPOSOFFS	
Programmbeispiel	Siehe Programmbeispiel bei ENCPOSOFFS.	

□ MENCTGREAD

Kurzinfo	Liest ein RS485 Telegramm vom Encoder.	
Syntax	erg = MENCTGREAD array	
Parameter	array = Benutzer-Array, in das die erhaltene Datenmenge geschrieben werden soll.	
Rückgabewerte	OK x (>0)	TG mit x Byte Anwenderdaten angekommen
	ACTIVE 0	Die Übertragung läuft noch.
	TIMEOUT -1	Keine Antwort innerhalb von 300 ms erhalten.
	BADFRAME -2	Der erhaltene Frame ist nicht gültig.
	OVERFLOW -4	Mehr Bytes erhalten, als der Empfangspuffer aufnehmen kann.
Beschreibung	<p>Nachdem ein Telegramm mit MENCTGWRITE gesendet wurde, kann die Antwort mit diesem Befehl gepollt werden. Der Rückgabewert wird ausgegeben, entweder wenn er schon angekommen ist oder wenn ein Timeout aufgetreten ist.</p> <p>Wird der Hiperface-Encoder als Slave-Signal statt eines Master-Signals benutzt, verwenden Sie den Befehl ENCPOSOFFS (siehe Parameter 32-52).</p>	
Befehlsgruppe	SYS	
Querverweise	ENCTGREAD, MENCTGWRITE	
Programmbeispiel	Siehe Programmbeispiel bei ENCTGREAD.	

□ MENCTGWRITE

Summary	Sendet ein RS485 Telegramm zum Encoder.
Syntax	erg = MENCTGWRITE länge array
Parameter	länge = Anzahl der zu sendenden Bytes (im Benutzer-Array). array = Benutzer-Array, das die zum Encoder zu sendende Datenmenge enthält.
Rückgabewerte	OK 0 Telegramm wurde gesendet BUSY -3 Es läuft noch eine andere Übertragung; bis jetzt noch kein Timeout.
Beschreibung	Dieser Befehl sendet ein RS485 Telegramm zum Encoder mit der ID „MENCODERID“. Der Anwender muss zuvor die Datenmenge in das Array eintragen. Der Befehl füllt dann diese Daten in einen normalen RS485 Datenübertragungsblock und fügt einen CRC Wert hinzu. Der Befehl wartet nicht bis die Daten gesendet wurden oder eine Antwort erhalten wurde; der Rückgabewert kommt sofort. Das Antworttelegramm muss mit MENCTGREAD gepollt werden. Wird der Hiperface-Encoder als Slave-Signal statt eines Master-Signals benutzt, verwenden Sie den Befehl ENCTGWRITE (siehe Parameter 32-52).
Befehlsgruppe	SYS
Querverweise	MENCTGREAD, ENCTGWRITE
Programmbeispiel	Siehe Programmbeispiel bei ENCTGREAD.



□ MIPOS

Kurzinfo Letzte Index- bzw. Markerposition des Masters abfragen.

Syntax erg = MIPOS

Rückgabewert erg = letzte Index- bzw. Markerposition des Masters absolut zum aktuellen Nullpunkt in qc

Beschreibung Diese Funktion liefert die letzte Index- bzw. Markerposition des Masters absolut zum aktuellen Nullpunkt in qc zurück.
Die Konfiguration von MIPOS, d.h. ob die Index- oder Markerposition des Master-Drehgebers (= geregelter Antrieb) zurückgeliefert wird, erfolgt über den Parameter 33-19 *Markertyp Master*.

**ACHTUNG!:**

Das Triggersignal für die Markerposition muss dabei zwingend an den Eingang 5 angeschlossen werden.

Der Positionswert in MIPOS ist auf ± 1 qc genau. Im Gegensatz zu der Positionsinformation in MAPOS, welche nur im Reglerzyklus von typisch 1 ms aktualisiert wird, wird der aktuelle Positionswert hardwaremäßig beim Auftreten des konfigurierten Signals (in einem internen Prozessorregister) in Echtzeit zwischengespeichert und dann in die Systemvariable MIPOS kopiert.

Falls gleichzeitig zur Markerposition ein Interrupt ausgelöst wird (ON INT 5 GOSUB ...) und in diesem Interrupt mit MIPOS gearbeitet wird, sollte im Interrupt-Unterprogramm eine Verzögerung von 2 Millisekunden (DELAY 2) vor dem Lesen von MIPOS verwendet werden. So kann sichergestellt werden, dass der gelachte Positionswert bereits vollständig in die Systemvariable MIPOS kopiert ist und nicht noch auf einen veralteten Wert zurückgegriffen wird.

**ACHTUNG!:**

Der Befehl MIPOS kann nicht verwendet werden:

- Bei Einsatz von Absolutgebern (siehe Par. 32-30 *Inkrementalgeber Signaltyp*).
- Wenn der Parameter 32-50 auf [3] – Motor Steuerung steht.

Befehlsgruppe SYS

Querverweise CPOS, DEFORIGIN, SETORIGIN, POSA, POSR, ON INT
Parameter: 32-12 *Benutzerfaktor Zähler*, 32-11 *Benutzerfaktor Nenner*, 33-19 *Markertyp Master*

Syntax-Beispiel PRINT MIPOS /* letzte Indexposition des Masters am PC ausgeben */

Beispiel

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

__ Befehlsreferenz __

```

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

```

□ MIPOSDIFF

Kurzinfo Overflow-Handling von Inkrementalgebern in Anwendungen.

Syntax erg = MIPOSDIFF oldpos

Parameter oldpos = MIPOS zu einem früheren Zeitpunkt

Rückgabewert Liefert die Differenz zwischen MIPOS und oldpos (erg = MIPOS - oldpos) in BE.

Beschreibung Dieser Befehl vereinfacht die Behandlung des Überlaufs von Inkrementalgebern in Anwendungen. Wenn zum Beispiel der Anwender eine aktuelle Position in seinem Programm speichert und später die Differenz berechnen will, muss er normalerweise den Überlauf der Position berücksichtigen. Statt dessen kann dieser Befehl benutzt werden; siehe unten.

Intern prüft diese Routine, ob die Differenz größer als POS_LIMIT (0x3FFFFFFF) ist. Falls dies der Fall ist, wird angenommen, dass ein Überlauf stattfand und korrekt behandelt.



ACHTUNG!

Dies löst aber nicht das Problem des Überlaufs, wenn in der Anwendung Benutzereinheit BE verwendet werden.

Portabilität Der Befehl ist ab MCO 5.00 verfügbar.

Befehlsgruppe SYS

Querverweise MIPOS

Syntax-Beispiel oldpos = MIPOS

```

..
diff = MIPOSDIFF oldpos
  // liefert die Differenz zwischen MIPOS und oldpos in BE
  // Behandlung eines Overflows falls notwendig (diff = MIPOS - oldpos)

```

□ MOTOR OFF

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

□ MOTOR ON

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

□ MOTOR STOP

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

□ MOVESYNCORIGIN

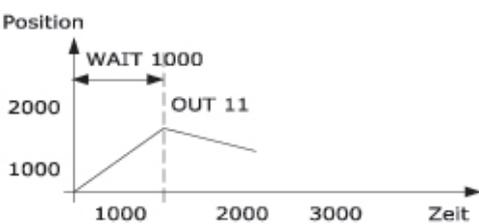
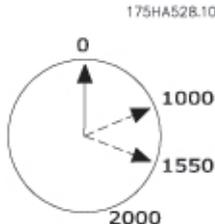
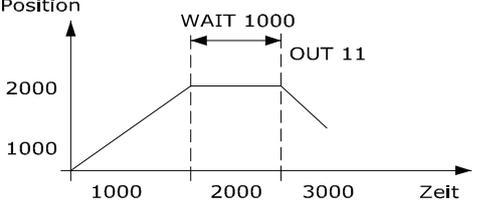
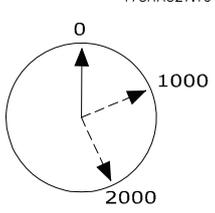
Kurzinfo	Synchronisationsursprung relativ verschieben.
Syntax	MOVESYNCORIGIN mwert
Parameter	<p>mwert = Relativer Offset in Bezug zum Master in qc</p> <p>Wertebereich: (-MLONG / Par. 33-11 SYNCFACTS) – (MLONG / Par. 33-11 SYNCFACTS)</p>
Beschreibung	<p>Der Befehl verschiebt den Synchronisationsursprung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

□ MSGVAL

Kurzinfo	Enthält den zweiten Teil der zuletzt gelesenen CAN-Nachricht.
Syntax	longval = MSGVAL
Parameter	-
Rückgabewert	longval = Byte 4 bis 7 der zuletzt gelesenen CAN-Nachricht
Beschreibung	MSGVAL ist eine Variable, die den Long-Wert der CAN-Message liefert. Die CAN-Message muss vorher mit INMSG oder INGLB gelesen werden.
	ACHTUNG! Der Wert ist nur gültig, solange kein neuer Befehl INMSG oder INGLB ausgeführt wurde.
Portabilität	Der Befehl ist ab MCO 5.00 verfügbar.
Befehlsgruppe	CAN
Querverweise	INMSG, INGLB, ON CANMSG
Syntax-Beispiel	a = INMSG -1 IF (a > -1) THEN b = MSGVAL ENDIF

__ Befehlsreferenz __

□ NOWAIT

Kurzinfo	Nach Positionierbefehlen warten / nicht warten.
Syntax	NOWAIT s
Parameter	s = Bedingung: ON = Programmausführung fortsetzen, bis die Zielposition erreicht ist. OFF = Programmausführung anhalten bis die Zielposition erreicht ist.
Beschreibung	Ein NOWAIT Befehl definiert das Verhalten nach Positionierbefehlen mit den beiden Zuständen NOWAIT ON und NOWAIT OFF: NOWAIT ON Der Befehl ermöglicht sowohl die Position anzufahren als auch die darauf folgenden Anweisungen auszuführen.
	<div style="display: flex; justify-content: space-around; align-items: flex-start;"> <div style="text-align: left;"> <p>NOWAIT ON POSA 2000 WAIT 1000 OUT 11 POSA 1000</p> </div> <div style="text-align: center;">  </div> <div style="text-align: center;">  </div> </div> <p>Bei NOWAIT ON wird nach dem Starten eines Positionierbefehls sofort die weitere Abarbeitung der Befehle fortgesetzt und der Positioniervorgang läuft quasi im Hintergrund ab. Im Zustand NOWAIT ON ist es somit auch möglich, während eines Positioniervorgangs die Istposition abzufragen, die Geschwindigkeit oder die Zielposition zu ändern.</p>
	<p>NOWAIT OFF Der Befehl erlaubt die zeilenweise Ausführung des Programms. Positionierbefehle werden vollständig, das heißt bis zur Zielposition ausgeführt, bevor mit der Abarbeitung der folgenden Befehle begonnen wird.</p> <div style="display: flex; justify-content: space-around; align-items: flex-start;"> <div style="text-align: left;"> <p>NOWAIT OFF POSA 2000 WAIT 1000 OUT 11 POSA 1000</p> </div> <div style="text-align: center;">  </div> <div style="text-align: center;">  </div> </div> <p>ANMERKUNG: Die Pfeile mit gestrichelter Linie zeigen die Positionen der Bewegung.</p>
	<p>ACHTUNG!: Der Default-Zustand ist NOWAIT OFF. Wenn also innerhalb eines Programms keine NOWAIT ON Anweisung enthalten ist, werden Positioniervorgänge immer vollständig ausgeführt, bevor mit der Abarbeitung des nächsten Befehls begonnen wird.</p> <p>Wenn im Zustand NOWAIT ON während eines noch aktiven Positioniervorgangs ein weiterer Positionierbefehl erfolgt, wird ohne Unterbrechung und ohne voriges Anfahren der ersten Zielposition sofort auf die neue Endposition verfahren.</p> <p>Sowohl der HOME, wie auch der INDEX Befehl werden im Zustand NOWAIT ON zu Ende abgearbeitet, bevor mit der Ausführung des nächsten Befehls begonnen wird.</p>
Befehlsgruppe	CON
Querverweise	WAITAX, AXEND, POSA, POSR, HOME, INDEX
Syntax-Beispiel	NOWAIT ON /* nach Positionierbefehlen nicht warten */ NOWAIT OFF /* nach Positionierbefehlen warten bis Ziel erreicht */
Programmbeispiele	NOWAI_01.M, MSTOP_01.M, OUT_01.M, VEL_01.M



□ ON CANINPUT

Kurzinfo	Unterprogramm aufrufen, wenn ein CAN-Telegramm vom Typ 'id' ankommt.
Syntax	ON CANINPUT id GOSUB name
Parameter	id = 0 name = Name des Unterprogramms
Beschreibung	<p>Mit der Anweisung ON CANINPUT wird ein Unterprogramm aufgerufen, wenn ein globales CAN-Telegramm mit der Id '0' ankommt. Dies ist das Eingangstelegramm, das auch mit INGLB und MSGVAL ausgelesen werden kann.</p> <p>Dieses globale Telegramm wird auch für den Programmabbruch verwendet. Daher ist darauf zu achten, dass Byte 0 und 1 nicht verwendet werden dürfen und immer auf 0 stehen sollten. Die Bytes 2 bis 7 dagegen können frei verwendet werden.</p> <p>Es besteht auch die Möglichkeit mit ON CANINPUT auf eintreffendes Nulltelegramm zu reagieren.</p>
	<p>ACHTUNG!</p> <ul style="list-style-type: none"> – Die Anweisung sollte am Programmstart stehen, damit sie für das ganze Programm Gültigkeit besitzt. – Das aufzurufende Unterprogramm muss innerhalb des durch SUBMAINPROG und ENDPROG. gekennzeichneten Programmbereichs definiert sein.
Portabilität	Der Befehl ist ab MCO 5.00 verfügbar.
Befehlsgruppe	INT
Querverweise	ON ERROR, ON INT, ON PERIOD, ENABLE .., DISABLE ..
Syntax-Beispiel	ON CANINPUT 0 GOSUB break /* Interrupt Prozedur wird definiert*/

□ ON CANMSG GOSUB

Kurzinfo	Aufruf eines Unterprogramms
Syntax	ON CANMSG GOSUB name
Parameter	name = Name des Unterprogramms
Beschreibung	Aufruf eines Unterprogramms bei einer Nachricht im Puffer. Unterprogramm 'name' wird aufgerufen, wenn mindestens eine Nachricht im CAN Empfangspuffer vorhanden ist
	<p>ACHTUNG!</p> <p>Die Anweisung ON CANMSG GOSUB sollte am Programmstart stehen, damit sie für das ganze Programm Gültigkeit besitzt.</p> <p>Das aufzurufende Unterprogramm muss innerhalb des durch SUBMAINPROG und ENDPROG. gekennzeichneten Programmbereichs definiert sein.</p>
Portabilität	Der Befehl ist ab MCO 5.00 verfügbar.
Befehlsgruppe	INT
Querverweise	ON ERROR, ON INT, ON PERIOD, ENABLE .., DISABLE ..
Syntax-Beispiel	ON CANMSG GOSUB can proc /* Interrupt Prozedur wird definiert */

□ ON COMBIT .. GOSUB

Kurzinfo	Unterprogramm aufrufen, wenn Bit n des Kommunikationspuffers gesetzt wird.
Syntax	ON COMBIT n GOSUB name
Parameter	n = Bit n des Kommunikationspuffers -32 <= n<=32, n!= 0 name = Name des Unterprogramms
	ON COMBIT bezieht sich auf die ersten 32 Bit des Prozessdatenspeichers.
Beschreibung	Mit der Anweisung ON COMBIT wird ein Unterprogramm aufgerufen, wenn Bit n des Kommunikationspuffers gesetzt wird.
	ACHTUNG!:
	<ul style="list-style-type: none"> - Das aufzurufende Unterprogramm muss innerhalb des durch SUBMAINPROG und ENDPROG gekennzeichneten Programmbereichs definiert sein. - Während der Ausführung von Unterprogrammen, die durch einen Interrupt ausgelöst wurden, ist automatisch NOWAIT ON gesetzt.
Priorität	Sollten mehrere Interrupts gleichzeitig auftreten, wird zuerst das dem niedrigsten Bit zugeordnete Unterprogramm abgearbeitet. Die anderen Interrupts werden anschließend abgearbeitet. Tritt während eines Interrupt-Unterprogramms der gleiche Interrupt auf (Ausnahme: Fehler-Interrupt), wird dieser ignoriert und geht verloren.
Kompatibilität	Bei COMOPTGET und COMOPTSEND wird der Offset von 2 Worten aus Kompatibilitätsgründen beibehalten.
Befehlsgruppe	INT
Querverweise	SUBPROG .. RETURN, COMOPTGET, COMOPTSEND, Prioritäten bei Interrupts, NOWAIT
Syntax-Beispiel	ON COMBIT 5 GOSUB test // Interrupt auf Bit 5 vom Feldbus setzen

□ ON DELETE .. GOSUB

Kurzinfo	Löscht einen Positions-Interrupt.
Syntax	ON DELETE pos GOSUB name
Parameter	pos = Wert name = Name des Unterprogramms
Beschreibung	<p>Der Befehl kann genutzt werden, um einen ON APOS Interrupt zu löschen, der zuvor wie folgt definiert wurde:</p> <p style="padding-left: 40px;">ON sign APOS xxx GOSUB name</p> <p>Der Parameter 'pos' kann jeden Wert annehmen, z.B. 0. Der Wert wird nicht geprüft und hat keine Bedeutung für das Löschen des Interrupts.</p> <p>Die Hauptbedeutung kommt dem Parameter 'name' zu, der den Namen des Unterprogramms enthalten muss, das vorher im ON APOS Befehl definiert worden ist. Daher löscht</p> <p style="padding-left: 40px;">ON DELETE pos GOSUB name</p> <p>jeden (!) Positions-Interrupt, der zum Unterprogramm gehört und durch den Namen erkannt wird. Siehe Beispiel 1.</p>

__ Befehlsreferenz __

**ACHTUNG!:**

Es werden nur Positions-Interrupts gelöscht, keine anderen Interrupt-Typen.

Umleiten eines
ON .. APOS .. GOSUB
Befehls

Ein Positions-Interrupt kann in ein anderes Unterprogramm umgeleitet werden. Dies definiert keinen neuen Interrupt, sondern modifiziert nur das Unterprogramm, das im Fall einer Interrupt-Erkennung ausgeführt werden muss.

Die Befehlsyntax ist die gleiche wie für den ON APOS Befehl:

ON sign APOS xxx GOSUB newname

Die Parameter 'sign' und 'xxx' müssen genau die gleichen sein, wie in der ursprünglichen Definition. Die Position, die es betrifft wird durch diese zwei Parameter identifiziert. Der Parameter 'newname' muss den aktualisierten Namen des Unterprogramms enthalten, das aufgerufen werden soll, wenn der Interrupt eintritt. Siehe Beispiel 2.

**ACHTUNG!:**

Es können nur Positions-Interrupts umgeleitet werden, keine anderen Interrupt-Typen.

Befehlsgruppe

INT

Querverweise

ON posint GOSUB, ON INT ..

Programmbeispiel 1

```
ON - APOS 20000      GOSUB hitinfo      // Interrupt #1
ON - APOS 10000     GOSUB hitinfo      // Interrupt #2
ON + APOS 10000     GOSUB hitinfo      // Interrupt #3
ON + APOS 0         GOSUB hitzero      // Interrupt #4
ON - APOS 0         GOSUB hitzero      // Interrupt #5
ON INT 3            GOSUB hitinfo      // Interrupt #6
```

...

ON DELETE 0 GOSUB hitinfo

...

ON + APOS 99999 GOSUB hitinfo // Neuer definierter Positions-Interrupt

Ergebnis:

Alle Positions-Interrupts (#1, #2, #3), die zu dem Unterprogramm 'hitinfo' gehören, werden gelöscht sobald 'ON DELETE 0 GOSUB hitinfo' ausgeführt wird. Diese Interrupts zählen nicht mehr für die maximale Anzahl der verfügbaren Interrupts und können daher auch nicht mehr freigegeben oder gesperrt werden. Alle anderen nicht-Positions-Interrupts, sogar diejenigen, die zum gleichen Unterprogramm gehören (z.B. ON INT 3) sind weiterhin gültig!

Sobald die Befehlszeile 'ON + APOS 99999 GOSUB hitinfo' ausgeführt wird, definiert dies einen neuen Positions-Interrupt, der zu dem Unterprogramm verweist, das bereits früher in Gebrauch war.

Programmbeispiel 2

```
ON - APOS 10000 GOSUB hitinfo      // Interrupt #1
ON + APOS 10000 GOSUB hitinfo      // Interrupt #2
...
ON + APOS 10000 GOSUB hitposdir    // Interrupt #2 umleiten
```

Ergebnis:

Sobald die zweite Definition des 'ON + APOS 10000 ...' Befehls ausgeführt ist, wird der Interrupt #2 zu einem neu definierten Unterprogramm 'hitposdir' umgeleitet. Es ist nach wie vor der gleiche Interrupt (d.h. kein weiterer), der nun ein anderes Unterprogramm aufruft. Die „alte“ Definition des Interrupts #1 'ON - APOS 10000 GOSUB hitinfo' ist weiterhin ohne jede Veränderung gültig.

□ ON DELETE .. SETOUT

Kurzinfo	Löscht alle Interrupts, die einen Ausgang setzen oder zurücksetzen.
Syntax	ON DELETE sign inttype SETOUT outno
Parameter	sign + = steigende Flanke - = fallende Flanke inttype = APOS IPOS MAPOS MCPOS MIPOS outno = Ausgangsnummer
Beschreibung	Der Befehl löscht alle Interrupts, die den Ausgang <i>outno</i> setzen oder zurücksetzen. ACHTUNG! Wenn der Ausgang <i>outno</i> im Befehl positiv ist, werden nur die Interrupts gelöscht, bei denen <i>outno</i> gesetzt ist. Wenn <i>outno</i> negativ ist, werden nur die Interrupts gelöscht, bei denen der Ausgang zurückgesetzt ist. Wenn Sie also beide Arten der Interrupt-Definition einsetzen, müssen Sie daher immer zwei Befehle verwenden.
	
Portabilität	Der Befehl ist ab MCO 5.00 verfügbar.
Befehlsgruppe	INT
Querverweise	ON INT .. GOSUB, ON posint GOSUB
Syntax-Beispiel	<pre> SET SYNCMPULSS 20000 // Abstand zwischen zwei Markern ON +ipos 500 SETOUT 1 ON +ipos 1000 SETOUT -1 ... (Programm) ... ON DELETE 0 SETOUT 1 ON DELETE 0 SETOUT -1 </pre>



□ ON ERROR GOSUB

Kurzinfo	Definition eines Unterprogramms für den Fehlerfall.
Syntax	ON ERROR GOSUB name
Parameter	name = Name des Unterprogramms
Beschreibung	<p>Mit der ON ERROR GOSUB Anweisung wird ein Unterprogramm definiert, das bei einem Fehler aufgerufen wird. Tritt nach dieser Definition zu einem beliebigen Zeitpunkt ein Fehler auf, wird das Programm nicht automatisch abgebrochen, sondern das definierte Unterprogramm aufgerufen.</p> <p>Innerhalb dieses Unterprogramms ist es dann möglich, gezielt auf den Fehler zu reagieren, auf Benutzereingriffe zu warten, die Fehlermeldung mit ERRCLR zu löschen oder bei nicht behebbaren Fehlern mit der EXIT Anweisung das Programm abzubrechen.</p> <p>Wird das Programm nicht abgebrochen, dann wird nach dem RETURN Befehl an der vor dem Aufruf des Fehlerunterprogramms bearbeiteten Programmposition fortgefahren.</p> <p>Mit dem CONTINUE Befehl kann man den durch den Fehler unterbrochenen Bewegungsablauf fortsetzen (Ausnahme: Synchronisationsbefehle)</p> <p>ACHTUNG! Die ON ERROR GOSUB Anweisung sollte am Programmanfang stehen, damit sie für das ganze Programm Gültigkeit besitzt.</p> <p>Das aufzurufende Unterprogramm muss innerhalb des durch SUBMAINPROG und ENDPROG gekennzeichneten Programmbereichs definiert sein.</p> <p>Das Erkennen eines Interrupts und der Aufruf des entsprechenden Unterprogramms benötigen maximal 2 Millisekunden.</p> <p>ACHTUNG! – Fehlerunterprogramme können nicht durch andere Interrupts unterbrochen werden. – Während der Ausführung eines Fehlerunterprogramms ist automatisch NOWAIT ON gesetzt.</p> <p>Wird das Fehlerunterprogramm mit einem Fehler verlassen, weil zum Beispiel kein ERRCLR durchgeführt wurde oder bereits ein neuer Fehler aufgetreten ist, erfolgt ein erneuter Aufruf.</p> <p>ACHTUNG! Die ON ERROR GOSUB Anweisung beendet nicht HOME und INDEX Befehle. Das heißt, diese werden zu Ende ausgeführt, sobald der Fehler gelöscht ist. Um dies zu verhindern, kann man ein ON TIME 1 in die ERROR Anweisung einfügen.</p>
Befehlsgruppe	INT
Querverweise	SUBPROG...RETURN, ERRCLR, ERRNO, CONTINUE, EXIT, Prioritäten bei Interrupts, ON TIME, NOWAIT
Syntax-Beispiel	<pre>ON ERROR GOSUB errhandle /* Definition eines Fehlerunterprogramms */ Befehlszeilen 1 ... n SUBMAINPROG /* Unterprogramm 'errhandle' muss definiert sein */ SUBPROG errhandle Befehlszeilen 1 ... n RETURN ENDPROG</pre>
Programmbeispiel	ERROR_01.M, IF_01.M, INDEX_01.M

□ ON INT .. GOSUB

Kurzinfo	Definition eines Interrupt-Eingangs.
Syntax	ON INT n GOSUB name
Parameter	<p>n = Nummer des zu überwachenden Eingangs, Reaktion auf steigende Flanke (Eingabebereich 1 ... 8 und FC 300 Eingänge 18 ... 33)</p> <p>-n = Nummer des zu überwachenden Eingangs, Reaktion auf fallende Flanke (Eingabebereich -8 .. -1 und FC 300 Eingänge -33 ... -18)</p> <p>= bzw. bei CAN-Anwendungen Modulnummer * 256 + I/O-Nummer</p> <p>name = Name des Unterprogramms</p>
Beschreibung	<p>Mit der Anweisung ON INT GOSUB wird ein Unterprogramm definiert, das aufgerufen wird, wenn an dem überwachten Eingang eine Pegeländerung auftritt. Pro Eingang kann maximal ein Unterprogramm definiert werden.</p> <p>Der Befehl ON INT erlaubt die Zuweisung eines positiven Interrupts <u>und</u> eines negativen Interrupt für einen Eingang <u>zur gleichen Zeit</u>:</p> <pre>ON INT 1 GOSUB posedge ON INT -1 GOSUB negedge</pre> <p>Die kann zu einem beliebigen Zeitpunkt definiert werden. Tritt nach dieser Definition der entsprechende Interrupt auf, wird das dazu gehörende Unterprogramm aufgerufen und abgearbeitet. Nach dem letzten Unterprogrammbefehl (RETURN) wird das Programm an der vor dem Interrupt bearbeiteten Programmposition fortgesetzt.</p> <p>Wenn Interrupt-Funktionen auf die CAN-Module gesetzt werden, müssen diese zuvor mit CANINI. initialisiert werden.</p> <p>ACHTUNG!:</p> <p>Die Anweisung ON INT GOSUB sollte am Programmstart stehen, damit sie für das ganze Programm Gültigkeit besitzt.</p> <p>Das aufzurufende Unterprogramm muss innerhalb des durch SUBMAINPROG und ENDPROG gekennzeichneten Programmbereichs definiert sein.</p> <p>Das Erkennen eines Interrupts und der Aufruf des entsprechenden Unterprogramms benötigen maximal 2 Millisekunden. Ein Interrupt vom Eingang FC 300 addiert im schlimmsten Fall zusätzlich 2 ms.</p> <p>Es ist eine minimale Signallänge von 1 ms notwendig, um eine Pegeländerung sicher zu erkennen. Informieren Sie sich bitte in den MCO 305 und FC 300 Produkthandbüchern über die Beschaltung und technischen Daten der Eingänge.</p> <p>ACHTUNG!:</p> <ul style="list-style-type: none"> - Die Anweisung ON INT GOSUB ist flanken- und nicht pegelgesteuert. - Während der Ausführung von Unterprogrammen, die durch einen Interrupt ausgelöst wurden, ist automatisch NOWAIT ON gesetzt. <p>Priorität Sollten mehrere Interrupts gleichzeitig auftreten, wird zuerst das, dem kleinsten Eingang zugeordnete Unterprogramm abgearbeitet. Die anderen Interrupts werden anschließend abgearbeitet.</p> <p>Tritt während eines Interrupt-Unterprogramms der gleiche Interrupt (Ausnahme: Fehler-Interrupt) auf, wird dieser ignoriert und geht verloren.</p> <p>Portabilität Zuweisung eines positiven <u>und</u> negativen Interrupts für einen Eingang zur gleichen Zeit ab MCO 5.00.</p>



__ Befehlsreferenz __

**ACHTUNG!**

Die CAN-Befehle arbeiten mit den vordefinierten PDOs von CAN-OPEN. Ändern Sie auf keinen Fall diese Default-Einstellung (minimum capability device), denn dann würden die CAN-Befehle nicht mehr funktionieren.

Befehlsgruppe INT**Querverweise** SUBPROG..RETURN, ON ERROR .. GOSUB, WAITI, DISABLE interrupts, ENABLE interrupts, Prioritäten bei Interrupts, NOWAIT, CANINI

Syntax-Beispiel

```
ON INT 4 GOSUB posin      /* Definition von Eingang 4 (positive Flanke) */
ON INT -5 GOSUB negin     /* Definition von Eingang 5 (negative Flanke) */
Befehlszeile 1
Befehlszeile n
SUBMAINPROG              /* Unterprogramme müssen definiert sein */
    SUBPROG posin
        Befehlszeile 1
        Befehlszeile n
    RETURN
    SUBPROG negin
        Befehlszeilen 1 ... n
    RETURN
ENDPROG
```

Programmbeispiel ONINT_01.M, DELAY_01.M**□ ON KEYPRESSED GOSUB****Summary** Interrupt wenn eine Taste gedrückt oder losgelassen wird.**Syntax** ON KEYPRESSED GOSUB name**Parameter** name = Name des Unterprogramms**Description** Mit der Anweisung ON KEYPRESSED kann man reagieren, wenn eine Taste des LCP Panels gedrückt oder losgelassen wird.**ACHTUNG!:**

- Das aufzurufende Unterprogramm muss innerhalb des durch SUBMAINPROG und ENDPROG gekennzeichneten Programmbereichs definiert sein.
- Während der Ausführung von Unterprogrammen, die durch einen Interrupt ausgelöst wurden, ist automatisch NOWAIT ON gesetzt.

Command Group INT**Cross Index** SUBPROG .. RETURN, INKEY

Syntax Example

```
ON KEYPRESSED GOSUB keyhandler
WHILE(1) DO          // Endlosschleife
ENDWHILE
////////////////////
SUBMAINPROG
SUBPROG keyhandler
    key = INKEY(-1)  // nicht auf Tastenbetätigung warten

    PRINT key
RETURN
ENDPROG
```

□ ON PARAM .. GOSUB

Kurzinfo	Unterprogramm aufrufen, wenn ein Parameter von außen geändert wird.
Syntax	ON PARAM n GOSUB name
Parameter	n = Parameternummer name = Name des Unterprogramms
Beschreibung	<p>Mit der Anweisung ON PARAM kann man reagieren, wenn ein Parameter im LCP Display geändert wurde und ein Unterprogramm aufrufen.</p> <p>Alle Parameter (32-xx, 33-xx) und alle Anwendungsparameter (19-xx) sowie allgemeine Parameter (z.B. 8-02, 5-00) können dazu benutzt werden.</p> <p>ON PARAM für Array-Elemente (z.B. ON PARAM 0x01210005) funktioniert auch, wenn das Array in ein erhaltenes PDO mit einem LINKPDO Befehl abgebildet wird, aber es ist begrenzt. Nur ein Array-Element kann in einem ON PARAM per verlinktem Array benutzt werden.</p> <p>Beispiel:</p> <pre>DIM test[100] LINKPDO 1 320 0x01210005 0 // die ersten 10 Longs des PDOs werden in die // ersten 10 Elemente des Arrays test verlinkt ON PARAM 0x01210007 GOSUB test // wenn sich das dritte Element des Arrays test ändert</pre> <p>Wenn ein weiterer Befehl ON PARAM hinzukommt, wie</p> <pre>ON PARAM 0x01210009 GOSUB testsub // wenn sich das 5. Element des Arrays test ändert</pre> <p>wird der erste ON PARAM Befehl überschrieben, weil nur einer per LINK-Array gehandhabt werden kann.</p> <p>Der ON PARAM Befehl wird aktiviert, sobald das Arrayelement durch ein ankommendes PDO oder durch ein SDOWRITE in das SDO 0x012100ss mit dem korrekten Subindex geschrieben wird.</p> <p>ON PARAM wird nicht aktiviert, wenn das Array in das Programm durch test[nn] = value geschrieben wird. Und es wird auch nicht aktiviert, wenn das komplette Array mittels 0x23FF SDO überschrieben wird.</p>
	 <ul style="list-style-type: none"> - ACHTUNG! Es sind maximal 10 ON PARAM Funktionen möglich. - Das aufzurufende Unterprogramm muss innerhalb des durch SUBMAINPROG und ENDPROG gekennzeichneten Programmbereichs definiert sein. - Während der Ausführung von Unterprogrammen, die durch einen Interrupt ausgelöst wurden, ist automatisch NOWAIT ON gesetzt.
Portabilität	Der Befehl ist ab MCO 5.00 verfügbar.
Befehlsgruppe	INT
Querverweise	SUBPROG .. RETURN, SYSVAR, LINKPDO
Syntax-Beispiel 1	<pre>ON PARAM 3267 GOSUB poserr // wenn Schleppfehler geändert wurde SUBMAINPROG SUBPROG poserr PRINT "Neuer Schleppfehler: ", GET POSERR RETURN</pre>

__ Befehlsreferenz __



Syntax-Beispiel 2 // Interrupt auslösen bei jeder Änderung der CANopen DS402 "Betriebsart"
ON PARAM 0x01606000 GOSUB OpModeUpdate
SUBMAINPROG
SUBPROG OpModeUpdate
 PRINT "Neuer DS402 Betriebsart: ",sysvar[0x01606000]
 // Befehlsabfolge in Abhängigkeit der neuen Betriebsart auslösen
 SWITCH (sysvar[0x01604000])
 CASE 1:
 PRINT "Profil Positionierung Modus"
 // Aktion
 BREAK
 CASE 2:
 PRINT "Geschwindigkeitsmodus"
 // Aktion
 BREAK
 // CASE
 // ...
 // BREAK
 DEFAULT:
 PRINT "Betriebsart wird nicht unterstützt"
 ENDSWITCH
RETURN
ENDPROG

Syntax-Beispiel 3 // Das RxPDO mit dem Benutzerparameter 1 verknüpfen ...
LINKPDO 1 32 0x01220101 0
// ... und bei jeder Änderung einen Interrupt auslösen
ON PARAM 0x01220101 GOSUB UserParamUpdate
SUBMAINPROG
SUBPROG UserParamUpdate
 PRINT "Benutzerparameter 1 updated: ",sysvar[0x01220101]
RETURN
ENDPROG

□ ON PERIOD

Kurzinfo	Ruft ein Unterprogramm in regelmäßigen Abständen auf.
Syntax	ON PERIOD n GOSUB name
Parameter	<p>n > 20 ms = Zeit in ms, nach der das Programm immer wieder aufgerufen wird (maximal MLONG)</p> <p>n = 0 = Funktion abschalten</p> <p>name = Name des Unterprogramms</p>
Beschreibung	<p>Mit ON PERIOD kann man ein Unterprogramm in regelmäßigen Abständen (zeitgeführt) aufrufen. ON PERIOD wirkt wie ein Interrupt und wird alle 20 ms überprüft.</p> <p>ACHTUNG!:</p> <ul style="list-style-type: none"> - Die Genauigkeit mit der die Zeit eingehalten wird, hängt vom restlichen Programm ab. Typischerweise beträgt die Genauigkeit ± 1 ms. - Das aufzurufende Unterprogramm muss innerhalb des durch SUBMAINPROG und ENDPORG gekennzeichneten Programmbereichs definiert sein. - Während der Ausführung eines ON PERIOD Unterprogramms ist automatisch NOWAIT ON gesetzt.
Befehlsgruppe	INT
Querverweise	ON TIME, GOSUB, DISABLE interrupts, ENABLE interrupts, Prioritäten bei Interrupts




□ ON posint .. GOSUB
Kurzinfo Unterprogramm aufrufen, wenn ein Positions-Interrupt auftritt.

Syntax ON sign postype position GOSUB name

Parameter sign + = steigende Flanke oder wenn die Position in positiver Richtung passiert wurde

- = fallende Flanke oder wenn die Position in negativer Richtung passiert wurde

$$100 \quad \underline{\hspace{2cm}} \quad xxx \quad 0 \quad \underline{\hspace{2cm}}$$

<----- positive Richtung

-----> negative Richtung

postype = APOS

IPOS

MAPOS

MCPOS

MIPOS

position= abhängig vom Befehl in Benutzereinheiten [BE], oder Master Benutzereinheiten [MU] oder Kurveneinheiten [CU]

name = Name des Unterprogramms

Beschreibung Wenn ein Befehl ON xPOS benutzt wird und eine Position liegt hinter einem Überlauf des Encoders, wird dies intern automatisch gehandhabt.

Wenn z.B. keine POSFACTs gesetzt sind:

```
testpos = 0x3FFFFFF0 // Position kurz vor Overflow
```

```
newpos = testpos + 200 // neue Testposition
```

```
ON +APOS newPos GOSUB myprog
```

Die neue Testposition wird intern korrekt in die Interrupt-Liste eingetragen.

Das gleiche gilt wenn POSFACT_Z und POSFACT_N gesetzt sind und der Benutzerwert einen Überlauf der internen qc Positionen verursacht.

Alle oder einzelne Positions-Interrupts können mit dem Befehl ON DELETE .. GOSUB gelöscht werden.


ACHTUNG!

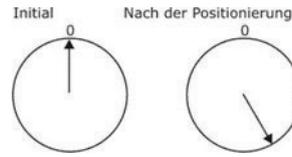
- Das aufzurufende Unterprogramm muss innerhalb eines Programmbereichs SUBMAINPROG und ENDPROG stehen.
- Während der Ausführung von Unterprogrammen, die durch einen Interrupt ausgelöst wurden, ist automatisch NOWAIT ON gesetzt.

ON APOS .. GOSUB Mit der Anweisung ON APOS kann man ein Unterprogramm aufrufen, wenn eine bestimmte Slave-Position xxx (BE) in positiver bzw. negativer Richtung passiert wurde. Die Anweisung kann für Positionier- und Synchronisiersteuerungen wie auch für Kurvenscheibensteuerungen und Nockenschaltwerke nützlich sein. Zum Beispiel um bei offenen Kurven die anwachsende Slave-Position nach jedem Zyklus durch einen wiederkehrenden Bezugspunkt zu ersetzen.

__ Befehlsreferenz __

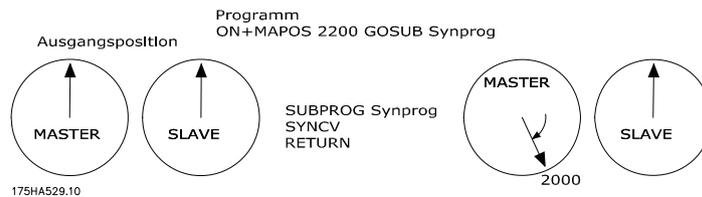
Beispiel:

```
CSTART
ON +APOS 2000 GOSUB STOP
SUBMAINPROG
    SUBPROG STOP
    CSTOP
    RETURN
ENDPROG
```



Gemäß dem Programm stoppt der Antrieb sobald er die Position 2000 erreicht hat. Dieser Positions-Interrupt betrachtet die Distanz zwischen der letzten Markerposition und der aktuellen Position. Dabei ist es wichtig, dass SYNCMPULSS korrekt gesetzt ist. Diese Information wird für die Ermittlung eines Überlaufs sowie für das Rückwärtsfahren verwendet. Beim Rückwärtsfahren wird (SYNCMPULSS + (APOS - IPOS)) benutzt, statt of (APOS - IPOS).

ON MAPOS .. GOSUB Unterprogramm aufrufen, wenn die Master-Position xxx (MU) in positiver oder negativer Richtung passiert wurde. Zum Beispiel um bei einem Linearantrieb (Slave) mit einem Verfahrbereich von 0 bis 10000 BE an einer beliebigen Position einen Ausgang zu setzen.



Programmgemäß startet die Geschwindigkeits-Synchronisation, nachdem der Master die Position 2200 qc in positiver Richtung erreicht hat. Dann fahren der Slave und der Master mit SYNCV.

ON MCPOS .. GOSUB Unterprogramm aufrufen, wenn die Master-Position xxx (MU) passiert wurde. Mit dieser für Kurvenscheibensteuerungen typischen Anweisung ON MCPOS kann man ein Unterprogramm aufrufen, wenn eine bestimmte Master-Position (MU) in positiver bzw. negativer Richtung passiert wurde. Auf diese Weise lassen sich nicht nur Nockenschaltwerke realisieren, sondern auch viel komplexere Aufgaben durchführen. Zum Beispiel könnte man abhängig von der Position online Parameter ändern.



ACHTUNG!

Vor dem Befehl ON MCPOS .. GOSUB muss immer ein DEFMCPOS oder ein SETCURVE stehen, weil sonst die Kurvenposition nicht bekannt ist.

ON MIPOS .. GOSUB Unterprogramm aufrufen, wenn die Distanz zwischen zwei Markern erreicht ist. Dieser Positions-Interrupt betrachtet die Distanz zwischen der letzten Markerposition und der aktuellen Position. Es ist wichtig, dass SYNCMPULSM korrekt gesetzt ist. Diese Information wird für die Ermittlung eines Überlaufs sowie für das Rückwärtsfahren verwendet. Beim Rückwärtsfahren wird (SYNCMPULSM + (MAPOS - MIPOS)) statt (MAPOS - MIPOS) benutzt.

Portabilität ON POSINT Overflow-Handhabung ab MCO 5.00.

Befehlsgruppe INT

Querverweise SUBPROG .. RETURN, DISABLE .., ENABLE .., Prioritäten der Interrupts, ON DELETE .. GOSUB, APOS, MAPOS, MIPOS, IPOS

__ Befehlsreferenz __

Syntax-Beispiel 1 ON -apos 800 GOSUB name
 // Unterprogramm 'name' aufrufen, wenn die
 // Slave-Position 800 in negativer Richtung passiert wurde

Syntax Beispiel 2 SET SYNCMPULSS 20000 // Abstand zwischen zwei Markern
 ON +ipos 5000 GOSUB prog1
 ON +ipos 15000 GOSUB prog2

In diesem Beispiel haben zwei Marker einen Abstand von 20000 qc. Angenommen, der erste Marker ist auf Position 0. Dann wird prog1 bei 5000, 25000, 45000 usw. aufgerufen und prog2 wird bei 15000, 35000 usw. ausgeführt.

Syntax Beispiel 3 ON +mapos 1200 GOSUB name
 // Subprogramm immer auf Position 1200 aufrufen

Syntax Beispiel 4 SET SYNCMPULSM 20000 // Abstand zwischen zwei Markern
 ON +mipos 5000 GOSUB prog1
 ON +mipos 15000 GOSUB prog2

In diesem Beispiele habe zwei Marker einen Abstand von 20000 qc. Angenommen der erste Marker ist auf Position 0. Dann wird prog1 bei 5000, 25000, 45000 usw. aufgerufen und prog2 wird bei 15000, 35000 usw. ausgeführt.

Syntax Beispiel 5 Auf einem Band werden Kartons in unregelmäßigen Abständen transportiert. Durch Setzen eines Ausgangs wird der Slave immer dann gestartet, wenn die Position xxx erreicht wird.

```
SUBMAINPROG // Unterprogramm output setzen
  SUBPROG output
    OUT 3 1 // 03 on
  RETURN
ENDPROG
ON +MCPOS 4500 GOSUB output
// Unterprogramm output immer an Position 4500 aufrufen
```

□ ON posint .. SETOUT (TOIN)

Kurzinfo	Simuliert ein Nockenschaltwerk (CAM-Box) (alle POSINT Typen)
Syntax	ON +/- type position SETOUT outno ON +/- type position SETOUT outno TOIN inno
Parameter	<p>type = alle POSINT APOS IPOS MAPOS MCPOS MIPOS</p> <p>position = abhängig vom Befehl in Benutzereinheiten [BE], Master-Benutzereinheiten [MU] oder Kurveneinheiten [CU]</p> <p>outno = kann jede gültige Nummer eines Ausgangs sein (oder negative Nummer des Ausgangs)</p> <p>inno = kann jede gültige Nummer eines Eingangs sein (oder negative Nummer des Eingangs)</p>
Beschreibung	<p>Alle Funktionen des Positions-Interrupts können diese Simulation eines Nockenschaltwerks (CAM-Box) benutzen. Dies ist mit allen Arten der POSINTs möglich. Im ersten Fall ist der Ausgang <i>outno</i> entweder gesetzt oder zurückgesetzt; dies hängt davon ab, ob <i>outno</i> positiv oder negativ ist.</p> <p>Im zweiten Fall ist der Ausgang auf den Wert des Eingangs <i>inno</i> gesetzt. (Oder auf den Eingang mit Negierung des Wertes, falls entweder <i>inno</i> oder <i>outno</i> negativ ist.). Wenn beide, <i>outno</i> und <i>inno</i> negativ sind, ist es das gleiche als wären beide positiv.</p> <p>Der Vorteil solcher Befehle ist, dass sie im Hintergrund arbeiten und daher das Anwendungsprogramm nicht unterbrechen. Sie sind auch schneller als wenn Unterprogramme aufgerufen werden müssten, die dann die Ausgänge setzen. Typische Reaktionszeit ist unter 1 ms.</p>
Portabilität	Der Befehl ist ab MCO 5.00 verfügbar.
Befehlsgruppe	INT
Querverweise	SUBPROG .. RETURN, APOS, IPOS, MAPOS, MIPOS,
Syntax-Beispiel	<pre>SET SYNCMPULSS 20000 // Abstand zwischen zwei Markern on +ipos 500 setout 1 toin 2 on +ipos 1000 setout -1</pre> <p>In diesem Beispiel wird der Ausgang 1 auf den Wert des Eingangs 2 auf der Position von 500 Benutzereinheiten nach dem Marker gesetzt.</p> <p>Dann wird der Ausgang 1 bei der Position von 1000 qc nach dem Marker wieder auf 0 gesetzt.</p>




□ ON STATBIT .. GOSUB

Kurzinfo	Unterprogramm aufrufen, wenn Bit n des Statuswortes gesetzt wird.																						
Syntax	ON STATBIT n GOSUB name																						
Parameter	<p>n = Bit n des Statuswortes</p> <p>Byte 1 + 2 Statuswort des FC 300 (siehe FC3xx Handbuch)</p> <p>Byte 3</p> <table border="0"> <tr><td>Bit 17</td><td>1 = Achse fährt</td></tr> <tr><td>Bit 18</td><td>1 = Überlauf Slave-Drehgeber</td></tr> <tr><td>Bit 19</td><td>1 = Überlauf Master-Drehgeber</td></tr> <tr><td>Bit 20</td><td>1 = FC 300 temporär abgeschaltet *)</td></tr> </table> <p>Byte 4 SYNCSTAT</p> <table border="0"> <tr><td>Bit 25</td><td>1 = SYNCREADY</td></tr> <tr><td>Bit 26</td><td>1 = SYNCFAULT</td></tr> <tr><td>Bit 27</td><td>1 = SYNCACCURACY</td></tr> <tr><td>Bit 28</td><td>1 = SYNCMMHIT</td></tr> <tr><td>Bit 29</td><td>1 = SYNCSTMHIT</td></tr> <tr><td>Bit 30</td><td>1 = SYNCMMERR</td></tr> <tr><td>Bit 31</td><td>1 = SYNCSTMERR</td></tr> </table> <p>name = Name des Unterprogramms</p> <p>*) Erläuterung: d.h. die Achse befindet sich innerhalb des Toleranzbereiches des Regelfensters Par. 32-71 REGWMAX / Par. 32-72 REGWMIN. Der FC 300 wird wieder eingeschaltet, sobald das Regelfenster verlassen wird.</p>	Bit 17	1 = Achse fährt	Bit 18	1 = Überlauf Slave-Drehgeber	Bit 19	1 = Überlauf Master-Drehgeber	Bit 20	1 = FC 300 temporär abgeschaltet *)	Bit 25	1 = SYNCREADY	Bit 26	1 = SYNCFAULT	Bit 27	1 = SYNCACCURACY	Bit 28	1 = SYNCMMHIT	Bit 29	1 = SYNCSTMHIT	Bit 30	1 = SYNCMMERR	Bit 31	1 = SYNCSTMERR
Bit 17	1 = Achse fährt																						
Bit 18	1 = Überlauf Slave-Drehgeber																						
Bit 19	1 = Überlauf Master-Drehgeber																						
Bit 20	1 = FC 300 temporär abgeschaltet *)																						
Bit 25	1 = SYNCREADY																						
Bit 26	1 = SYNCFAULT																						
Bit 27	1 = SYNCACCURACY																						
Bit 28	1 = SYNCMMHIT																						
Bit 29	1 = SYNCSTMHIT																						
Bit 30	1 = SYNCMMERR																						
Bit 31	1 = SYNCSTMERR																						
Beschreibung	Mit der Anweisung ON STATBIT wird ein Unterprogramm aufgerufen, wenn Bit n des FC 300 Status gesetzt ist. Diese 32 Bits des FC 300 Status setzen sich aus dem Statuswort des FC 300, dem Byte 3 des internen Status (zum Beispiel „Achse fährt“) und dem Bit n von SYNCSTAT zusammen																						
	 <p>ACHTUNG!:</p> <ul style="list-style-type: none"> – Das aufzurufende Unterprogramm muss innerhalb des durch SUBMAINPROG und ENDPROG gekennzeichneten Programmbereichs definiert sein. – Während der Ausführung von Unterprogrammen, die durch einen Interrupt ausgelöst wurden, ist automatisch NOWAIT ON gesetzt. 																						
Priorität	Sollten mehrere Interrupts gleichzeitig auftreten, wird zuerst das dem niedrigsten Bit zugeordnete Unterprogramm abgearbeitet. Die anderen Interrupts werden anschließend abgearbeitet. Tritt während eines Interrupt-Unterprogramms der gleiche Interrupt auf (Ausnahme: Fehler-Interrupt), wird dieser ignoriert und geht verloren.																						
Befehlsgruppe	INT																						
Querverweise	SUBPROG ..RETURN, DISABLE interrupts, ENABLE interrupts, Prioritäten der Interrupts																						
Syntax-Beispiel	<pre>ON STATBIT 30 GOSUB markererror /* Interrupt, wenn Fehler-Flag Master */ SUBMAINPROG SUBPROG markererror SYNCSTATCLR 32 /* Fehler-Flag SYNCMMERR löschen */ /* Wert 32 von Parameter SYNCSTATCLR, nicht Bit-Nummer! */ RETURN ENDPROG</pre>																						

□ ON TIME

Kurzinfo	Einmalauf Ruf eines Unterprogramms.	
Syntax	ON TIME n GOSUB name	
Parameter	n	= Zeit in ms, nach der das Unterprogramm aufgerufen wird (maximal MLONG)
	name	= Name des Unterprogramms
Beschreibung	Nach Ablauf der gesetzten Zeit wird das entsprechende Unterprogramm einmal aufgerufen. Das Programm läuft in der Zwischenzeit normal weiter.	
	ACHTUNG!:	
	-	Die Genauigkeit mit der die Zeit eingehalten wird, hängt von der eingesetzten Hardware und vom restlichen Programm ab. Typischerweise beträgt die Genauigkeit ± 1 ms.
	-	Das aufzurufende Unterprogramm muss innerhalb des durch SUBMAINPROG und ENDPROG gekennzeichneten Programmbereichs definiert sein.
	-	Während der Ausführung eines ON TIME Unterprogramms ist automatisch NOWAIT ON gesetzt.
Befehlsgruppe	INT	
Querverweise	ON PERIOD, GOSUB, DISABLE interrupts, ENABLE interrupts, Prioritäten der Interrupts	
Syntax-Beispiel	<pre> OUT 1 1 /* Lampe an */ ON TIME 200 GOSUB off1 /* Lampe nach 200 ms wieder aus */ SUBMAINPROG SUBPROG off1 OUT 1 0 RETURN ENDPROG </pre>	




OUT

Kurzinfo	Digitale Ausgänge setzen oder zurücksetzen.
Syntax	OUT n s oder OUT X/n s
Parameter	<p>n = Nummer des Ausgangs MCO 305: 1 – 8 (6) FC 301 outputs: 27 Relais Ausgänge: 21 (Relais 1) und 22 (Relais 2); Achtung: FC 301 < 11 kW hat nur Relais 1. MCB 105, Relais Ausgänge: X34/1 (Relais 7), X34/5 (Relais 8) und X34/10 (Relais 9).</p> <p>bzw. für CANopen I/O-Module: CAN-Bus + (Modul-CAN-ID * 256) + Ausgangsnummer (bzw. Ausgangsbyte)</p> <p>X/n = Klemmenblock / Pin Nummer</p> <p>s = Zustand 0 = OFF 1 = ON</p>
Beschreibung	<p>Der OUT Befehl kann die 8 (6) digitalen Ausgänge der MCO 305 Option, die digitalen und Relaisausgänge des FC 300 und die Relaisausgänge des MCB 105 setzen oder zurücksetzen.</p> <p>Der Modus für die Ausgänge 7,8 wird in Par. 33-60 IOMODE ausgewählt.</p> <p>Ob die Ausgänge NPN- oder PNP-schaltend benutzt werden, hängt von der Auswahl der Standardausgänge des FC 300 ab, die in Par. 5-00 gesetzt werden.</p> <p>CAN-Module, die die CANopen Spezifikationen erfüllen, können Sie ebenfalls mit dem OUT Befehl ansprechen, und zwar über die entsprechende Nummer, die wie folgt definiert ist:</p> <p style="padding-left: 40px;">CAN-Bus + (Modul-CAN-ID * 256) + Ausgangsnummer (bzw. Ausgangsbyte)</p> <p>Beim Ausführen eines solchen Befehls werden temporär die entsprechenden CAN-Objekte angelegt, ausgewertet und anschließend wieder freigegeben; daher können Sie beliebig viele Module ansprechen.</p> <p> ACHTUNG! Wenn eine falsche Kombination oder Pin-Nummer für X/n benutzt wird, die nicht gesetzt werden kann, wird Fehler 171 gemeldet. Aber es gibt keine Überprüfung, falls ein Eingang statt eines Ausgangs oder umgekehrt benutzt wird.</p> <p> ACHTUNG!</p> <ul style="list-style-type: none"> - Nach dem Einschalten der Anlage sind alle Ausgänge OFF. - Der Schaltzustand von Ausgängen, die gemäß den I/O-Parametern vordefinierte Funktionen besitzen, wird mit dem OUT Befehl ebenfalls beeinflusst! - Der aktuelle Schaltzustand bleibt auch nach Beendigung oder Abbruch eines Programms erhalten. - Die Schaltlogik sowie die maximale Strombelastbarkeit sind in den Produkt-handbüchern von MCO 305 und FC 300 beschrieben. <p> ACHTUNG! Der Befehl arbeitet mit den vordefinierten PDOs von CAN-Open. Ändern Sie auf keinen Fall diese Default-Einstellung (minimum capability device), denn dann funktioniert der Befehl nicht mehr.</p>
Portabilität	CAN-Module setzen oder rücksetzen ab MCO 5.00.

__ Befehlsreferenz __

Befehlsgruppe	I/O
Querverweise	OUTB, IN, INB, Parameter: 33-60 Klemme X59/1 und X59/2 Modus, IOMODE, 33-63...70 Klemme X59/n Digitaler Ausgang, O_FUNCTION_n
Syntax-Beispiele	OUT 3 1 // MCO 305 Ausgang 3 auf 1 setzen OUT 27 1 // Ausgang 27 der FC 300 Hauptplatine auf 1 setzen OUT X59/3 1 // MCO 305 Ausgang 3 auf 1 setzen OUT X34/1 1 // erstes Relais der Relais-Option (Relais 7) auf 1 setzen
Programmbeispiel	OUT_01.M

□ OUTAN

Kurzinfo	Drehzahlsollwert setzen.
Syntax	OUTAN w
Parameter	w = Bus-Sollwert Bereich: -0X4000 – 0X4000 = -100 % – 100 %
Beschreibung	Mit dem OUTAN Befehl können Sie den Sollwert für den FC 300 Bus vorgeben. (Der Drehzahl- oder Drehmoment-Sollwert hängt davon ab, wie der FC 300 Par. 1-00 gesetzt ist.) Mit OUTAN kann man auch in „Open Loop“ mit MOTOR OFF die Regelung abschalten und den FC 300 als reinen Frequenzumrichter betreiben. So können Sie APOSS benutzen um direkt gesetzte Werte auszugeben, Eingänge zu lesen usw.
	ACHTUNG!: Vor dem Befehl OUTAN muss MOTOR OFF ausgeführt werden. Daher ist die Schleppfehlerüberwachung nicht mehr aktiv.
Befehlsgruppe	I/O
Querverweise	MOTOR OFF, MCO 305 Produkthandbuch, FC 300 Produkthandbuch
Syntax-Beispiel	MOTOR OFF // Regelung ausschalten OUTAN 0X2000 // Drehzahlsollwert auf 50 % setzen


OUTB

Kurzinfo	Zustand der digitalen Ausgänge bytewise verändern.
Syntax	OUTB n w
Parameter	<p>n = Ausgangsbyte 0 = 1 – 8 1 = 27,29</p> <p>bzw. für CANopen I/O-Module: CAN-Bus + (Modul-CAN-ID * 256) + Ausgangsnummer (bzw. Ausgangsbyte)</p> <p>w = Wert (0 ... 255)</p>
 Beschreibung	<p>ACHTUNG! Die Nummerierung der Bytes beginnt bei 0; im Gegensatz zu den einzelnen <u>Ausgängen</u>, deren Nummerierung bei 1 beginnt.</p> <p>Mit dem OUTB Befehl kann der Zustand der digitalen Ausgänge bytewise verändert werden. Der übergebene Bytewert bestimmt den Zustand der einzelnen Ausgänge. Das niederwertigste Bit des Bytewertes entspricht dabei dem Sollzustand des Ausgangs 1.</p> <p>CAN-Module, die die CAN-OPEN Spezifikationen erfüllen, können Sie ebenfalls mit dem OUTB Befehl ansprechen, und zwar über die entsprechende Nummer, die wie folgt definiert ist:</p> <p style="padding-left: 40px;">CAN-Bus + (Modul-CAN-ID * 256) + Ausgangsnummer (bzw. Ausgangsbyte)</p> <p>Beim Ausführen eines solchen Befehls werden temporär die entsprechenden CAN-Objekte angelegt, ausgewertet und anschließend wieder freigegeben; daher können Sie beliebig viele Module ansprechen.</p>
 ACHTUNG!:	<p>Nach dem Einschalten der Anlage sind alle Ausgänge auf OFF. Ausgänge, die gemäß den I/O-Parameter-Einstellungen vordefinierte Funktionen besitzen, werden durch den OUTB Befehl ebenfalls beeinflusst! Der aktuelle Schaltzustand bleibt auch nach Beendigung oder Abbruch eines Programms erhalten.</p> <p>Schaltlogik sowie maximale Strombelastbarkeit siehe MCO 305 Produkthandbuch.</p>
 ACHTUNG!:	<p>Der Befehl arbeitet mit den vordefinierten PDOs von CANopen. Ändern Sie auf keinen Fall diese Default-Einstellung (minimum capability device), denn dann funktioniert der Befehl nicht mehr.</p>
Portabilität	Der Befehl OUTB für CAN-Module ab MCO 5.00.
Befehlsgruppe	I/O
Querverweise	OUT, IN, INB, Parameter: 33-63...70 <i>Klemme X59/n Digitaler Ausgang</i> , O_FUNCTION_n
Syntax-Beispiele	<pre>OUTB 0 10 // Ausgang 2 + 4 durchschalten, sonstige Ausgänge sperren OUTB 0 245 // Ausgang 2 und 4 sperren, alle anderen durchschalten OUTB 0 128 // nur Ausgang 8 durchschalten, andere sperren OUTB 256 1 // Ausgang 1 auf CAN-Modul 1 setzen</pre>
Programmbeispiel	OUTB_01.M

□ OUTDA

Kurzinfo	Analogen FC 300 Ausgang setzen.
Syntax	OUTDA n w
Parameter	n = Ausgangsnummer (42) bzw. für CANopen I/O-Module: CAN-Bus + (Modul-CAN-ID * 256) + Ausgangsnummer (bzw. Ausgangsbyte) w = Wert (0 – 100000)
	ACHTUNG!: Parameter 6-50 muss auf „MCO gesteuert“ eingestellt sein.
Beschreibung	Mit dem OUTDA Befehl kann man den analogen Ausgang der FC 300 Steuerkarte setzen. Der FC 300 hat einen analogen Ausgang, der in Parameter 6-50 konfiguriert wird. Ein Ausgang der FC 300 Steuerkarte kann nur vom Anwendungsprogramm gesetzt werden, wenn er als Optionsausgang im entsprechenden Parameter konfiguriert ist. CAN-Module, die die CANopen Spezifikationen erfüllen, können Sie ebenfalls mit dem OUTDA Befehl ansprechen, und zwar über die entsprechende Modulnummer * 256 + die I/O-Nummer. Beim Ausführen eines solchen Befehls werden temporär die entsprechenden CAN-Objekte angelegt, ausgewertet und anschließend wieder freigegeben; daher können Sie beliebig viele Module ansprechen.
	ACHTUNG!: Der Befehl arbeitet mit den vordefinierten PDOs von CANopen. Ändern Sie auf keinen Fall diese Default-Einstellung (minimum capability device), denn dann funktioniert der Befehl nicht mehr.
Portabilität	Ausgang für CAN-Module setzen ab MCO 5.00.
Befehlsgruppe	I/O
Querverweis	FC 300 Produkthandbuch, Parameter 6-50
Syntax-Beispiel	/* Voraussetzung: Parameter 6-50 ist auf "MCO gesteuert" gesetzt */ OUTDA 42 50000 /* FC 300 Ausgang auf 10 mA setzen */

□ OUTMSG

Kurzinfo	Sendet eine CAN-Nachricht
Syntax	OUTMSG intval longval
Parameter	intval Bytes 2 und 3 der CAN-Nachricht longval Bytes 4 bis 7 der CAN-Nachricht
Beschreibung	Sendet eine CAN-Nachricht (gepuffert). Die CAN-Id (CAN-Identifikationsnummer) ergibt sich aus den Einstellungen der 'slaven'. OUTMSG behandelt immer Objekte, die 8 Bytes lang sind. Für den Benutzer sind nur die Bytes von 2 bis 7 vorgesehen. Byte 0 und 1 sind reserviert.
Portabilität	Der Befehl ist ab MCO 5.00 verfügbar.
Befehlsgruppe	CAN
Querverweise	INMSG, ON CANMSG, INAD
Syntax-Beispiel	temperatur = INAD 1 OUTMSG 20 temperatur

□ PCD

Kurzinfo	Pseudo-Array für den direkten Zugriff auf den Feldbus-Datenbereich.
Syntax	PCD[n]
Parameter	n = Index
Beschreibung	Ohne einen zusätzlichen Befehl COMOPTGET oder COMOPTSEND können Sie mit dem Befehl PCD direkt auf den Feldbus-Datenbereich zugreifen. Es wird wortweise (16-Bit) der Kommunikationsspeicher beschrieben oder gelesen.
	ACHTUNG! Zusätzlich müssen die Parameter 9-15 und 9-16 mit den richtigen Werten gesetzt sein.
Befehlsgruppe	Kommunikationsoption
Querverweise	COMOPTGET, COMOPTSEND, SYSVAR
Syntax-Beispiel	<pre>Variable = PCD[1] // Wort 1 Variable = PCD[1].2 // Bit 2 von Wort 1 Variable = PCD[2].b1 // Byte 1 von Wort 2 PCD[1] = Variable PCD[1].3 = Variable</pre>
Syntax-Beispiel	<pre>_IF (PCD[2]= = 256) THEN // Wert vergleichen _IF (PCD[3].2) THEN // ist Bit 2 von PDO 3 high?</pre>

□ PDO

Kurzinfo	Pseudo-Array für den direkten Zugriff auf die CANopen-PDO.
Syntax	PDO[n]
Parameter	<p>n = 1001 für 1. PDO (Erste 4 Bytes), 1002 für 1. PDO (Nächste 4 Bytes) 2001 für 2. PDO (Erste 4 Bytes), 2002 für 2. PDO (Nächste 4 Bytes) 3001 für 3. PDO (Erste 4 Bytes), 3002 für 3. PDO (Nächste 4 Bytes) 4001 für 4. PDO (Erste 4 Bytes), 4002 für 4. PDO (Nächste 4 Bytes) 5001 für 5. PDO (Serielle PDO werden unterstützt)</p> <p>Aus Kompatibilitätsgründen ebenfalls unterstützt: n = 1, 2 (1. PDO, erste und zweite 4 Bytes)</p> <p>Der Offset zum Lesen anderer PDOs ist immer 1000.</p> <p>Die nächsten 4 Bytes des PDOs werden immer über einen Offset von +1 angesprochen. PDOs können systemabhängig eventuell mehr als 8 Bytes enthalten. Der Zugriff auf diese weiteren Bytes erfolgt nach identischem Schema, z.B. mit 1001, 1002, 1003, 1004, etc. Identisches gilt für das „serielle“ PDO 5 (mit Zugriff über USB oder RS232).</p> <p> <u>PDO Aktivierung (enable / disable)</u> Entsprechend der CANopen-Norm ist nur das PDO 1 (RxPDO = 0x200 + Node-ID / TxPDO = 0x180 + Node-ID) standardmäßig enabled. Die weiteren PDOs müssen bei Bedarf enabled (= freigegeben) werden. Dies kann durch das direkte Setzen des „Valid“-Bits (0x1400 -0x1404 bzw. 0x1800 - 0x1804, Subindex 1) oder durch eine Mapping-Konfiguration mit den Befehlen LINKSDO bzw. LINKPDO erfolgen, wobei das Bit des entsprechenden PDOs automatisch gesetzt wird.</p> <p> <u>CANopen PDO Größe</u> Ein PDO ist immer 8 Byte lang; es kann daher maximal 8 Objekte enthalten.</p> <p> <u>PDO 5 (= „Serielle PDO“) Größe</u> Die Mailbox-Größe des PDO 5 kann eine Länge von maximal circa 250 Bytes besitzen. Das PDO 5 wird ebenfalls von dem Oszilloskop-Tool der APOSS Entwicklungsumgebung genutzt. Es wird deshalb empfohlen dieses PDO nicht in Applikationsprogrammen zu verwenden, die später mit dem Oszilloskop-Tool gedebugged werden sollen.</p>
Beschreibung	<p>Der Befehl PDO ermöglicht den direkten Zugriff auf die CANopen Prozessdatenobjekte.</p> <p>Beim Lesen eines PDO (Reaktion auf ein ankommendes PDO) gibt es drei Möglichkeiten:</p> <ol style="list-style-type: none"> 1. Nach dem Befehl LINKPDO, wird das ankommende Telegramm immer in die entsprechende Systemvariable umgeleitet. Das „Valid“-Bit des PDOs wird automatisch gesetzt. 2. Das Programm greift selbst lesend auf das PDO-Array zu. 3. Mit ON COMBIT, wird eine Funktion aufgerufen, sobald sich das Bit n des PDOs ändert.



__ Befehlsreferenz __

Beim ausgehenden PDO sind zwei Verfahren zu unterscheiden:

1. Nach dem Befehl LINKSDO wird in das PDO geschrieben, sobald sich die Variable ändert. Zyklisches Update des Parameters alle 10 ms. Dieser Default-Wert kann über SDO Eintrag 0X1800 - 0x1804 Subindex 5 entsprechend der CANopen-Spezifikation geändert werden. Das „Valid“-Bit des PDOs wird automatisch gesetzt.
2. Der Befehl PDO schreibt direkt in das ausgehende PDO

Das erste PDO-Arrayelement (z.B. PDO[1001]) enthält die Datenbytes 1 - 4 der maximal 8 Datenbytes des PDOs im Falle eines CANopen-PDOs. Das zweite PDO-Arrayelement (z.B. PDO[1002]) enthält die Datenbytes 5 - 8. Weitere PDO-Arrayelemente gibt es nur bei speziellen Systemen oder erweiterten Bus-Protokollen, z.B. für das „serielle“ PDO 5.

Jedes PDO-Arrayelement (z.B. PDO[1002], PDO[2002]) enthält einen 32-Bit Wert. Man muss hierbei jedoch die Reihenfolge der Bytes in dem 32-Bitwert bei der Auswertung beachten. Am besten lässt sich die Byte-Reihenfolge mit dem byte- (.b) oder wortweisen (.w) Zugriff auf das PDO-Arrayelement überprüfen.

PDO[1001].b1 -> Byte 1 des PDO 1
 PDO[1001].b2 -> Byte 2 des PDO 1
 PDO[1001].b3 -> Byte 3 des PDO 1
 PDO[1001].b4 -> Byte 4 des PDO 1
 PDO[1002].b1 -> Byte 5 des PDO 1
 PDO[1002].b2 -> Byte 6 des PDO 1
 PDO[1002].b3 -> Byte 7 des PDO 1
 PDO[1002].b4 -> Byte 8 des PDO 1



ACHTUNG!

Die beiden Befehle LINKPDO und LINKSDO verknüpfen auf Seiten der Steuerung ein PDO direkt mit internen Systemvariablen und setzen das „Valid“-Bit des PDOs.

Auch wenn Mapping (via LINKSDO oder LINKPDO) konfiguriert wurde, können Sie weiterhin mit dem PDO-Array direkt auf das PDO zugreifen. Allerdings müssen Sie darauf achten, dass beim Schreiben in das PDO-Array nur Bytes verwendet werden, die nicht im Mapping enthalten sind. Andernfalls ist die Integrität der Daten nicht gewährleistet!



ACHTUNG!

Standardgemäß wird ein veränderter PDO-Inhalt automatisch verschickt (Asynchron Modus). Wenn dies nicht gewünscht ist, kann man den SDO Index 0x1800 - 0x1804 Subindex 2 auf einen anderen Wert einstellen (z.B. 254, statt Standard 255). Dadurch wird nicht mehr aktiv verschickt, sondern das PDO muss per Remote-Frame abgeholt werden.

Natürlich kann das PDO Mapping auch extern von einer übergeordneten Steuerung (z.B. SPS, PC) mit den Standard CANopen Prozeduren gesetzt werden. Das heißt, es werden die Mapping-Parameter über die Objekten 0x1600 - 0x1604 und 0x1A00 - 0x1A04 konfiguriert. Bei einer solchen externen Mapping-Konfiguration können jedoch nur über SDOs zugängliche Objekte gemappt werden. Systemdaten, die ausschließlich über interne SYSVAR Nummern zugänglich sind, können nicht in das externe Mapping einbezogen werden.

In der gleichen Weise ist es auch möglich, die PDOs 1 - 5 über das „Valid“-Bit des Subindex 1 der Objekte 0x1400 - 0x1404 bzw. 0x1800 - 0x1804 zu enablen oder disablen.

__ Befehlsreferenz __

Mit den Subindices 2-5 von 0x1800 - 0x1804 kann der Übertragungstyp („Transmission Type“), sowie die Sperr- und Ereigniszeit der TxPDOs definiert werden. Die Einstellung des Übertragungstyps des TxPDOs gilt bei der MCO-Option identisch für das RxPDO.

RxPDOs benutzen keinen extra CAN-Speicher, aber TxPDOs benötigen ein Objekt pro TxPDO, falls sie enabled sind, d.h. das "Valid" Bit gesetzt ist.

Portabilität Der Befehl ist ab MCO 5.00 verfügbar.

Befehlsgruppe CAN

Querverweise SYSVAR

Syntax-Beispiel

```
Variable = PDO[1]           // Datenwert Byte 1 - 4
Variable = PDO[1].2        // Bit 2 von Datenwert Byte 1 - 4
Variable = PDO[2].b1       // Byte 1 von Datenwert Byte 5 - 8
PDO[1] = Variable
PDO[1].3 = Variable
```

Programmbeispiel

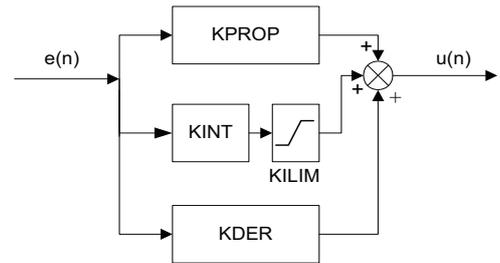
```
_IF (PDO [1] ==256) THEN    // Wert vergleichen
_IF (PDO [2].2) THEN       // Ist Bit 2 von PDO 2 Datenwert Byte 5-8 ?
```



__ Befehlsreferenz __

□ PID

Kurzinfo	PID-Filter berechnen.	
Syntax	$u(n) = \text{PID } e(n)$	
Parameter	$e(n)$ = aktuelle Abweichung (Fehler) auf die der PID-Filter angewendet werden soll	
Rückgabewert	$u(n)$ = Ergebnis der PID-Berechnung	
Beschreibung	<p>Mit dieser Funktion kann ein PID-Filter berechnet werden. Der PID-Filter arbeitet nach der Formel:</p> $u(n) = (KP * e(n) + KD *(e(n)-e(n-1)) + KI*\sum e(n)) / \text{timer}$ <p>wobei gilt:</p> <p>$e(n)$ Fehler zum Zeitpunkt n</p> <p>KP <i>Proportionalfaktor</i> des PID-Reglers</p> <p>KD <i>Differentialfaktor</i></p> <p>KI <i>Integrialfaktor</i> (begrenzt durch <i>Integrationslimit</i>)</p> <p>timer <i>Abtastzeit</i></p> <p>Die entsprechenden Faktoren können mit folgenden Befehlen gesetzt werden:</p> <pre> SET PID KPROP 1 /* setze KP 1 */ SET PID KDER 1 /* setze KD 1 */ SET PID KINT 0 /* setze KI 0 */ SET PID KILIM 0 /* Grenzwert für die Integralsumme = 0 */ SET PID TIMER 1 /* Abtastzeit = 1 */ </pre> <p>wobei das folgende Syntax-Beispiel auch gleich die Default-Belegung der Faktoren zeigt.</p>	
Befehlsgruppe	SYS	
Syntax-Beispiel	<pre> e = INAD 53 u = PID e PRINT "Eingang = ",e, "Ausgabe = ",u </pre>	

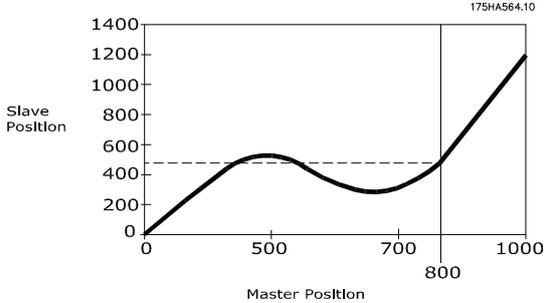


□ POSA

Kurzinfo	Absolut zum aktuellen Nullpunkt positionieren.
Syntax	POSA p
Parameter	p = Position in Benutzereinheiten (BE) absolut zum aktuellen Nullpunkt; in der Standardeinstellung entsprechen die BE der Anzahl der Quadcounts.
Beschreibung	<p>Mit dem POSA Befehl wird die Achse auf eine Position absolut zum aktuellen Nullpunkt bewegt.</p> <p>Wenn durch den Befehl POSA die <i>Negative</i> oder <i>Positive Software-Wegbegrenzung</i> (Parameter 33-41 oder 33-42) überschritten werden, wird nach dem Fehler mit dem nächsten Befehl fortgefahren.</p>
	<p>ACHTUNG!</p> <p>Wenn ein mit SETORIGIN gesetzter und aktiver Temporärnullpunkt existiert, wird die Positionsangabe auf diesen Nullpunkt bezogen.</p>
	<p>ACHTUNG!</p> <p>Sollte beim Aufruf des POSA Befehls noch keine Beschleunigung und/oder Geschwindigkeit definiert sein, wird mit den Werten der Parameter 32-84 <i>Default-Geschwindigkeit</i> und 32-85 <i>Default-Beschleunigung</i> gefahren.</p>
Befehlsgruppe	ABS
Querverweise	VEL, ACC, POSR, HOME, DEFORIGIN, SETORIGIN Parameter: 32-12 <i>Benutzerfaktor Zähler</i> , 32-11 <i>Benutzerfaktor Nenner</i>
Syntax-Beispiel	POSA 50000 /* Achse auf Position 50000 fahren */
Programmbeispiel	POS_01.M



□ POSA CURVEPOS

Kurzinfo	Slave auf die Kurvenposition fahren, die der Master-Position entspricht.											
Syntax	POSA CURVEPOS											
Beschreibung	Dieser Befehl wirkt wie ein POSA Befehl und bewegt den Slave zur entsprechenden Position der Kurve, die durch die aktuelle Master-Position vorgegeben ist.											
	ACHTUNG!:	Wenn ein mit SETORIGIN gesetzter und aktiver Temporärnullpunkt existiert, wird die Positionsangabe auf diesen Nullpunkt bezogen.										
	ACHTUNG!:	Sollte beim Aufruf des POSA Befehls noch keine Beschleunigung und/oder Geschwindigkeit definiert sein, wird mit den Werten der Parameter 32-84 <i>Default-Geschwindigkeit</i> und 32-85 <i>Default-Beschleunigung</i> gefahren.										
Befehlsgruppe	ABS, CAM											
Querverweise	CURVEPOS, SETORIGIN											
Syntax-Beispiel	POSA CURVEPOS // Slave auf die der Master-Position entsprechenden Kurvenposition fahren.											
Beispiel	<table border="1"> <thead> <tr> <th>Master</th> <th>Slave</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> </tr> <tr> <td>500</td> <td>500</td> </tr> <tr> <td>700</td> <td>300</td> </tr> <tr> <td>1000</td> <td>1200</td> </tr> </tbody> </table>	Master	Slave	0	0	500	500	700	300	1000	1200	
Master	Slave											
0	0											
500	500											
700	300											
1000	1200											
	Angenommen, die Istposition ist 800 (vertikale Linie).											
	Fall 1:	Istposition Master ist 800 und Istposition Slave ist 200. POSA CURVEPOS fährt den Slave auf Position 450.										
	Fall 2:	Istposition Master ist 800 und Istposition Slave ist 700. POSA CURVEPOS fährt den Slave ebenfalls auf Position 450.										

□ POSR

Kurzinfo	Relativ zur Istposition positionieren.	
Syntax	POSR d	
Parameter	d = Distanz zur Istposition in Benutzereinheiten [BE]; dies entspricht in der Standardeinstellung der Anzahl Quadcounts.	
Beschreibung	Der POSR Befehl bewegt die Achse auf eine Position relativ zur Istposition.	
	ACHTUNG!:	Wenn beim Aufruf des POSA Befehls noch keine Beschleunigung und/oder Geschwindigkeit definiert ist, wird mit Werten der Parameter 32-84 <i>Default-Geschwindigkeit</i> und 32-85 <i>Default-Beschleunigung</i> gefahren.
	Befehlsgruppe	REL
Querverweise	VEL, ACC, POSA; Parameter: 32-12, 32-11 <i>Benutzerfaktor Zähler und Nenner</i>	
Syntax-Beispiel	POSR 50000 /* Achse relativ um 50000 BE verfahren */	
Programmbeispiel	POS_01.M	

__ Befehlsreferenz __

□ PRINT

Kurzinfo	Ausgabe von Informationen.	
Syntax	PRINT i oder PRINT i;	
Parameter	i = Information, z.B. Variable, Text, CHR (n) getrennt durch Kommata. Der Befehl CHR liefert zu einer Zahl das entsprechende ASCII-Zeichen.	
Beschreibung	<p>Mit dem PRINT Befehl können Rechenergebnisse, Variableninhalte und Textinformationen über die RS485 Kommunikationsschnittstelle am angeschlossenen PC ausgegeben werden, wenn das APOSS-Programm geöffnet und die Kommunikation hergestellt ist.</p> <p>Um mehrere Informationen mit einem einzigen PRINT Befehl auszugeben, müssen die einzelnen Elemente (Variablen, Texte etc.) durch ein Komma getrennt werden. Textinformationen müssen in Anführungszeichen gesetzt werden.</p> <p>Nach jeder PRINT Anweisung wird normalerweise ein Zeilenvorschub erzeugt. Dieser automatische Zeilenvorschub lässt sich durch einen Strichpunkt (;) nach dem letzten Ausgabeelement unterdrücken.</p>	
Befehlsgruppe	SYS	
Querverweise	INKEY	
Syntax-Beispiel	<pre>PRINT "Information ist wichtig !" /* Textinformation ausgeben */ PRINT "Information ist wichtig !"; /* Info ohne Zeilenumbruch ausgeben */ variable = 10 PRINT variable /* Variableninhalt ausgeben */ PRINT APOS /* Abgefragte Istposition ausgeben */ PRINT "Variable", variable,"Pos.:",APOS /* Gemischte Infos ausgeben */</pre>	
Programmbeispiel	Verwendung siehe in allen Programmbeispielen.	

□ PRINT DEV

Kurzinfo	Stoppt die Ausgabe von Informationen.	
Syntax	PRINT DEV nn printlist	
Parameter	nn = Nummer der Ausgabeschnittstelle 0 = Standard Ausgabe -1 = Keine Ausgabe nach dieser Zeile 1 = CAN-Bus 2 = seriell	
Beschreibung	<p>printlist = normales Argument für einen PRINT Befehl</p> <p>PRINT DEV kann benutzt werden um alle PRINT Befehle in einem Programm zu deaktivieren, ohne jeden einzelnen Befehl kommentieren zu müssen.</p>	
		<p>ACHTUNG!: Mit der Anweisung [-1] wird die Standard-Ausgabeschnittstelle neu definiert und gilt dann sofort für alle PRINTs, die kein DEV enthalten.</p>
Befehlsgruppe	SYS	
Querverweise	PRINT, INKEY	
Syntax-Beispiel	<pre>PRINT DEV -1 "ab hier keine Ausgabe mehr" ... PRINT "normaler Print " ... PRINT DEV 0 "jetzt wieder Info ausgeben"</pre>	

□ PULSACC

Kurzinfo Beschleunigung für den virtuellen Master setzen.

Syntax PULSACC a

**ACHTUNG!:**

Der Änderung der Beschleunigung in PULSACC wirkt erst nach dem nächsten PULSVEL Befehl.

Parameter a = Beschleunigung in Hz/s

Beschreibung Mit PULSACC setzen Sie die Beschleunigung für den virtuellen Master (Drehgeberausgang).

Das virtuelle Master-Signal bildet ein Drehgebersignal nach. Zur Berechnung der Pulsbeschleunigung PULSACC sind daher der Parameter *Drehgeberauflösung*, die Geschwindigkeit des Masters und die Rampenzeiten zu berücksichtigen.

Die erzeugten Signale werden auch gleichzeitig als Master-Eingang ausgewertet, so dass MAPOS, MIPOS etc. wie in einem externen Master funktionieren.

Das virtuelle Drehgebersignal kann nur bei Steuerungen mit Encoder-Ausgang ausgegeben werden. Die Zuordnung des Encoder-Ausgangs ist hardware-abhängig. Durch die Konfiguration und Aktivierung des virtuellen Drehgebers (d.h. PULSACC ungleich 0), wird der entsprechende Encoder-Anschluss automatisch als Ausgang geschaltet.

PULSACC = 0 ist Bedingung für das Abschalten des Modus „virtueller Master“, vorausgesetzt es folgt ein PULSVEL Befehl.

Befehlsgruppe SYN

Querverweis PULSVEL

Beispiel Das virtuelle Master-Signal soll einem Drehgebersignal von 1024 Strichen/Umdr. entsprechen. Die maximale Geschwindigkeit von 25 Drehgeber-Umdrehungen/s soll in 1 s erreicht werden.

$$\begin{aligned} \text{PULSACC} &= \frac{\Delta \text{Pulsegeschwindigkeit (PULSVEL) [Hz]}}{\Delta t [\text{s}]} \\ &= \frac{25 \text{ Umdr/s} * 1024 \text{ Striche/Umdr.}}{1 \text{ s}} \\ &= 25600 \text{ Striche/s}^2 = 25600 \text{ Hz/s} \end{aligned}$$

□ PULSVEL

Kurzinfo	Geschwindigkeit für den virtuellen Master setzen.
Syntax	PULSVEL v
Parameter	v = Geschwindigkeit in Pulsen pro Sekunde [Hz]
Beschreibung	<p>Mit PULSVEL setzen Sie die Geschwindigkeit für den virtuellen Master (Drehgeberausgang).</p> <p>Das virtuelle Master-Signal bildet ein Drehgebersignal nach. Zur Berechnung der Pulsgeschwindigkeit sind daher der Parameter <i>Drehgeberauflösung</i> und die Master-Geschwindigkeit zu berücksichtigen.</p>
Befehlsgruppe	SYN
Querverweis	PULSACC
Beispiel	<p>Das virtuelle Master-Signal soll einem Drehgebersignal von 2048 Strichen/Umdr. bei einer Drehgerbdrehzahl von 50 Umdr./s entsprechen.</p> $\text{PULSVEL} = \text{Drehgeberstriche/Umdr.} \cdot \frac{\text{Umdrehungen}}{\text{s}}$ $= 2048 \cdot 50 \text{ Hz} = 102400 \text{ Hz}$

□ REPEAT .. UNTIL ..

Kurzinfo	Bedingte Schleife mit Endkriterium (Wiederhole ... bis Bedingung erfüllt)
Syntax	REPEAT UNTIL Bedingung
Parameter	Bedingung = Abbruchkriterium
Beschreibung	<p>Mit einer REPEAT .. UNTIL Konstruktion kann man den dazwischen liegenden Programmbereich in Abhängigkeit von einem beliebigen Abbruchkriterium ein- oder mehrfach wiederholen. Das Abbruchkriterium setzt sich aus einer oder mehreren Vergleichsoperationen zusammen und wird stets am Ende der Schleife überprüft. Solange das Abbruchkriterium nicht erfüllt ist, wird der Schleifeninhalt wiederholt abgearbeitet.</p>
	<p>ACHTUNG!</p> <p>Da das Abbruchkriterium erst am Ende der Schleife überprüft wird, werden die Befehle innerhalb der Schleifenkonstruktion mindestens einmal ausgeführt</p> <p>Um eine Endlosschleife zu vermeiden, müssen die innerhalb der Schleife abgearbeiteten Befehle direkt oder indirekt Einfluss auf das Ergebnis der Abbruchüberprüfung haben.</p>
Befehlsgruppe	CON
Querverweise	LOOP, WHILE .. DO .. ENDWHILE
Syntax-Beispiel	REPEAT /* Schleife starten */ Befehlszeile 1 Befehlszeile n UNTIL (A !=1) /* Abbruchbedingung */
Programmbeispiel	REPEA_01.M, DIM_01.M, ONINT_01.M, OUT_01.M, INKEY_01.M


□ RSTORIGIN

Kurzinfo	Temporären Nullpunkt löschen.
Syntax	RSTORIGIN
Beschreibung	Mit dem RSTORIGIN Befehl wird ein zuvor mit SETORIGIN gesetzter temporärer Nullpunkt wieder gelöscht, und alle folgenden absoluten Positionierbefehle (POSA) beziehen sich wieder auf den Realnullpunkt.
Befehlsgruppe	INI
Querverweise	SETORIGIN, DEFORIGIN, POSA
Syntax-Beispiel	RSTORIGIN /* temporären Nullpunkt zurücksetzen */
Programmbeispiel	TORIG_01.M, OUT_01.M, VEL_01.M

□ SAVE part

Kurzinfo	Arrays oder Parameter im EPROM sichern.
Syntax	SAVE part part = ARRAYS, AXPARS, GLBPARS oder USRPARS
Beschreibung	Werden Array-Elemente oder Parameter geändert, während das Programm läuft, können die geänderten Werte mit diesen Befehlen im EPROM gespeichert werden. SAVE GLBPARS Sichert die globalen Parameter der Gruppen 30-5* und 33-8*) und die Anwendungsparameter (Gruppe 19-**) im EPROM. SAVE AXPARS Sichert alle anderen Achsenparameter. SAVE USRPARS Sichert nur Anwendungsparameter (Gruppe 19-**).
	ACHTUNG!: Das EPROM kann diesen Befehl nur bis zu 10000-mal ausführen.
Befehlsgruppe	INI
Querverweise	DELETE ARRAYS, SAVEPROM

□ SAVEPROM

Kurzinfo	Speicher in EPROM sichern.
Syntax	SAVEPROM
Beschreibung	<p>Wenn Arrayelemente oder Anwendungsparameter (Gruppe 19-**) geändert werden, während das Programm läuft, bietet SAVEPROM die Möglichkeit, die geänderten Werte zu speichern. Dies muss mit SAVEPROM explizit ausgelöst werden.</p> <p>Der Befehl löst denselben Vorgang aus, wie er auch im Menü <i>Steuerung</i> gestartet werden kann.</p> <p>Wenn Sie nur Array-Elemente oder nur globale Parameter und Benutzerparameter sichern wollen, benutzen Sie die entsprechenden Befehle SAVE .. ARRAY, GLBPAR oder USRPARS.</p>
	<p>ACHTUNG!: Die Ausführungszeit von SAVEPROM hängt von der Menge der zu sichernden Daten ab. Es können bis zu 4 Sekunden sein.</p>
	<p>ACHTUNG!: Bitte beachten Sie, dass Achsparameter nicht mit SAVEPROM gespeichert werden. Dazu müssen Sie den Befehl SAVE AXPARS benutzen.</p>
	<p>ACHTUNG!: Das EEPROM kann diesen Befehl nur bis zu 10000-mal ausführen.</p>
Befehlsgruppe	INI
Syntax-Beispiel	<pre>PRINT "Einen Moment Geduld" SAVEPROM PRINT "Danke"</pre>

□ SDOREAD

Kurzinfo	Liest SDO eines angeschlossenen CANopen-Gerätes.
Syntax	<pre>wert = SDOREAD id index sub id = CAN id (1...127) -id = führt den Befehl ohne Warten auf eine Antwort aus index = Index des Objekts (0x0000...0xFFFF) sub = Subindex (0x00 - 0xFF)</pre>
Rückgabewert	Wert des SDO mit Index und Subindex
Beschreibung	<p>Mit diesem Befehl kann man das SDO eines angeschlossenen CANopen-Gerätes lesen.</p> <p>Nach dem Lesen des SDO, wird der Wert in <i>wert</i> zurückgegeben. Bei Problemen wird ein APOSS Fehler gemeldet.</p> <p>Es ist möglich SDOREAD mit negativen CAN-Id Nummern aufzurufen. Dann wird der Befehl ausgeführt, aber es wird nicht auf die Antwort gewartet. SDOREAD liefert jedoch in solch einem Fall kein aussageführiges Ergebnis. Deshalb sollte mit SDOSTATE das Ergebnis einer aktiven Kommunikation geprüft werden.</p>
Portabilität	Der Befehl ist ab MCO 5.00 verfügbar.
Befehlsgruppe	CAN
Querverweise	SDOWRITE, SDOSTATE

□ SDOREADSEG

Kurzinfo	Segmentiertes Lesen von SDOs (ungepackt).
Syntax	erg = SDOREADSEG id, index, subindex, arrayname
Parameter	<p>id = CAN ID Nummer -id = führt den Befehl ohne Warten auf eine Antwort aus</p> <p>index = 0x2000</p> <p>subindex = Parameternummer</p> <p>arrayname = Name des vorhandenen Arrays</p>
Rückgabewert	Wert im Array festgelegt als Parameter: 1 Byte in 1 Array-Element
Beschreibung	<p>Der Befehl erlaubt das segmentierte Lesen von ungepackten SDOs. Dies ist besonders nützlich für Strings oder binäre Daten.</p> <p>SDOREADSEG liest segmentiert (falls möglich) und liefert das Ergebnis als Parameter in einem Array. Dieses Array enthält ein Byte der segmentierten Daten (Charakter) in einem Array-Element. Siehe unten Beispiel eines SDOREAD mit Warten und ungepackt.</p> <p>Der Befehl kann entweder mit Warten auf das Ergebnis und Fehlermeldung, falls etwas schief geht, oder ohne Warten und ohne Fehlermeldung benutzt werden. Im zweiten Fall, muss das Ergebnis mit SDOSTATE geprüft werden, siehe dort.</p>
Portabilität	Der Befehl ist ab MCO 5.00 verfügbar.
Befehlsgruppe	CAN
Querverweise	SDOREAD, SDOSTATE
Beispiel	<pre> DIM test[20] id = 3 // Routine um einen String auszugeben. // Dabei enthält das Array nur ein Zeichen pro Element long printstring (long[] arr, long len) { long ind ind = 1 WHILE(ind <= len) DO PRINT chr(arr[ind]); ind++ ENDWHILE PRINT " " } // Test segmented SDO ungepackt mit Warten PRINT "" PRINT "Test segmented SDO ungepackt mit Warten " subindex = 1549 // Sw Version value = SDOREADSEG id, 0x2000, subindex, test PRINT " Anzahl Zeichen ",value printstring(test,value) </pre>

□ SDOREADSEGP

Kurzinfo	Segmentiertes Lesen von SDOs (gepackt).
Syntax	erg = SDOREADSEGP id, index, subindex, arrayname
Parameter	<p>id = CAN ID Nummer -id = führt den Befehl ohne Warten auf Antwort aus</p> <p>index = 0x2000</p> <p>subindex = Parameternummer</p> <p>arrayname = Name eines existierenden Arrays</p>
Rückgabewert	Wert im Array gegeben als Parameter: 4 Bytes in 1 Array-Element
Beschreibung	<p>Dieser Befehl erlaubt das segmentierte Lesen von gepackten SDOs. Dies ist besonders für Strings oder binäre Daten nützlich.</p> <p>SDOREADSEGP liest segmentiert (falls möglich) und liefert das Ergebnis als Parameter gepackt in ein Array. Der Befehl SDOREADSEGP packt die Bytes in die Array-Elemente. Das heißt, jedes Array-Element enthält vier Bytes. Siehe unten Beispiel mit Warten und gepackt.</p> <p>Der Befehl kann entweder mit Warten auf das Ergebnis und Fehlermeldung, falls etwas schief geht, oder ohne Warten und ohne Fehlermeldung benutzt werden. Im zweiten Fall, muss das Ergebnis mit SDOSTATE geprüft werden, siehe dort.</p>
Portabilität	Der Befehl ist ab MCO 5.00 verfügbar.
Befehlsgruppe	CAN
Querverweise	SDOREAD, SDOSTATE
Beispiel	<pre> DIM test[20] vltid = 3 // Routine um einen gepackten String auszugeben. // hierbei enthält jedes Element max 4 Zeichen long printpackedstring (long[] arr, long len) { long ind,cc,rel ind = 1 cc = 1 WHILE(cc <= len) DO ind = ((cc-1) % 4) + 1 rel = ((cc-1) mod 4) + 1 PRINT chr(arr[ind].b rel); cc++ ENDWHILE PRINT " " } // Test der segmented SDO gepackt ohne Warten PRINT "" PRINT "Segmented SDO mit SdoReadSegP gepackt ohne Warten" para = 1549 res = 0 wert = sdoreadsegp -vltid, (0x2000 + para), 0, test WHILE(wert == 0) do wert = sdostate vltid res ENDWHILE IF(wert > 0) THEN PRINT "Parameter ",para," hat die Länge ",res printpackedstring(test,res) ELSE PRINT "Lesen von ",para," ist fehlgeschlagen - Fehler ",wert ENDIF </pre>



□ SDOSTATE

Kurzinfo	Ergebnis einer aktiven Kommunikation prüfen.
Syntax	erg = SDOSTATE id wert
Parameter	<p>id = CAN id</p> <p>wert = zusätzlicher Rückgabewert, dessen Bedeutung vom Rückgabewert „erg“ abhängig ist</p>
Rückgabewert	<p>0 = ID ist busy, wartet auf Antwort</p> <p>1 = Antwort angekommen, Ergebnis steht in „wert“</p> <p>2 = segmentiertes Lesen beendet, Daten sind im Array, Anzahl der erhaltenen Bytes steht in „wert“</p> <p>-2 = Timeout trat während des Wartens auf Rückgabewert (index << 8 + subindex) auf</p> <p>-12 = CAN-Error trat auf (Bus Error), (wird intern zurückgesetzt)</p> <p>-xx = interne Fehler in der Firmware</p> <p>-33 = ID nicht benutzt (ID ist frei für neue Kommunikation)</p> <p>-50 = SDO wurde durch den Slave abgebrochen, SDO Abbruch-Code steht in „wert“</p> <p>-51 = Array war zu klein für segmentiertes Lesen (minimale Größe in „wert“)</p> <p>-52 = Toggle Bit Fehler in segmentierten Transfer</p> <p>-53 = zu viele Daten hereingekommen (mehr als im SDOREAD angegeben)</p> <p>-54 = nicht genug Daten erhalten, als <u>Fertig</u> festgestellt wurde</p> <p>-55 = Array Write Fehler beim segmentierten Lesen</p>
Beschreibung	<p>Es ist möglich SDOREAD oder SDOWRITE mit negativen CAN-ID Nummern aufzurufen. Dabei wird der Befehl ausgeführt, aber es wird nicht auf die Antwort gewartet. In diesem Fall liefert SDOREAD jedoch kein brauchbares Ergebnis.</p> <p>Deshalb erlaubt SDOSTATE einen Check des Ergebnisses einer aktiven Kommunikation. SDOSTATE liefert 0 so lange wie die Kommunikation busy ist. SDOSTATE liefert negative Ergebnisse im Fall eines Fehlers (TIMEOUT, SDO-Abbruch, ...). Wenn SDOSTATE ein positives Ergebnis liefert, war der letzte SDO-Befehl vollständig erfolgreich. War der letzte Befehl ein SDOREAD, dann liefert SDOSTATE das Ergebnis im Parameterwert.</p> <p>Es ist möglich mehrere Transaktionen (bis zu 5) zu verschiedenen IDs parallel zu starten. (Natürlich nicht mit der gleichen ID). Zum Beispiel können drei SDOREADS zu den IDs 1,2,3 gesendet und dann nach den Ergebnissen abgefragt werden (polling).</p>
Portabilität	Der Befehl ist ab MCO 5.00 verfügbar.
Befehlsgruppe	CAN
Querverweise	SDOWRITE, SDOREAD
Syntax-Beispiel	<pre>id = 1 wert = 0 erg = SDOREAD -id idx subidx WHILE(erg == 0) DO erg = SDOSTATE id wert ENDWHILE IF(erg > 0) THEN // verwende wert als Ergebnis von SDOREAD ELSE // Fehlerbehandlung ENDIF</pre>

□ SDOWRITE

Kurzinfo	Setzt SDO eines angeschlossenen CANopen-Gerätes.
Syntax	SDOWRITE id index sub wert
Parameter	id = CAN id (1...127) -id = führt den Befehl ohne Warten auf Antwort aus index = Index des Objekts (0x0000...0xFFFF) sub = Subindex (0x00 ... 0xFF) wert = Parameterwert
Rückgabewert	-
Beschreibung	Mit diesem Befehl kann ein SDO zu einem angeschlossenen CANopen-Gerät geschrieben werden. Nach dem Schreiben des SDO, wird der Wert in <i>wert</i> zum entsprechenden Objekt geschrieben. Bei Problemen wird ein APOSS Fehler gemeldet. Es ist möglich SDOWRITE mit negativen CAN-Id Nummern aufzurufen. Dann wird der Befehl ausgeführt, aber nicht auf eine Antwort gewartet. SDOWRITE liefert allerdings in solch einem Fall kein brauchbares Ergebnis. Deshalb sollte das Ergebnis einer aktiven Kommunikation mit SDOSTATE geprüft werden.
Portabilität	Der Befehl ist ab MCO 5.00 verfügbar.
Befehlsgruppe	CAN
Querverweise	SDOREAD, SDOSTATE
Syntax-Beispiel	SDOWRITE 127 0x2300 2 3000 // max. Geschwindigkeit auf 3000 U/Min setzen

□ SET

Kurzinfo	Parameter setzen.
Syntax	SET par v
Parameter	par = Parameterkennung v = Parameterwert
Beschreibung	Mit dem SET Befehl können bestimmte Achsparameter und globale Parameter temporär während der Programmlaufzeit verändert werden. Die zulässigen Parameterkennungen finden Sie im Kapitel „Parameter-Referenz“ im Handbuch MCO 305 Design Guide.
	ACHTUNG!: Die Änderungen der Parameter gelten nur solange das Programm läuft. Nach dem Programmende oder -abbruch sind wieder die ursprünglichen Parameter gültig. Die Änderungen der Parameter können mit dem Befehl SAVEPROM permanent gespeichert werden.
Befehlsgruppe	PAR
Portabilität	SET I_xxx Befehle funktionieren weiterhin und werden automatisch mit den neuen I_FUNKTION Parameter umgesetzt.
Querverweise	GET, Parameter-Referenz
Syntax-Beispiele	SET POSLIMIT 100000 /* Positive Wegbegrenzung setzen */ SET KPROP 150 /* Proportionalfaktor ändern */ SET PRGPAR 2 /* Aktivierte Programmnummer ändern */ SET I_FUNCTION_9_n /* ersetzt bisherigen Befehl SET I_BREAK */

□ SETCURVE

Kurzinfo	CAM-Kurve setzen
Syntax	SETCURVE array
Parameter	array = Name des Arrays bzw. der Kurve
Beschreibung	<p>Mit SETCURVE wird die CAM-Kurve, die in dem 'array' beschrieben ist, ausgewählt. SETCURVE array muss immer vor den Befehlen CURVEPOS, SYNCCxx, SYNCCSTART oder SYNCCSTOP. benutzt werden.</p> <p>Sobald der Befehl ausgeführt wird, sind die notwendigen Vorberechnungen bereits durchgeführt.</p> <p>Siehe CAM Erweiterungen und neue Kurventypen in Curve Arrays and Curve Types.</p> <p>ACHTUNG!: Vor dem Befehl SETCURVE bzw. am Anfang des Programms muss die DIM-Anweisung mit dem Namen der Kurve bzw. des Arrays und der Anzahl der Array-Elemente stehen. Sind mehrere Arrays bzw. Kurven in der zbc (oder cnf) Datei, dann muss die Reihenfolge in der DIM-Anweisung mit der Reihenfolge der Arrays in der zbc-Datei übereinstimmen</p> <p>ACHTUNG!: <u>Wenn SYNCC nicht aktiv ist:</u> Wird SETCURVE benutzt, wenn SYNCC nicht aktiv ist, dann wird durch den Befehl SETCURVE die Kurven-Master-Position zurückgesetzt, und zwar abhängig von der aktuellen Master-Position. Das bedeutet, dass CMASTERCPOS (SYSVAR 4230) aus MAPOS. berechnet wird. Diese Position wird also nicht mehr durch SYNCC zurückgesetzt, sondern kann nur durch ein DEFMCPOS oder durch eine neue SETCURVE außerhalb des SYNCC-Modus zurückgesetzt werden.</p> <p><u>Wenn SYNCC aktiv ist:</u> Wird SETCURVE aber benutzt, während SYNCC aktiv ist, wird CMASTERCPOS nicht verändert. Alle anderen Parameter wie 32-11 <i>Benutzerfaktor Zähler</i>, 32-12 <i>Benutzerfaktor Nenner</i>, 33-23 <i>Startverhalten für Sync.</i>, 33-15 und 33-16 <i>Markeranzahl Master und Slave</i>, 33-17 und 33-18 <i>Markerabstand Master und Slave</i>, 33-21 und 33-22 <i>Master und Slave-Marker Toleranzfenster</i> und alle Kurven-Array-Informationen werden nach dem nächsten Re-Start der Kurve aktualisiert.</p> <p>Während SYNCC aktiv ist, kann die Position CMASTERCPOS nur durch einen Befehl DEFMCPOS, der mit dem nächsten Re-Start der Kurve ausgeführt wird, oder MOVE SYNCORIGN, der sofort ausgeführt wird, beeinflusst werden.</p> <p>CMASTERCPOS (SYSVAR) und CURVEPOS werden nun auch aktualisiert, sogar wenn SYNCC nicht mehr aktiv ist. Diese Werte werden aktualisiert nach einem Befehl SETCURVE (falls SYNCMSTART < 2000 ist) oder nach SYNCC und dem ersten Master-Marker (falls SYNCMSTART = 2000).</p> <p>ACHTUNG!: Das Übertragen des Arrays zum DSP kann einige ms dauern. Ein Kurvenarray mit 900 Werten dauert etwa 40 ms. Deshalb ist die maximale Größe eines Arrays auf 2000 begrenzt. (Die meisten Kurven haben ohnehin nicht mehr als einige Hundert Werte.)</p> <p>Siehe auch Abbildung Curve Array im Kapitel Technische Referenz.</p>
Portabilität	CAM Erweiterungen und neue Fixpunkt-Typen sind ab MCO 5.00 verfügbar. Interpolationspunkte werden ab MCO 5.00 nicht mehr benutzt.
Befehlsgruppe	PAR

__ Befehlsreferenz __

Querverweise	DIM, CMASTERCPOS (siehe Sysvar 4230 Axis Process Data), CURVEPOS, Curve Arrays and Curve Types in „Array Structure of CAM Profiles“ in Kapitel Technische Referenz
Syntax-Beispiel	DIM curve [280] // siehe Anzahl der Elemente in der Titelleiste des CAM-Editors SETCURVE curve

□ SETMORIGIN

Kurzinfo	Beliebige Position als Nullpunkt für den Master setzen.
Syntax	SETMORIGIN wert
Parameter	wert = absolute Position
Beschreibung	Mit dem SETMORIGIN Befehl können Sie eine beliebige Position als neuen Nullpunkt für den Master setzen.
	ACHTUNG!: Der Befehl SETMORIGIN hebt den Befehl DEFMORIGIN auf.
	ACHTUNG!: Um den Nullpunkt für den Master wieder zu ändern, müssen Sie ihn daher mit SETMORIGIN oder DEFMORIGIN neu setzen. RSTORIGIN hat für den Nullpunkt des Masters keine Auswirkung.
Befehlsgruppe	INI
Querverweise	DEFMORIGIN, MAPOS
Syntax-Beispiel	SETMORIGIN 10000 /* Nullpunkt für den Master auf 10000 setzen */

□ SETORIGIN

Kurzinfo	Absolute Position als Temporärnullpunkt setzen.
Syntax	SETORIGIN p
Parameter	p = Absolute Position in Bezug zum Realnullpunkt
Beschreibung	Mit dem SETORIGIN Befehl kann eine beliebige absolute Position vorübergehend als neuer Bezugspunkt für den absoluten Positionierbefehl (POSA) gesetzt werden. Diese Position wird Temporärnullpunkt genannt. In Verbindung mit dem Befehl CURVEPOS kann man festlegen, dass die aktuelle Slave-Position mit dem entsprechenden Wert der Kurve übereinstimmt.
	ACHTUNG!: Es ist möglich, mehrere SETORIGIN ohne ein vorheriges RSTORIGIN auszuführen. Die absolute Positionsangabe bezieht sich immer auf den Realnullpunkt. Der letzte ausgeführte SETORIGIN bestimmt somit die Lage des temporären Nullpunkts in Bezug zum Realnullpunkt.
Befehlsgruppe	INI
Querverweise	RSTORIGIN, DEFORIGIN, POSA, CURVEPOS
Syntax-Beispiel	SETORIGIN 50000 /* Temporärnullpunkt auf 50000 setzen */
Syntax-Beispiel	SETORIGIN (-CURVEPOS) // Temporärnullpunkt auf den Anfang der Kurve setzen
Programmbeispiel	TORIG_01.M, OUT_01.M, VEL_01.M

□ SETVLT

Kurzinfo	Setzt einen FC 300 Parameter.
Syntax	SETVLT par v
Parameter	par = Parameternummer v = Parameterwert
Beschreibung	<p>Mit dem SETVLT Befehl können bestimmte Parameter im FC 300 temporär verändert und somit auch die Konfiguration des FC 300 temporär werden.</p> <p>Da ausschließlich Ganzzahlenwerte übertragen werden, muss der zu übertragende Parameterwert mit dem zugehörigen Umwandlungsindex angepasst werden.</p> <p>Die Liste der FC 300-Parameter mit den zugehörigen Umwandlungsindizes finden Sie im FC 300 Produkthandbuch.</p>
	<p>ACHTUNG!: Die Parameteränderungen werden nur im RAM gespeichert. Nach dem Aus- und Wiedereinschalten werden wieder die ursprünglichen Parameter eingelesen.</p>
Befehlsgruppe	PAR
Querverweise	GETVLT
Syntax-Beispiel	<pre>/* Par. 3-03 "Maximaler Sollwert" high auf 60 Hz ändern */ /* -Umwandlungsindex = -3 (multipliziert mit 10³ beim Senden) */ SETVLT 303 60000</pre>

□ SETVLTSUB

Kurzinfo	Setzt einen FC 300-Parameter mit Indexnummer.
Syntax	SETVLTSUB par indxno v
Parameter	par = Parameternummer indxno = Indexnummer v = Parameterwert
Beschreibung	<p>Mit SETVLT Befehlen können FC 300 Parameter und somit auch die Konfiguration des FC 300 vorübergehend geändert werden, in diesem Fall auch alle Parameter mit Indexnummer.</p> <p>Da ausschließlich Ganzzahlenwerte übertragen werden, muss der zu übertragende Parameterwert mit dem zugehörigen Umwandlungsindex angepasst werden.</p> <p>Die Liste der FC 300-Parameter mit den zugehörigen Umwandlungsindizes finden Sie im FC 300 Produkthandbuch.</p>
	<p>ACHTUNG!: Die Parameteränderungen werden nur im RAM gespeichert. Nach dem Aus- und Wiedereinschalten werden die ursprünglichen Parameter wieder eingelesen.</p>
Befehlsgruppe	PAR
Querverweise	GETVLTSUB
Syntax-Beispiel	<pre>SETVLT 025 1 100 // Index 1 des Parameters 25 "Quick Menü" auf 100 "Konfiguration" setzen</pre>

□ STAT

Kurzinfo	Status der Achse und der Steuerung abfragen.
Syntax	erg = STAT
Rückgabewert	erg = Achs- und Steuerungsstatus (4-Byte-Wert): Byte 3 MSB Bit 0 1 = Achse fährt Bit 1 1 = Überlauf Slave-Drehgeber Bit 2 1 = Überlauf Master-Drehgeber Bit 3 1 = Lageregelung ist temporär abgeschaltet *) Byte 2 Statusbyte der Lagereglereinheit Bit 7 1 = Lageregelung abgeschaltet Bit 2 1 = Position erreicht Bit 0,1,3-6 sind ohne Bedeutung Byte 1 wird nicht verwendet Byte 0 LSB Bit 7 1 = Endschalter aktiv Bit 6 1 = Referenzschalter aktiv Bit 5 1 = Startschalter aktiv Bit 2 1 = Lageregelung abgeschaltet Bit 0,1,3,4 werden nicht verwendet *) d.h. die Achse befindet sich innerhalb des Toleranzbereiches des Regelfensters REGWMAX / REGWMIN. Sobald das Regelfenster verlassen wird, wird die Lageregelung wieder eingeschaltet. SYSVAR 4258 PFG_LASTERROR enthält mehr Information über den Fehlertyp.
Beschreibung	Der STAT Befehl liefert den aktuellen Status der Lagereglereinheit sowie der abgefragten Achse zurück. Zum Beispiel ob die Lageregelung abgeschaltet, die Bewegung beendet oder der Endschalter aktiv ist. Der Zustand der Programmausführung kann nicht mit STAT, sondern nur mit AXEND abgefragt werden. Der Status wird aus vier Byte zusammengesetzt; Bedeutung siehe oben.
Befehlsgruppe	SYS
Querverweise	AXEND
Syntax-Beispiel	PRINT STAT /* Statuswort ausgeben */
Programmbeispiel	STAT_01.M



□ SUBMAINPROG .. ENDPROG

Kurzinfo	Definition des Unterprogrammbereichs.
Syntax	SUBMAINPROG ENDPROG
Beschreibung	<p>Das Kennwort SUBMAINPROG leitet den Unterprogrammbereich ein, ENDPROG beendet diesen speziellen Programmbereich. Unter einem Unterprogramm versteht man Befehlsfolgen, die mit der GOSUB Anweisung von verschiedenen Programmpositionen aus aufgerufen und ausgeführt werden können.</p> <p>Innerhalb des Unterprogrammbereichs müssen alle verwendeten Unterprogramme enthalten sein. Es ist zwar möglich, den Unterprogrammbereich an beliebiger Stelle innerhalb eines Programms einzufügen, aus Gründen der Übersichtlichkeit wird jedoch empfohlen, ihn an den Anfang oder Ende des Programms zu stellen.</p>
	<p>ACHTUNG!: Es darf nur einen Unterprogrammbereich innerhalb eines Programms geben.</p>
Befehlsgruppe	CON
Querverweise	SUBPROG .. RETURN, GOSUB, ON ERROR GOSUB, ON INT n GOSUB
Syntax-Beispiel	<pre>SUBMAINPROG /* Beginn des Unterprogrammbereichs */ Unterprogramm 1 Unterprogramm n ENDPROG /* Ende des Unterprogrammbereichs */</pre>
Programmbeispiel	GOSUB_01.M, AXEND_01.M, ERROR_01.M, INCL_01.M, STAT_01.M

□ SUBPROG name .. RETURN

Kurzinfo	Definition eines Unterprogramms.
Syntax	SUBPROG name RETURN
Parameter	name = Name der Unterprogramms
Beschreibung	<p>Die Anweisung SUBPROG kennzeichnet den Beginn eines Unterprogramms. Direkt nach SUBPROG muss der Name des Unterprogramms stehen. Der Name kann aus einem oder mehreren Zeichen bestehen und muss eindeutig sein, das heißt es darf nicht mehrere Unterprogramme mit dem gleichen Namen geben.</p> <p>Mit der GOSUB Anweisung und dem Namen kann ein Unterprogramm zu jedem beliebigen Zeitpunkt aufgerufen und ausgeführt werden.</p> <p>Ein Unterprogramm kann beliebig viele Befehlszeilen enthalten und auf alle Programmvariablen zugreifen. Der letzte Befehl innerhalb eines jeden Unterprogramms muss die RETURN Anweisung sein, durch die das Unterprogramm verlassen und die Programmausführung mit dem auf die GOSUB Anweisung folgenden Befehl fortgesetzt wird.</p>
	<p>ACHTUNG!: Alle Unterprogramme müssen sich innerhalb des durch SUBMAINPROG und ENDPROG definierten Programmbereichs befinden.</p> <p>Es ist nicht zulässig innerhalb eines Unterprogramms ein zweites Unterprogramm zu deklarieren.</p>
Befehlsgruppe	CON
Querverweise	SUBMAINPROG .. ENDPROG, GOSUB, ON ERROR GOSUB, ON INT .. n GOSUB
Syntax-Beispiel	<pre> SUBMAINPROG /* Beginn Unterprogrammbereich */ SUBPROG up1 /* Beginn von up1 */ Befehlszeile 1 Befehlszeile n RETURN /* Ende von up1 */ ENDPROG /* Ende Unterprogrammbereich */ </pre>
Programmbeispiel	GOSUB_01.M, AXEND_01.M, ERROR_01.M, IF_01.M, STAT_01.M



□ SYNCC

Kurzinfo Synchronisation im CAM-Modus

Syntax SYNCC num

Parameter num = Anzahl der Kurven, die abgearbeitet werden sollen;
 0 = der Antrieb bleibt im CAM-Modus bis mit Befehlen wie MOTOR STOP, CSTART, POSA etc. ein anderer Mode gestartet wird
 < 0 = startet SYNCC ohne die Marker zurückzusetzen

Beschreibung Der Befehl SYNCC startet den CAM-Modus (Kurvenscheibensteuerung). Von diesem Augenblick werden die Kurvenpositionen des Masters hoch gezählt, und zwar abhängig von den tatsächlichen Master-Positionen und dem in Par. 33-23 definierten *Startverhalten für Sync*: Wo und wann angefangen wird, zu zählen.

Mit dem Parameter SYNCCMSTART = 2000 werden die Kurvenpositionen des Masters erst nach dem nächsten Master-Marker gezählt.

**ACHTUNG!:**

Der Befehl SYNCC startet weder den Slave, noch unterbricht er eine Fahrbewegung (z. B. CVEL); nur SYNCCSTART startet tatsächlich.

ACHTUNG!:

Der Antrieb bleibt solange im CAM-Modus bis *num* Kurven erfolgreich abgearbeitet wurden.

Wenn (nach *num* Kurven) die Synchronisation normal verlassen wird, wird – falls kein SYNCCSTOP mit einem entsprechenden Punktepaar definiert ist, – das Start-Stop-Punktepaar 2 benutzt, um den Antrieb anzuhalten. Dieser wird dann an der Position *slavepos* (siehe Parameter) stehen bleiben.

SYNCC kann mit einer negativen Zahl gestartet werden. Damit wird SYNCC ohne Zurücksetzen der Marker gestartet. Die negative Zahl wird dann um 1 verringert bevor der absolute Wert als Zähler benutzt wird. So kann der Zähler weiterhin benutzt werden.

Portabilität Starten mit negativen Zahlen ist ab MCO 5.00 möglich.

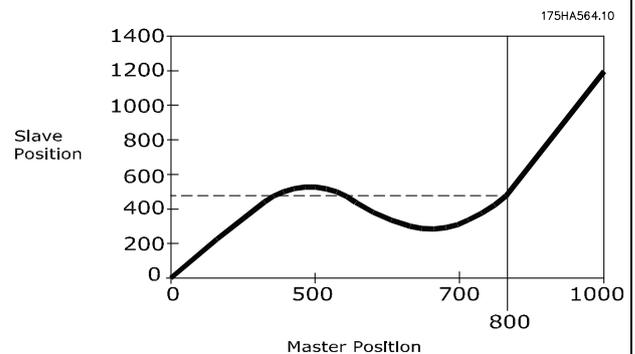
Befehlsgruppe CAM

Querverweise SYNCCSTART

Syntax-Beispiel DIM curve [280] // siehe Anzahl der Elemente in der Titelleiste des CAM-Editors
 SETCURVE curve // Kurve setzen
 SYNCC // Synchronisation im CAM-Modus

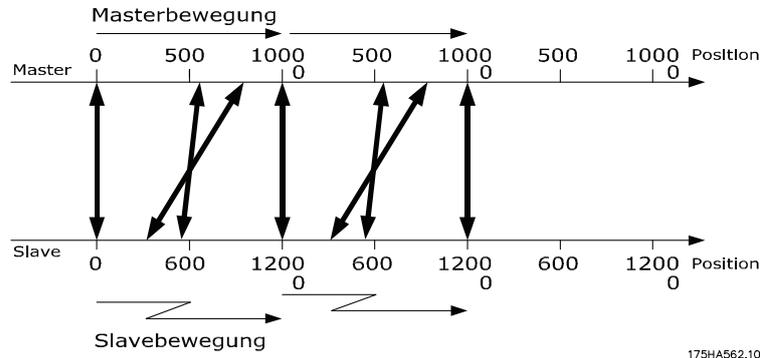
Beispiel Fixpunkte einer Kurve:

Master	Slave
0	0
500	500
700	300
1000	1200



__ Befehlsreferenz __

Demzufolge sperrt SYNCC 1 die Slave- und Master-Position gemäß der Array-Definition.



□ SYNCCMM

Kurzinfo Synchronisation im CAM-Modus mit Markerkorrektur.

Syntax SYNCCMM num

Parameter num = Anzahl der Kurven, die abgearbeitet werden sollen;
0 = der Antrieb bleibt im CAM-Modus bis mit Befehlen wie MOTOR STOP, CSTART, POSA etc. ein anderer Mode gestartet wird

Beschreibung Der Befehl SYNCCMM bewirkt wie SYNCC eine Synchronisation im CAM-Modus, führt aber zusätzlich eine Markerkorrektur (nur, wenn der Master vorwärts fährt) durch. Um den Abstand zwischen Sensor und Arbeitspunkt zu speichern, wird der Par. 33-17 *Markerabstand Master* benutzt. Damit kann die Markerposition ohne Änderung der Kurve korrigiert werden. Und es sind auch größere Sensorabstände als die eigentliche Kurvenlänge möglich. In diesem Fall wird für die Markerkorrektur ein FIFO benutzt (siehe Beispiel).

Der Marker kann der Nullimpuls des Drehgebers oder ein externes 24-Volt-Signal sein.

**ACHTUNG!:**

Der Befehl SYNCCMM startet weder den Slave, noch unterbricht er eine Fahrbe-
wegung (z. B. CVEL); nur SYNCCSTART startet tatsächlich.

ACHTUNG!:

Der Antrieb bleibt solange im CAM-Modus bis 'num' Kurven erfolgreich abgearbeitet wurden.

Wenn (nach 'num' Kurven) die Synchronisation normal verlassen wird, wird – falls kein SYNCCSTOP mit einem entsprechenden Punktepaar definiert ist, – das Start-Stop-Punktepaar 2 benutzt, um den Antrieb anzuhalten. Dieser wird dann an der Position 'slavepos' (siehe Parameter) stehen bleiben.

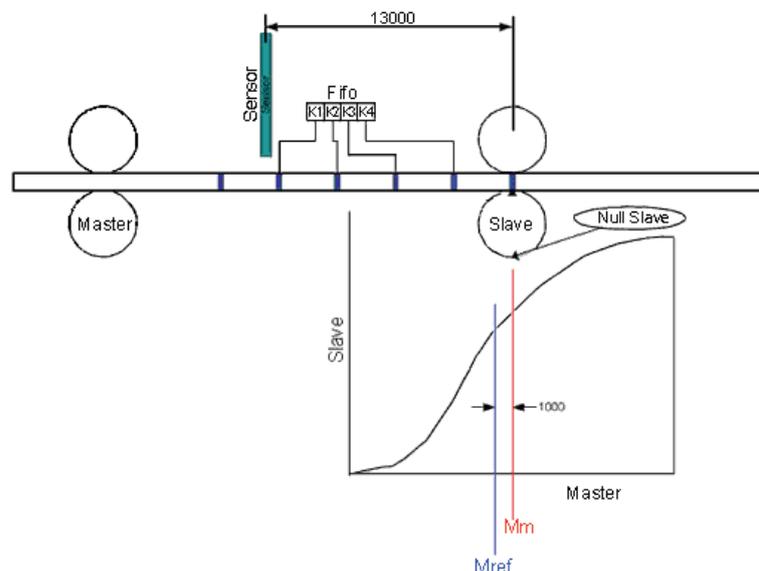
Befehlsgruppe CAM

Querverweis Par. 33-17 *Markerabstand Master*

Syntax-Beispiel SETCURVE curve

SYNCCMM 1 // 1 x Synchronisieren im CAM-Modus mit Markerkorrektur

Beispiel Wenn zum Beispiel die Kurvenlänge 3000 und der Abstand des Sensors zum Arbeitspunkt 13000 ist, gibt es ein FIFO mit 4 Registern und einen Offset von 1000, der betrachtet werden muss.



□ SYNCCMS

Kurzinfo	Synchronisation im CAM-Modus mit Markerkorrektur des Slaves.
Syntax	SYNCCMS num
Parameter	num = Anzahl der Kurven, die abgearbeitet werden sollen; 0 = der Antrieb bleibt im CAM-Modus bis mit Befehlen wie MOTOR STOP, CSTART, POSA etc. ein anderer Mode gestartet wird.
Beschreibung	<p>Der Befehl SYNCCMS bewirkt wie SYNCC, eine Synchronisation im CAM-Modus, führt aber zusätzlich eine Markerkorrektur des Slaves durch. Hier wird nicht die Kurvenposition korrigiert, sondern die Slave-Position.</p> <p>Es wird – im Gegensatz zu SYNCCMM, – kein FIFO gebildet.</p> <p>Der Marker kann der Nullimpuls des Drehgebers oder ein externes 24-Volt-Signal sein.</p> <p>ACHTUNG!: Der Befehl SYNCCMS startet weder den Slave, noch unterbricht er eine Fahrbewegung (z.B. CVEL); nur SYNCCSTART startet tatsächlich.</p> <p>ACHTUNG!: Der Antrieb bleibt solange im CAM-Modus bis 'num' Kurven erfolgreich abgearbeitet wurden.</p> <p>Wenn (nach 'num' Kurven) die Synchronisation normal verlassen wird, wird – falls kein SYNCCSTOP mit einem entsprechenden Punktepaar definiert ist, – das Start-Stop-Punktepaar 2 benutzt, um den Antrieb anzuhalten. Dieser wird dann an der Position 'slavepos' (siehe Parameter) stehen bleiben.</p>
Befehlsgruppe	CAM
Querverweise	Par. 33-18 <i>Markerabstand Slave</i>
Syntax-Beispiel	<pre>SETCURVE curve SYNCCMS 0 // Synchronisieren im CAM-Modus mit Markerkorrektur des Slaves.</pre>



□ SYNCSTART

Kurzinfo	Slave zur Synchronisation im CAM-Modus starten.
Syntax	SYNCSTART pnum
Parameter	<p>pnum = Start-(Stop-)Punktepaar-Nummer</p> <p>pnum > 0 Bei Erreichen des entsprechenden A-Punktes wird mit dem Einkuppeln begonnen; vorausgesetzt, der Master fährt in positiver Richtung; die Einkuppelkurve wird im B-Punkt beendet. Wenn A- und B-Punkt identisch sind, wird der Slave mit der eingestellten Maximalgeschwindigkeit – also ohne Kurve – eingekuppelt, sobald der Master diesen Punkt erreicht hat.</p> <p>pnum = 0 Der Slave wird sofort mit der eingestellten Maximalgeschwindigkeit eingekuppelt (aufsynchronisiert). Dabei ist es egal, in welcher Richtung der Master fährt und ob er überhaupt fährt.</p> <p>pnum < 0 Es wird ebenfalls das entsprechende Punktepaar verwendet, allerdings beginnt das Einkuppeln beim B-Punkt und wird beim A-Punkt – also in negativer Richtung beendet.</p>
Beschreibung	<p>Der Befehl startet die Bewegung des Slaves. Mit 'pnum' wählt man das Punktepaar aus, das festlegt, an welcher Master-Position die Synchronisation startet und wo sie beendet sein soll.</p> <p>Beim Vorwärtsfahren startet die Synchronisation am A-Punkt und wird bis zum B-Punkt beendet. Beim Rückwärtsfahren wird sie am B-Punkt gestartet und bis zum A-Punkt beendet.</p>
Befehlsgruppe	CAM
Querverweise	SETCURVE, Start-Stop-Punkte
Syntax-Beispiel	<pre>SETCURVE curve SYNC 0 // Synchronisieren im CAM-Modus SYNCSTART 1 // Slave am A-Punkt vom Start-Stop-Punktepaar 1 einkuppeln</pre>

__ Befehlsreferenz __

□ SYNCCSTOP

Kurzinfo Slave nach der CAM-Synchronisation anhalten.

Syntax SYNCCSTOP pnum slavepos

Parameter pnum = (Start-)Stop-Punktepaar

pnum > 0 Bei Erreichen des entsprechenden A-Punktes wird mit dem Auskuppeln begonnen; vorausgesetzt, der Master fährt in positiver Richtung; die Auskuppelkurve wird im B-Punkt beendet.

Wenn A- und B-Punkt identisch sind, wird der Slave mit der eingestellten Maximalgeschwindigkeit – also ohne Kurve – ausgekuppelt, sobald der Master diesen Punkt erreicht hat.

pnum = 0 Der Slave wird sofort mit der eingestellten Maximalgeschwindigkeit ausgekuppelt. Dabei ist es egal, in welcher Richtung der Master fährt und ob er überhaupt fährt.

pnum < 0 Es wird ebenfalls das entsprechende Punktepaar verwendet, allerdings beginnt das Auskuppeln beim B-Punkt und wird beim A-Punkt – also in negativer Richtung beendet.

slavepos = Position, an der der Slave nach dem Auskuppeln stehen soll.

**ACHTUNG!:**

Beim Vorwärtsfahren beginnt das Auskuppeln am A-Punkt und endet am B-Punkt; beim Rückwärtsfahren umgekehrt.

ACHTUNG!:

Wird das Programm ohne SYNCCSTOP Befehl verlassen, wird standardgemäß mit dem zweiten Punktepaar zum ausgekuppelt und an der in den → *Kurvendaten* definierten Slave-Stop-Position angehalten.

Beschreibung Der Befehl stoppt die Synchronisation ohne den SYNCC Modus zu verlassen. Der Slave wird entsprechend dem Punktepaar, das in 'pnum' definiert ist, ausgekuppelt. Dann erst wird der Slave tatsächlich angehalten. Wenn der Stopp-Punkt erreicht ist, muss der Slave auf 'slavepos' sein.

Befehlsgruppe CAM

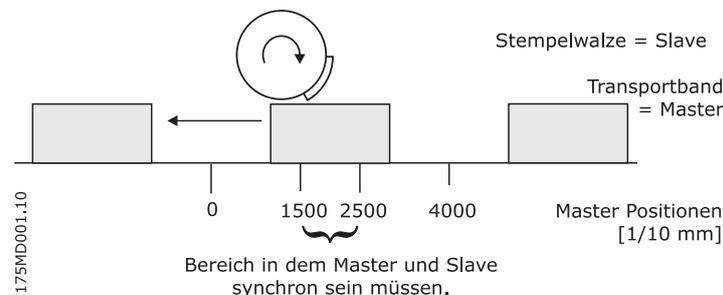
Querverweise Registerkarte Kurvendaten: Slave-Stop-Position

Syntax-Beispiel SETCURVE curve

```
SYNCC 0 // Synchronisieren im CAM-Modus
SYNCCSTART 1 // Slave mit Start-Punktepaar 1 starten
SYNCCSTOP 2 0 // Slave mit Stopp-Punktepaar 2 an der
// Slave-Position 0 bzw. 3600 anhalten
```

Beispiel

Slave Positionen[Grad]
Stempel Anfang 120°, Ende 240°



□ SYNCERR

Kurzinfo Aktuellen Synchronisationsfehler einer Achse bzw. des Slaves abfragen.

Syntax erg = SYNCERR

Rückgabewert erg = aktueller Synchronisationsfehler des Slaves in BE [qc] bzw. im CAM-Modus in BE und

- als absoluter Wert, wenn in Par. 33-13 SYNCACCURACY die Größe des Genauigkeitsfensters mit positivem Vorzeichen definiert ist;
- mit Vorzeichen, wenn in SYNCACCURACY die Größe des Fensters mit negativem Vorzeichen definiert ist.

Beschreibung SYNCERR liefert den aktuellen Synchronisationsfehler in qc bzw. im CAM-Modus in Benutzereinheiten BE zurück. Das ist der Abstand zwischen der aktuellen Master-Position (umgerechnet mit Getriebefaktor und Offset) und der Istposition der entsprechenden Achse bzw. des Slaves.

Wenn der Par. 33-13 *Genauigkeitsfenster für Positionssynchronisation* mit negativem Vorzeichen definiert wird, können Sie auch feststellen, ob die Synchronisation vorausläuft (negatives Ergebnis) oder hinterherläuft (positives Ergebnis).



ACHTUNG!:

Bis Optionskarte Version < 5.00: SYNCERR funktioniert nur im Synchronisationsbetrieb. Sobald man SYNCM oder SYNCPL verlässt, werden die Pulse nicht mehr gezählt. SYNCERR wird nur innerhalb eines Synchronisationsbefehls aktualisiert.

Mit Optionskarte ab Software 5.00: SYNCERR wird (ebenso wie die interne Master-Sollposition G_Mpcmd) aktualisiert, wenn SYNCPL oder SYNCM nicht mehr aktiv sind, z.B. nach einem MOTOR STOP.

Befehlsgruppe I/O

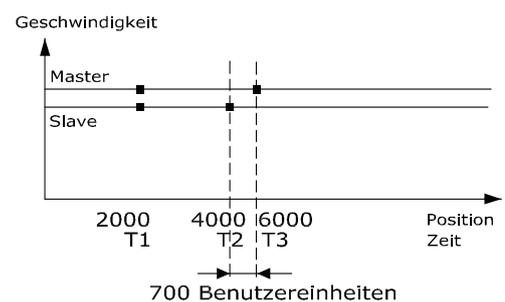
Querverweise TRACKERR, MAPOS, APOS,

Parameter: 33-12 *Positionsoffset für Synchronisation* (SYNCPOSOFFS), 33-10 und 33-11 *Synchronisationsfaktor Master und Slave*, 33-13 *Genauigkeitsfenster für Positionssynchronisation* (SYNCACCURACY)

Syntax-Beispiel PRINT SYNCERR /* aktuellen Synchronisationsfehler abfragen */

Beispiele SYNCACCURACY = 1000

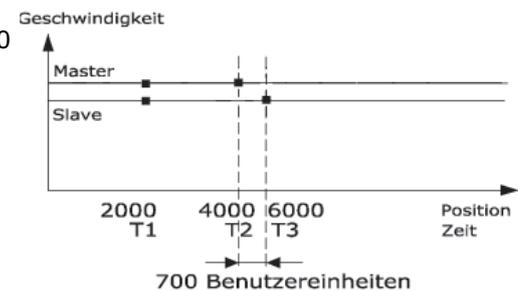
Demzufolge gibt SYNCERR den absoluten Wert 700 zurück.



SYNCACCURACY = 1000

SYNCERR wird den absoluten Wert 700 anzeigen, auch wenn der Slave dem Master voraus ist.

Daher gibt SYNCERR den Wert -700 zurück und zeigt damit, dass der Slave vor dem Master ist.

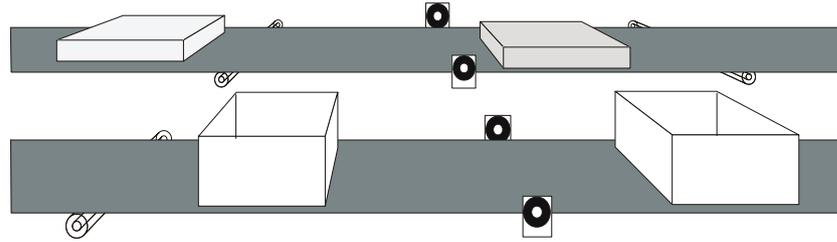


□ SYNCM

Kurzinfo	Winkel-/Positionssynchronisation mit dem Master mit Markerkorrektur.
Syntax	SYNCM
Beschreibung	<p>Der Befehl SYNCM bewirkt genau wie der folgende SYNCP Befehl eine Winkel-/Positionssynchronisation mit dem Master, führt aber zusätzlich eine Markerkorrektur durch. Dabei wird bereits während des Anlaufs der Synchronisation auf den nächsten errechneten Marker aufsynchronisiert. So kann man zum Beispiel unterschiedliches Laufverhalten wie Schlupf ausgleichen.</p> <p>Wenn RAMPTYPE ≥ 2 wird die Synchronisation mit Ruckbegrenzung gestartet. Dies betrifft nur den Start; das Erreichen der Master-Geschwindigkeit sieht nach wie vor wie ein Trapez aus. Dies hilft die Startprozedur bei schweren Lasten oder empfindlicher Mechanik zu glätten.</p> <p>Nachdem die Synchronisation hergestellt ist, wird bei jedem Marker überprüft, welche Abweichung vorliegt (oder bei jedem n-ten Marker, falls die Markeranzahl für Master und Slave nicht identisch ist). Diese wird als neuer Offset in die Regelung eingebracht und es wird sofort versucht, die Abweichung auszugleichen. Dabei werden allerdings die eingestellten Werte für Geschwindigkeit VEL sowie Beschleunigung ACC oder DEC nicht überschritten</p>
	<p>ACHTUNG!:</p> <p>Zu den Parametern, die bei der SYNCP schon verwendet werden, sind hier auch noch Par. 33-25 SYNCREADY und Par. 33-24 SYNCFAULT von Bedeutung.</p>
	<p>ACHTUNG!:</p> <p>Da folgende Parameter zu einer Überbestimmung führen können, sollten Sie darauf achten, dass die Werte sinnvoll sind, zueinander passen und mit denen der Getriebefaktoren konsistent sind.</p> <p>Par. 33-15, 33-16 <i>Markeranzahl Master und Slave</i> Par. 33-17, 33-18 <i>Markerabstand Master und Slave</i> Par. 33-19, 33-20 <i>Markertyp Master und Slave</i></p>
	<p>ACHTUNG!:</p> <p>SYNCM sollten Sie nur einmal aufrufen, denn die Synchronisierung läuft bis zum nächsten Fahr- oder Stoppbefehl. Jeder weitere SYNCM Befehl führt dazu, dass die Synchronisation von vorne beginnt, was normalerweise nicht beabsichtigt ist, außer Sie setzen den aktuellen SYNCERR zurück.</p>
	<p>ACHTUNG!:</p> <p>Wenn in Par. 33-23 <i>Startverhalten bei Sync.</i> definiert, wird beim Start von SYNCM auf die erste Auswertung der Markerpulse gewartet und erst dann der in Par. 33-12 definierte <i>Positionsoffset für Synchronisation</i> angewandt.</p>
Marker Signal	<p>Der Marker kann der Nullimpuls des Drehgebers oder ein externes 24-Volt-Signal sein. (I5 = Master; I6 = Slave)</p>
Portabilität	Startverhalten bei RAMPTYPE ≥ 2 steht ab MCO 5.00 zur Verfügung.
Befehlsgruppe	SYN
Syntax-Beispiel	SYNCM /* Synchronisieren der Position mit Markerkorrektur */



__ Befehlsreferenz __

Beispiel

Selbst wenn beide Bänder synchron laufen, würden die Deckel nie auf gleicher Höhe mit den Schachteln sein. Mit SYNCM wird durch die Auswertung der externen Marker die Positionsabweichung zwischen Master und Slave ermittelt und ausgeglichen.

□ SYNCMARKERSTART

Kurzinfo	Setzt einen Marker oder die Markerbehandlung zurück.
Syntax	SYNCMARKERSTART restarttype
Parameter	restarttype = 0 oder 1
Beschreibung	<p>Dieser Befehl ersetzt die SYNCMPAR 64 Funktionalität. In neueren Anwendungen sollte dieser Befehl statt des Parameters SYNCMPAR 64 verwendet werden.</p> <p><u>restarttype = 0</u></p> <p>Dieser Befehl führt einen Reset der Marker durch; das heißt</p> <ul style="list-style-type: none"> – Löschen der Flags (PG_FLAG_SYNCMMERR, PG_FLAG_SYNCSMERR) – Reset des SyncWindows, der Korrekturwerte, der Getriebekorrektur, der simulierten (faked) Zähler – Reset der ungültigen Zähler, bereinigten Markerabstände, Differenzwerte und aller Mfilter. – 2 Flags setzen (SYN_ACCEPTNXTM, SYN_ACCEPTNXTS) in SYSVAR 4206 PFG_SYNCSTART. <p>Dies führt zu folgendem Ergebnis:</p> <ul style="list-style-type: none"> – Der nächste Master- und nächste Slave-Marker wird in jedem Fall akzeptiert. – Die Korrektur wird entsprechend des aktuellen Werts von SYNCMSTART berechnet. – Die resultierende Korrektur wird wie eine Startkorrektur behandelt. – Das n:m Verhältnis der Marker sollte beibehalten werden (wenn möglich). – Die Marker-Windows werden entsprechend der Parameterwerte zurückgesetzt. – Neustart aller Filter mit den Default-Werten. <p>Dieser Befehl startet auch die Behandlung der Marker, falls nicht schon aktiv.</p> <p>Dieser Befehl kann benutzt werden, um das Marker-Handling außerhalb von SYNCM zu starten (Frühere SYNCMPAR – 64 Funktionalität).</p> <p>Er kann auch zum Neustarten eines laufenden SYNCM ohne Verlust des n:m Verhältnisses benutzt werden.</p> <p><u>restarttype = 1</u></p> <p>Es wird ein echter Reset der Behandlung der Marker durchgeführt. Das heißt, dass zu den oben in Marker Restart beschriebenen Aktionen auch die folgenden Werte zurückgesetzt werden:</p> <p>Markercounts – das heißt, ein n:m Verhältnis geht verloren</p> <p>SYNCMSTART wird neu bewertet. Das kann möglicherweise zu einer neuen Startsequenz führen, falls SYNCM aktiv ist.</p> <p>SYNCSTART wird auf die Startbedingungen besetzt. Das heißt, alle Flags werden wie folgt gesetzt: SYN_START + SYN_NOSLAVE + SYN_NOMASTER + SYN_NOMVEL + SYN_ACCEPTNXTM + SYN_ACCEPTNXTS.</p> <p>Diese Funktion erlaubt einen vollständigen Marker-Neustart außerhalb von SYNCM. Sie bewirkt exakt dasselbe wie ein SYNCM Befehl. Es werden die nächsten Marker akzeptiert und es gibt keine Simulation (Faking), wenn keine Marker erkannt werden. Das ist der Hauptunterschied zur normalen Neustart-Funktion.</p>
Portabilität	Der Befehl ist ab MCO 5.00 verfügbar.
Befehlsgruppe	SYN
Querverweise	Par. 33-28 <i>Markerfilter Konfiguration</i> (SYNCMPAR), Par. 33-23 <i>Startverhalten für Sync</i> . (SYNCMSTART), SYNCM SYSVAR 4209 PFG_SYNCSTART siehe Axis Process Data

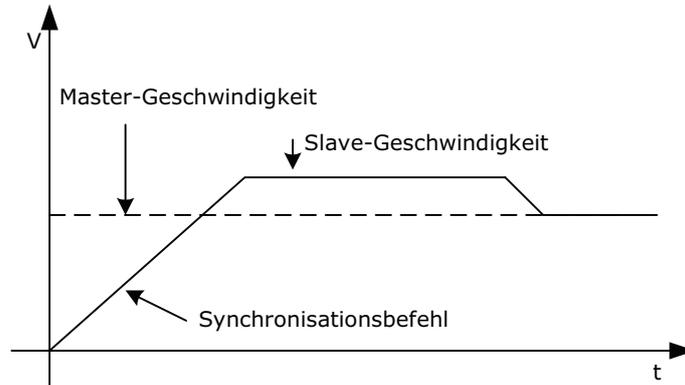


□ SYNCP

Kurzinfo Winkel-/Positionssynchronisation mit dem Master.

Syntax SYNCP

Beschreibung Der Befehl SYNCP führt eine Winkel-/Positionssynchronisation mit dem Master durch. Dabei wird die Position entsprechend der Getriebefaktoren zum Master synchron gehalten, das heißt bei einer Störung von außen wird anschließend versucht, die entsprechende Strecke wieder aufzuholen.



Dabei werden allerdings die eingestellten Werte für Geschwindigkeit VEL sowie Beschleunigung ACC oder DEC nicht überschritten.

Die folgenden Parameter beeinflussen das Verhalten:

Par. 33-10 *Synchronisationsfaktor Master* und

Par. 33-11 *Synchronisationsfaktor Slave* (Getriebefaktoren)

Par. 33-12 *Positionsoffset für Synchronisation*

Par. 33-13 *Genauigkeitsfenster für Positionssync.* (Genauigkeit für Flag)

Par. 32-68 *Reversierungsverhalten Slave*

Beim Aufsynchronisieren wird wie folgt vorgegangen:

Wenn der Befehl SYNCP startet, wird die aktuelle Master-Position ermittelt und als Bezugsposition festgehalten. Aus der Master-Geschwindigkeit wird – unter Berücksichtigung der erlaubten Beschleunigung – die erforderliche Slave-Geschwindigkeit errechnet, um die Master-Position zu erreichen. Der Slave wird solange beschleunigt, bis die errechnete Position erreicht ist oder bis er dicht genug an der Bezugsposition ist, um diese zu erreichen.

Wenn RAMPTYPE ≥ 2 wird die Synchronisation mit Ruckbegrenzung gestartet.

Dies betrifft nur den Start; das Erreichen der Master-Geschwindigkeit sieht nach wie vor wie ein Trapez aus. Dies hilft die Startprozedur bei schweren Lasten oder empfindlicher Mechanik zu glätten.



ACHTUNG!:

Sobald die Abweichung zwischen Slave- und Master-Position kleiner als Par. 33-13 SYNACCURACY ist, wird das ACCURACY-Flag gesetzt.

Wenn Par. 32-68 REVERS so eingestellt ist, dass Zurückfahren nicht erlaubt ist, aber aus irgendeinem Grund der Slave weiter als der Master ist (z.B. weil nur der Master rückwärts gefahren ist), wartet der Slave mit Geschwindigkeit 0 solange, bis die entsprechende Bezugsposition kommt. Dabei berücksichtigt der Slave seine Beschleunigungszeit und fährt ggf. schon los, bevor der Master die Slave-Position erreicht hat.

Statt diesem Aufholverfahren kann man auch zuerst den Slave mit CVEL auf die ungefähre Master-Geschwindigkeit bringen und dann SYNCP auslösen.

__ Befehlsreferenz __



Eine Veränderung des Par. 33-12 *Positionsoffset für Synchronisation* während des Aufholens führt zu erneutem Aufsynchronisieren mit Rampen (siehe oben).

ACHTUNG!:

SYNCP sollten Sie nur einmal aufrufen, denn die Synchronisierung läuft bis zum nächsten Fahr- oder Stoppbefehl. Jeder weitere SYNCP Befehl führt dazu, dass die Synchronisation von vorne beginnt, was normalerweise nicht beabsichtigt ist, außer Sie setzen den aktuellen SYNCERR zurück.

Portabilität	Startverhalten wenn RAMPTYPE >= 2 ist ab MCO 5.00 verfügbar.	
Befehlsgruppe	SYN	
Syntax-Beispiel	SYNCP	/* Normales Synchronisieren der Position */
	CVEL 50	/* Vor dem Synchronisieren Geschwindigkeit erreichen */
	CSTART	
	WAITT 500	
	SYNCP	

□ SYNCSTAT

Kurzinfo	Flag für Synchronisationsstatus abfragen.	
Syntax	erg = SYNCSTAT	
Rückgabewert	erg = Synchronisationsstatus mit folgender Bedeutung:	
	<u>Wert</u>	<u>Bit</u>
	Par. 33-25 SYNCREADY	0
	Par. 33-24 SYNCFAULT	1
	Par. 33-13 SYNCACCURACY	2
	SYNMMMHit	3
	SYNCSMHIT	4
	SYNMMMERR	5
	SYNCSMERR	6
Beschreibung	Folgende Flags sind definiert und können mit SYNCSTAT abgefragt werden: READY, FAULT, ACCURACY und jeweils für Master und Slave die MHIT und MERR.	
SYNCACCURACY	Jede ms wird geprüft ob SYNCERR < SYNCACCURACY gilt und falls dies erfüllt ist, wird SYNCACCURACY (56) gesetzt, andernfalls wird das Flag zurückgesetzt. Diese Überprüfung findet sowohl bei SYNCP als auch bei SYNCM statt. Dieses Flag wird nicht bei SYNCV verwendet. Beim Ausführen eines SYNCP oder SYNCM Befehls wird dieses Flag zurückgesetzt.	
SYNCFAULT / SYNCREADY	Bei jedem SYNCP bzw. SYNCM Befehl werden diese Flags zurückgesetzt. Danach wird bei jedem Markerpuls des Slaves (SYNCP) bzw. bei Vorhandensein eines Markerpulses des Masters und eines Markerpulses des Slaves (SYNCM) geprüft, ob SYNCACCURACY gesetzt ist oder nicht. Wenn es gesetzt ist, wird der Ready-Zähler erhöht und der Fault-Zähler auf 0 gesetzt, andernfalls wird der Fault-Zähler erhöht und der Ready-Zähler auf 0 gesetzt. Ist der Ready-Zähler größer als der in Par. 33-25 SYNCREADY vorgegebene Wert, wird das Flag SYNCREADY gesetzt, im anderen Fall wird es zurückgesetzt. Ist der Fault-Zähler größer als der in Parameter SYNCREADY bzw. SYNCFAULT, wird das Flag SYNCFAULT gesetzt, andernfalls wird es zurückgesetzt.	
SYNMMMHit / SYNCSMHIT	SYNMMMHit und SYNCSMHIT werden gesetzt, wenn der Master-Marker bzw. Slave-Marker erkannt wird. Bei jedem SYNCM Befehl werden diese Flags zurückgesetzt. Danach wird nach dem ersten Auftauchen eines Markerpulses bzw. beim n-ten Markerpuls (Par. 33-15 <i>Markeranzahl Master</i>) das Flag SYNMMMHit gesetzt. Analoges gilt für SYNCSMHIT beim Slave.	

__ Befehlsreferenz __

**ACHTUNG!:**

Dieses Flag wird nicht mehr zurückgesetzt, es sei denn SYNCM wird neu gestartet oder mit dem Befehl SYNCSTATCLR explizit gelöscht.

SYNCMMERR /
SYNCSMERR

Wenn in den Parametern 33-21 oder 33-22 ein *Master- oder Slave-Marker Toleranzfenster* definiert ist, werden SYNCMMERR bzw. SYNCSMERR gesetzt, sobald die maximal erlaubte Distanz erreicht ist und kein Marker erkannt wurde.

Beispiel:

Abstand zwischen 2 Master-Markern Par. 33-17 *Markerabstand* = 30000

Par. 33-21 *Master-Marker Toleranzfenster* = 1000

Das Flag wird also bei 31000 gesetzt, wenn kein Marker erkannt wurde.

Bei jedem SYNCM Befehl werden diese Flags gelöscht.

Wenn *Master-Marker Toleranzfenster* = 0 und damit kein Toleranzfenster definiert ist, wird bei jedem Markerpuls (bzw. bei jedem n-ten) geprüft, ob der Abstand zwischen den zwei zuletzt registrierten Markern kleiner ist als das 1,8-fache des in Par. 33-17 vorgegebenen *Markerabstands*. Wenn nicht, wird das entsprechende Flag gesetzt.

**ACHTUNG!:**

Diese Flags werden automatisch zurückgesetzt: Bei der nächsten erfolgreichen Markerkorrektur und bei erneutem Start von SYNCM oder durch den Befehl SYNCSTATCLR.

Befehlsgruppe SYN

Querverweise SYNCSTATCLR

Syntax-Beispiel IF (SYNCSTAT & 4) THEN OUT 1 1
/* Wenn ACCURACY dann Ausgang setzen */
ENDIF

□ SYNCSTATCLR

Kurzinfo	Zurücksetzen der Flags MERR und MHIT.
Syntax	SYNCSTATCLR wert Der SYNCSTATCLR Befehl sollte nur in einem Unterprogramm zur Fehlerbehandlung eingesetzt werden (siehe ON ERROR GOSUB).
Parameter	wert = 8 = SYNCMMHIT 16 = SYNCSMHIT 32 = SYNCMMERR 64 = SYNCSMERR
Beschreibung	Mit SYNCSTATCLR kann man die dem 'wert' entsprechenden Bits im SYNCSTAT zurücksetzen und damit die Fehler-Flags MERR und die HIT-Flags MHIT löschen. Alle anderen Flags können nicht verändert werden.
Befehlsgruppe	SYN
Querverweise	ON STATBIT, ON ERROR GOSUB, ERRNO, CONTINUE, MOTOR ON
Syntax-Beispiel	SYNCSTATCLR 32 /* aktuelle Fehlermeldung löschen */

□ SYNCV

Kurzinfo Geschwindigkeitssynchronisation mit dem Master.

Syntax SYNCV

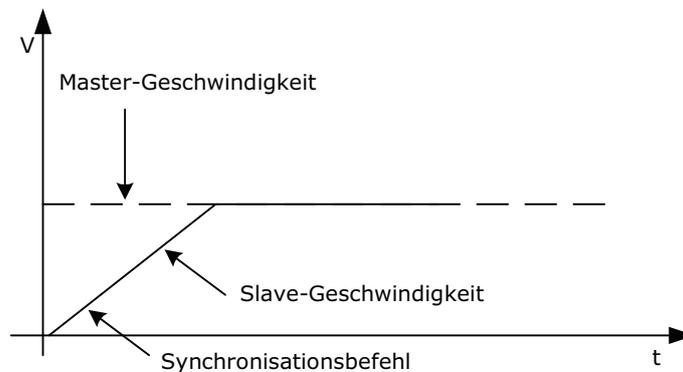
**ACHTUNG!:**

SYNCV sollten Sie nur einmal aufrufen, denn die Synchronisierung läuft bis zum nächsten Fahr- oder Stoppbefehl. Jeder weitere SYNCV Befehl führt dazu, dass die Synchronisierung von vorne beginnt, was normalerweise nicht beabsichtigt ist, außer Sie setzen den aktuellen Synchronisationsfehler SYNCERR zurück.

**ACHTUNG!:**

Schlepp- und Synchronisationsfehler werden im SYNCV Modus nicht überwacht; daher wird empfohlen die Hardware-Drehgeberüberwachung zu benutzen.

Beschreibung Mit SYNCV wird eine Geschwindigkeitssynchronisation mit dem Master durchgeführt, zum Beispiel nach einer Störung von außen. Dabei wird ausschließlich die Geschwindigkeit betrachtet und nicht versucht, die Position aufzuholen.



Beim Aufsynchronisieren sowie während der Synchronisation wird darauf geachtet, dass weder die voreingestellte Geschwindigkeit VEL noch die voreingestellte Beschleunigung ACC oder DEC überschritten wird.

Für die Synchronisation werden die Parameter der Getriebefaktoren verwendet: Par. 33-10 *Synchronisationsfaktor Master*, Par. 33-11 *Synchronisationsfaktor Slave*. Außerdem werden die Geschwindigkeiten nicht einfach durch Differenz der aktuellen minus der letzten Geschwindigkeit ermittelt (Master/Slave), sondern die Werte werden entsprechend der Einstellungen Par. 33-26 *Geschwindigkeitsfilter* (SYNCVFTIME) gefiltert. Dabei wird der Filter für den Slave aus der Maximalgeschwindigkeit bestimmt.

Das heißt, $VELMAX * 5$ entspricht der Drehgeberauflösung für die Filtertabelle, wobei VELMAX die Geschwindigkeit in qc/ms ist. (Die Formel ergibt sich unter der Annahme, dass die Filtertabelle für die Drehgeberauflösung mit einer Maximalgeschwindigkeit von 3000 U/Min gemacht wurde.)

Beim Übergang von der Geschwindigkeitsregelung zur Positionsregelung wird versucht, dies möglichst ruckfrei auszuführen. Dazu wird die neue Sollposition so definiert, dass gilt:

$$\text{command_pos} = \text{actual_pos} + \text{error}$$

old_error, CVEL und AVEL werden beibehalten.

Befehlsgruppe SYN

Querverweise Parameter der Gruppe AXS.


□ SYSVAR

Kurzinfo	Systemvariable (Pseudo-Array) liest Systemwerte.
Syntax	SYSVAR[n] n = index
Beschreibung	<p>Mit der Systemvariablen SYSVAR – einem vorbereitetem Pseudo-Array – können System und Prozessdaten gelesen werden. Dieser Index kann auch benutzt werden, um die Systemvariable mit LINKSDO oder LINKPDO zu verknüpfen oder Aufzeichnungsdaten mit TESTSETP oder TESTSETINDEX festlegen.</p> <p>Dieser Index kann auch benutzt werden, wenn Sie mit LINKSYSVAR die Systemvariable mit dem LCP-Display verknüpfen.</p> <p>Aus Kompatibilitätsgründen ist es weiterhin möglich diese SYSVAR Nummern zu benutzen. Aber es wird empfohlen, für die Adressierung die SDO Dictionary Nummern zu benutzen:</p> <p>0x01nnnnss = Can Index nnnn, Subindex ss siehe SDO Dictionary.</p>
	ACHTUNG!: Die Werte der Systemvariablen sind interne, hardwareabhängige Werte, die sich verändern können.
Befehlsgruppe	CON
Querverweise	LINKSYSVAR, LINKPDO, LINKSDO, TESTSETP, TESTSETINDEX, Axis Parameters (CAN-SDO Nummer 0x2300) siehe SDO Object Dictionary im Anhang in der Online-Hilfe in Kapitel „Technische Referenz“.

Systemprozessdaten

Index	Beschreibung Systemprozessdaten
1	Eingangs-Byte 0 (Eingänge 1..8 von MCO 305)
2	Eingangs-Byte 1 (Eingänge 18..33 von CC)
3	Eingangs-Byte 2 (Eingänge 9..10 / 12 von MCO 305)
9	Ausgangs-Byte 0
17	Oberen 2 Byte die vom APOSS Befehl STAT geliefert werden
22	Interner Millisekunden Zähler: Wert, den auch der APOSS Befehl TIME liefert.
28	Aktueller Motorstrom [1/100 A]; (Parameter 16-14)
30	Motorspannung [1/10 V]; (Parameter 16-12)
31	FC 300 Status (Parameter 16-03)
32	Hauptistwert (HIW) (Parameter 16-05)
33	Aktuelle Zeilennummer des APOSS Programms, falls mit #DEBUG NOSTOP gearbeitet wurde.
34	Motorfrequenz (Parameter 16-13)
35	Motordrehmoment (Parameter 16-16)

□ TESTSETDEST

Kurzinfo Definiert den Speicherbereich in dem eine Datenaufzeichnung abgelegt wird.

Syntax TESTSETDEST Arrayname

Parameter Arrayname = Name des Arrays, das für die Aufzeichnung verwendet wird oder Schlüsselwort DYNMEM um in den freien Speicherplatz aufzuzeichnen.



ACHTUNG!

Bitte stellen Sie sicher, dass die Größe des Arrays oder freien Speichers (DYNMEM) für die Aufzeichnung ausreicht. Für den Datenheader mit den Informationen zur der Aufzeichnung werden 10 Elemente benötigt. Hinzukommt ein Element für die Anzahl der Testreihen plus die Anzahl der konfigurierten Systemvariablen multipliziert mit der Anzahl an Abtastungen. Falls 100 Abtastungen (d.h. TESTSTART 100) mit je 6 Systemvariablen gespeichert werden sollen, ist somit mindestens eine Array-Größe von 611 Elementen oder im Falle von DYNMEM mindestens $611 \times 4 \text{ Bytes} = 2'444$ Bytes freier Speicher notwendig.

Innerhalb des Arrays oder DYNMEM werden die Werte wie folgt gespeichert:
(Alle Werte besitzen eine Datenlänge von 4 Bytes):

Bezeichnung	Inhalt	Bedeutung
Version	999...	Identifikation und Version der Datenstruktur
Achsnummer	0 ...	Achsnummer weniger 1. Aktuell ohne Bedeutung !
ms	1	Abstand zwischen zwei Messungen in ms
Anzahl sysvar	3	Anzahl definierter Systemvariablen (immer 3 bei TESTSETP Konfigurationen)
svi 1	i	Index der 1. aufgezeichneten Systemvariablen (sysvar)
svi 2	i	Index der 2. aufgezeichneten Systemvariablen (sysvar)
...	i	Index der ...
svi n	i	Index der n. aufgezeichneten Systemvariablen (sysvar)
Anzahl Samples	nn	Anzahl der aufgezeichneten Abtastungen
erster Eintrag	0 ...	Relativer Index des Datenelements mit dem ältesten aufgezeichneten Datenwert: Bei einmaliger Aufzeichnung: 0 Bei zyklischer Aufzeichnung: 0 max. Anzahl Daten
Stopp-Zeit	...	Interne Systemzeit am Ende der Aufzeichnung
Daten	...	Aufzeichnungsdaten, Anzahl an Messwerten =
...	...	Abtastungen * Anzahl Systemvariablen (svi1...svin)
Anzahl Testreihen	0-mm	Anzahl Messreihen (falls weitere im Speicher vorhanden). 0 oder nicht vorhanden, falls keine weiteren Messreihen
weitere Messreihen...		(siehe oben)



ACHTUNG!

Sollte in dem definierten Speicher (Array oder DYNMEM) nicht genügend Platz für die definierte Anzahl Messungen (= Parameter anz > 0) zur Verfügung stehen, tritt zur Programmlaufzeit der Fehler 171 „Array zu klein“ bzw. der Fehler 194 „DYNMEM zu klein“ auf. Es wird deshalb empfohlen mit TESTSTART 0 zu arbeiten, welches maximal den zur Verfügung gestellten Speicher auffüllt.

Beschreibung TESTSETDEST konfiguriert, wo aufgezeichnete Daten abgespeichert werden. Als Ablageort stehen die beiden folgenden Möglichkeiten zur Verfügung:

- Ein (ausreichend großes) Array, welches am Programmanfang angelegt wurde (DIM ...).
- Der freie Speicherbereich. Dieser wird durch das Schlüsselwort DYNMEM ausgewählt.

__ Befehlsreferenz __

Falls der Befehl TESTSETP nach TESTSETDEST ausgeführt wird, so gilt der bei TESTSETP definierte Speicherort. Prinzipiell wird immer der letzte, vor dem TESTSTART Befehl konfigurierte Speicherort verwendet. Falls weder TESTSETDEST noch TESTSETP ausgeführt wurde, so gilt DYNMEM als Grundeinstellung.

Portabilität Der Befehl wird ab MCO 5.00 unterstützt.

Zum Lesen der resultierenden Daten mit APOSS sollte eine aktuelle APOSS-Version benutzt werden. Denn aktuelle APOSS-Versionen bieten eine einfache Weise, um zwischen Arrays und DYNMEM zu wählen. Ältere Versionen von APOSS können dann benutzt werden, wenn die Array-Nummer auf den speziellen Wert von 65535 gesetzt wird.

Befehlsgruppe SYS

Querverweise TESTSETP, TESTSETINDEX, TESTSETTIME, TESTSETTYPE, TESTSTART, TESTSTOP

Syntax-Beispiel

```

DIM testdata[6011] // Array für die Datenspeicherung anlegen
// Systemvariablen für die Datenaufzeichnung festlegen:
// Slave-Istposition, Slave-Sollposition, Master-Position,
// Schleppfehler, Synchronisationsfehler, Slave-Geschwindigkeit
TESTSETINDEX 4096, 4097, 4105, 4101, 4207, 4186
TESTSETDEST testdata // Array als Speicherort festlegen
TESTSTART 1000 // Aufzeichnung von 1000 Abtastungen starten
SYNCP // Positionssynchronisation starten

```

Syntax-Beispiel

```

// Systemvariablen für die Datenaufzeichnung festlegen:
// Slave-Istposition, Slave-Sollposition, Schleppfehler, Geschwindigkeit
TESTSETINDEX 4096, 4097, 4101, 4186
TESTSETDEST DYNMEM // Freien Speicher für Datenaufzeichnung verwenden
NOWAIT ON // Nicht warten bei einem Positionierbefehl
DEFORIGIN // Nullpunkt setzen
VEL 100 // Maximale Geschwindigkeit
TESTSTART 0 // Aufzeichnung starten (bis TESTSTOP oder DYNMEM voll)
POSA 100000 // Positionierung starten
DELAY 500 // 500 ms warten
VEL 20 // Geschwindigkeit reduzieren
POSA 100000 // Positionierung mit neuer Geschwindigkeit fortsetzen
NOWAIT OFF // Auf das Ende der Positionierung warten
DELAY 200 // 200 ms warten
TESTSTOP 0 0 // Aufzeichnung beenden

```

□ TESTSETINDEX

Kurzinfo	Systemvariablen für die Datenaufzeichnung festlegen.
Syntax	TESTSETINDEX svi1, svi2, svi3, svi4, ..., svin
Parameter	svi1...svin: Indizes der maximal 20 Systemvariablen (sysvar[...]), deren Werte aufgezeichnet werden sollen.
	ACHTUNG! Es können maximal 20 Systemvariablen für die gleichzeitige Erfassung definiert werden. Falls mehr Variablen bei TESTSETINDEX angegeben werden, tritt zur Programmlaufzeit der Fehler 195 „Zu viele Index bei TESTSETINDEX definiert“ auf.
	ACHTUNG! Bitte beachten Sie, dass die Größe des Speicherbereichs (Array oder DYNMEM) für die Datenaufzeichnung maßgeblich von der Anzahl der konfigurierten Systemvariablen multipliziert mit der Anzahl der Abtastungen abhängt.
	ACHTUNG! Falls mehr als 3 Systemvariablen für die Erfassung konfiguriert werden, wird eine APOSS PC-Software mit mindestens Version MCO 5.00 für das Auslesen und Anzeigen der Daten mit <i>Tools</i> → <i>Oszilloskop (TESTSETP)</i> benötigt.
	ACHTUNG! Falls eine Aufzeichnung (durch TESTSTOP oder die definierte Anzahl an Abtastungen) beendet wurde und ein TESTSETINDEX Befehl vor einem weiteren TESTSTART ausgeführt wird, wird der konfigurierte Speicher (Array oder DYNMEM) wieder vom Anfang gefüllt. Dies bedeutet, dass die letzte(n) Aufzeichnungen im identischen Datenspeicher überschrieben werden.
Beschreibung	<p>TESTSETINDEX definiert die bei der Datenaufzeichnung zu erfassenden Systemvariablen (sysvar[...]). Der TESTSTART Befehl legt anschließend die Anzahl an Abtastungen fest und startet die konfigurierte Aufzeichnung. Die aufgezeichneten Daten können anschließend mit <i>Tools</i> → <i>Oszilloskop (TESTSETP)</i> ausgelesen und visualisiert werden.</p> <p>Der TESTSETINDEX Befehl ist eine neuere APOSS-Erweiterung und bietet eine bessere Flexibilität als der schon vorhandene TESTSETP Befehl. Es wird deshalb empfohlen bei neuen Applikationsprogrammen den TESTSETINDEX Befehl zu verwenden. TESTSETINDEX bietet insbesondere die Möglichkeit mehr als drei Systemvariablen gleichzeitig aufzuzeichnen. Die vollständige Konfiguration einer Datenaufzeichnung geschieht über die Befehle TESTSETINDEX, TESTSETDEST, TESTSETTYPE, TESTSETTIME.</p> <p><i>Tools</i> → <i>Oszilloskop (Single Shot)</i> bietet die Möglichkeit automatisch eine interne TESTSETINDEX Konfiguration zur Programmlaufzeit vorzunehmen, ohne dass TESTSET... Befehle im APOSS Applikationsprogramm vorhanden sind. Damit sind ebenfalls Aufzeichnungen im 1 ms Raster möglich. Der Unterschied und Vorteil bei der Integration der Befehle direkt im Programm ist jedoch, dass die Aufzeichnung exakt bei einem bestimmten Codeteil ausgeführt wird.</p> <p><i>Tools</i> → <i>Oszilloskop (Free Run)</i> bietet die Möglichkeit interne System- und Programmvariablen online permanent aufzuzeichnen. Die Aufzeichnungsrate des Free Run Oszilloskop ist jedoch durch verschiedene Faktoren, wie die verwendete Schnittstelle, die Datenanzahl pro Abtastung oder die Firmwareversion der Steuerung beschränkt. Die minimale Abtastrate ist 10 ms, wenn die eingebaute RS485 benutzt wird.</p>
Portabilität	Der Befehl ist ab MCO 5.00 verfügbar.
Befehlsgruppe	SYS

__ Befehlsreferenz __

Querverweise SYSVAR, TESTSTART, TESTSETDEST, TESTSETTIME, TESTSETTYPE, TESTSTOP, Oszilloskop (TESTSETP)

Syntax-Beispiel

```
// Systemvariablen für die Datenaufzeichnung festlegen:
// Slave-Istposition, Slave-Sollposition, Master-Position,
// Schleppfehler, Synchronisationsfehler, Slave-Geschwindigkeit
TESTSETINDEX 4096, 4097, 4105, 4101, 4207, 4186
TESTSETTIME 20 // Daten im 20 ms Raster aufzeichnen
TESTSETDEST DYNMEM // Datenablage im freien Speicher
TESTSETTYPE 1 // Zyklische Datenaufzeichnung (falls Speicher nicht ausreicht)
TESTSTART 0 // Aufzeichnung starten (bis TESTSTOP)
SYNCP // Positionssynchronisation starten
DELAY 5000 // 5 Sekunden im Synchron-Modus
CVEL 10 // 10% der Maximal-Geschwindigkeit setzen
CSTART X(1) // Im Drehzahlmodus weiterfahren
DELAY 2000 // ... für 2 Sekunden
MOTOR STOP X(1) // Antrieb abbremesen
DELAY 100 // 100 ms warten
TESTSTOP 0 0 // Aufzeichnung stoppen
```

□ TESTSETP

Kurzinfo Aufzeichnungsdaten für Testfahrt festlegen.

Syntax TESTSETP ms wi1 wi2 wi3 arrayname

Parameter

ms = Abstand in Millisekunden zwischen zwei Messungen

vi 1...3 = Indizes der drei Werte, die aufgezeichnet werden sollen. Es gelten die Vereinbarungen für das Systemarray. Es werden immer drei Werte aufgezeichnet.

arrayname = Name des Arrays, das für die Aufzeichnung verwendet wird.

Array-Format Innerhalb des Arrays werden die Werte wie folgt gespeichert (alle Werte 4 Byte):

Bezeichnung	Inhalt	Bedeutung
Version	000	Version der Datenstruktur
ms	1	Abstand zwischen zwei Messungen in ms
Anzahl Indizes	3	Anzahl definierter Indizes (immer 3 bei TESTSETP)
svi 1	i	Index der 1. aufgezeichneten Systemvariablen (SYSVAR)
svi 2	i	Index der 2. aufgezeichneten Systemvariablen (SYSVAR)
svi 3	i	Index der 3. aufgezeichneten Systemvariablen (SYSVAR)
Anzahl Samples	nn	Anzahl der aufgezeichneten Abtastungen
erster Eintrag	0 ...	Relativer Index des Datenelements mit dem ältesten aufgezeichneten Datenwert: Bei einmaliger Aufzeichnung: 0 Bei zyklischer Aufzeichnung: 0 max. Anzahl Daten
Stopp-Zeit	...	Interne Systemzeit am Ende der Aufzeichnung
Daten	...	Aufzeichnungsdaten, Anzahl an Messwerten =
...	...	Anzahl Samples * Anzahl Variablen (svi1...svin)
Anzahl Testreihen	0-mm	Anzahl Messreihen (falls weitere im Array vorhanden). 0 oder nicht vorhanden, falls keine weiteren Messreihen im gleichen Array
weitere Messreihen...		(siehe oben)

ACHTUNG!

Bitte beachten Sie, dass die Größe des Arrays oder freien Speichers (DYNMEM) für die Aufzeichnung ausreicht. Für den Datenheader mit den Informationen zu der Aufzeichnung werden 10 Elemente benötigt. Hinzu kommt ein Element für die

Anzahl der Testreihen und 3 Elemente pro Abtastpunkt. Falls 100 Datenwerte aufgezeichnet werden sollen (d.h. TESTSTART 100) ist somit mindestens eine Array-Größe von 311 Elementen oder im Falle von DYNMEM mindestens $311 \times 4 \text{ Bytes} = 1'244 \text{ Bytes}$ freier Speicher notwendig.

**ACHTUNG!**

Bitte beachten Sie, dass bei älteren Firmware-Versionen (< MCO 5.00) ein etwas kleineres Array ausreichend war. Der Datenheader bestand aus nur 7 Elementen. Es reichte deshalb minimal eine Arraygröße mit einem Overhead von 7 Elementen (oder 8, falls mehr als eine Testreihe enthalten war) aus. Bei neueren Firmware-Versionen (>MCO 5.00) muss ein Overhead von mindestens 11 Elementen bei der Berechnung der notwendigen Arraygröße berücksichtigt werden.

**ACHTUNG!**

Falls das Schlüsselwort DYNMEM benutzt und die Daten im freien Speicher abgelegt werden sollen, muss auf der Steuerung mindestens die Firmware-Version MCO 5.00 installiert sein.

Beschreibung

TESTSETP konfiguriert die Datenaufzeichnung. Es wird festgelegt, welche Systemvariablen wie oft aufgezeichnet werden sollen und in welchen Speicherbereich. Die Abtastperiode zwischen den Messungen kann minimal auf 1 ms eingestellt werden. Der TESTSTART Befehl legt die Anzahl an Abtastungen fest und startet die konfigurierte Aufzeichnung. Mindestens die Befehle TESTSETP und TESTSTART müssen hierfür in einem APOSS Applikationsprogramm enthalten sein. Die aufgezeichneten Daten können nach der Ausführung der Befehle mit *Tools* → *Oszilloskop* (TESTSETP) ausgelesen und visualisiert werden.

Die Aufzeichnungsdaten können in einem Array oder in den so genannten DYNMEM-Speicherbereich abgelegt werden. Das Schlüsselwort DYNMEM bezeichnet den freien Speicherbereich, der weder vom Programm noch Variablen benötigt wird. Falls der verfügbare DYNMEM-Bereich (ebenso wie bei einem Array) nicht ausreichend groß ist, um die aufgezeichneten Daten aufzunehmen, tritt zur Programmlaufzeit eine entsprechende Fehlermeldung auf. Bei der Verwendung von DYNMEM wird deshalb empfohlen TESTSTART 0 zu nutzen. In diesem Fall wird automatisch maximal der verfügbare DYNMEM Speicher genutzt, ohne dass ein Überlauf und Fehlermeldung auftreten könnte. Die Verwendung von DYNMEM erfordert eine Version ab MCO 5.00.

Der TESTSETINDEX Befehl ist eine neuere APOSS-Erweiterung und bietet eine bessere Flexibilität als der TESTSETP Befehl. Es wird deshalb empfohlen bei neuen Applikationsprogrammen den TESTSETINDEX Befehl zu verwenden. TESTSETINDEX steht bei Steuerungen ab MCO 5.00 zur Verfügung und bietet die Möglichkeit mehr als drei Systemvariablen gleichzeitig aufzuzeichnen. Die vollständige Konfiguration einer solchen Testaufzeichnung geschieht über die Befehle TESTSETINDEX, TESTSETDEST, TESTSETTYPE, TESTSETTIME. Diese Befehle haben auch bereits bei einer TESTSETP Konfigurationen Auswirkung und können die entsprechenden Einstellungen von dieser überschreiben. Es gelten jeweils die zuletzt (vor TESTSTART) ausgeführten Einstellungen.

Tools → *Oszilloskop* (Single Shot) bietet die Möglichkeit automatisch eine interne TESTSETINDEX Konfiguration zur Programmlaufzeit vorzunehmen, ohne dass TESTSET... Befehle im APOSS Applikationsprogramm vorhanden sind. Damit sind ebenfalls Aufzeichnungen im 1 ms Raster möglich. Der Unterschied und Vorteil bei der Integration der Befehle direkt im Programm ist jedoch, dass die Aufzeichnung exakt bei einem bestimmten Codeteil ausgeführt wird.



__ Befehlsreferenz __

Tools → *Oszilloskop (Free Run)* bietet die Möglichkeit interne System- und Programmvariablen online permanent aufzuzeichnen. Die Aufzeichnungsrate des Free Run Oszilloskop ist jedoch durch verschiedene Faktoren beschränkt, wie die verwendete Kommunikationsschnittstelle, die Anzahl der Systemvariablen, die pro Abtastung aufgezeichnet werden oder die Firmwareversion der Steuerung. Die minimale Abtastrate ist 10 ms bei Verwendung des Onboard RS485.

Mit *Tools* → *Oszilloskop (Tune)* können Sie eine Testfahrt durchführen, die Soll- und Istposition, Geschwindigkeit, Beschleunigung und Strom aufzeichnet und deren Ergebnis Sie in einer Grafik sehen können. Diese Funktion ist eine Art vorkonfigurierte, auf Menüs basierende Datenaufzeichnung für einen spezifizierten Bewegungsvorgang.

Portabilität Ab MCO 5.00 wurde der Befehl TESTSETP durch vier neue Befehle mit viel mehr Möglichkeiten ersetzt. Der bisherige Befehl TESTSETP funktioniert weiterhin, aber es wird empfohlen die neuen Befehle zu benutzen.

Befehlsgruppe SYS

Querverweis TESTSTART, DIM, SYSVAR,
TESTSETINDEX, TESTSETDEST, TESTSETTIME, TESTSETTYPE,
Oszilloskop (Tune), *Oszilloskop (TESTSETP)*

Syntax-Beispiel DIM tstfahrtarray[311] // Array mit 311 Elementen für Datenspeicherung
// Systemvariablen, Abtastrate und Speicher für die Datenaufzeichnung festlegen:
// Slave-Pos., Master-Pos., Schleppfehler im 3ms Takt
TESTSETP 3 0X1001 0X1009 0X1005 tstfahrtarray
TESTSTART 100 // Aufzeichnung von 100 Datenwerten starten
POSR 100000 // Positionierung starten

Syntax-Beispiel DIM tstfahrtarray[311] // Array mit 311 Elementen
// Identisches Beispiel mit dezimalem Index der Systemvariablen
TESTSETP 3 4096 4105 4101 tstfahrtarray

Syntax-Beispiel // Identisches Beispiel mit Datenspeicherung im DYNMEM
TESTSETP 3 4096 4105 4101 DYNMEM

□ TESTSETTIME

Kurzinfo	Konfiguriert die Abtastperiode für die Datenaufzeichnung.
Syntax	TESTSETTIME ms
Parameter	ms = Abtastperiode in Millisekunden Falls dieser Befehl nicht vor einem TESTSTART benutzt wird, wird der Default-Wert von 1 ms benutzt.
Beschreibung	<p>Mit diesem Befehl wird die Abtastperiode für Datenaufzeichnungen festgelegt. (Dies war der erste Parameter des früheren TESTSETP Befehls.) Entsprechend dem konfigurierten Wert in Millisekunden werden in diesem Zeitraster die Werte aller konfigurierten Systemvariablen erfasst und in dem konfigurierten Speicher (Array oder DYNMEM) abgelegt.</p> <p>Die minimale Abtastrate ist 1 Millisekunde. Dies bedeutet, dass jede Millisekunde die Datenerfassung aller konfigurierten Systemvariablen stattfindet.</p> <p>Falls der Befehl TESTSETP nach TESTSETTIME ausgeführt wird, so gilt die innerhalb des Befehls TESTSETP definierte Abtastperiode. Prinzipiell besitzt immer die letzte, vor dem TESTSTART Befehl gesetzte Einstellung Gültigkeit. Falls weder TESTSETTIME noch TESTSETP vor dem TESTSTART ausgeführt wurden, wird als Grundeinstellung eine Abtastrate von 1 ms verwendet.</p>
Portabilität	Der Befehl ist ab MCO 5.00 verfügbar.
Befehlsgruppe	SYS
Querverweise	TESTSETP, TESTSETINDEX, TESTSETDEST, TESTSETTYPE, TESTSTART, TESTSTOP
Syntax-Beispiel	<pre> TESTSETDEST DYNMEM // Daten in DYNMEM ablegen TESTSETTYPE 0 // Eine einmalige Aufzeichnung konfigurieren TESTSETTIME 5 // Abtastperiode auf 5 ms setzen // Systemvariablen für die Datenaufzeichnung festlegen: // Aktuelle Pos., Schleppfehler, Eingang 1-8, Ausgang 1-8 TESTSETINDEX 4096, 4101, 0x01220202, 0x0122020A DEFORIGIN // Nullpunkt auf aktuelle Position setzen TESTSTART 0 // Aufzeichnung starten (bis TESTSTOP oder DYNMEM voll) VEL 10 // 10% der Maximal-Geschwindigkeit ACC 10 // 10% der Maximal-Beschleunigung DEC 50 // 50% der Maximal-Bremsrampe POSA 100000 // Positionierung starten (und zu Ende ausführen) DELAY 200 // 200 ms warten TESTSTOP 0 0 // Aufzeichnung beenden </pre>



□ TESTSETTYPE

Kurzinfo	Definiert, ob eine einmalige oder zyklische Datenaufzeichnung ausgeführt werden soll.
Syntax	TESTSETTYPE Typ
Parameter	<p>Typ: 0 = Einmalige Aufzeichnung (= Grundeinstellung) 1 = Zyklische Aufzeichnung</p> <p>Wenn kein Typ gesetzt ist, bevor ein TESTSTART ausgeführt wird, dann wird Default (einmalige Aufzeichnung) benutzt.</p>
	<p> ACHTUNG! Eine zyklische Aufzeichnung wird auch fortgesetzt, wenn das Programm beendet ist oder durch einen Fehler abbricht. Die Aufzeichnung wird nur durch einen TESTSTOP-Befehl oder den Benutzerabbruch (= Drücken der [Esc]-Taste) beendet.</p> <p> ACHTUNG! Bei der zyklischen Aufzeichnung wird der Speicher immer wieder überschrieben, sobald die Speichergrenze oder die konfigurierte Anzahl an Abtastungen erreicht ist. Die effektiv gespeicherte Anzahl an Abtastungen hängt somit von der vorhandenen Speichergröße (TESTSTART 0) oder der definierten Anzahl an Abtastungen (TESTSTART-Wert > 0) ab.</p>
Beschreibung	<p>Dieser Befehl konfiguriert, ob die Datenaufzeichnung einmalig entsprechend den weiteren Einstellungen ausgeführt wird oder ob diese zyklisch endlos im Hintergrund läuft bis ein TESTSTOP ausgeführt wird.</p> <p>Falls die zyklische Ausführung mit TESTSTART 0 aktiviert wurde, wird der gesamte zur Verfügung gestellte Speicher (Array oder DYNMEM) genutzt und immer wieder überschrieben. Falls mit einer definierten Anzahl an Samples gestartet wurde (TESTSTART-Wert > 0), so wird nur der entsprechende Speicherbereich genutzt und dieser immer wieder überschrieben.</p> <p>Eine einmalige Aufzeichnung (TESTSETTYPE 0) ist beendet, sobald die definierte Anzahl Abtastungen (TESTSTART Wert > 0) erreicht ist ... der definierte Speicher (Array oder DYNMEM) komplett gefüllt ist (TESTSTART 0) ... der Befehl TESTSTOP ausgeführt wird ... die Programmabarbeitung durch den Benutzer mit der Esc-Taste abgebrochen wird</p> <p>Eine zyklische Aufzeichnung (TESTSETTYPE 1) ist erst beendet, wenn der Befehl TESTSTOP ausgeführt wird ... die Programmabarbeitung durch den Benutzer mit der [Esc]-Taste abgebrochen wird</p> <p>Die zyklische Aufzeichnung ist sehr gut für Debugging-Zwecke bei der Fehlersuche geeignet. Insbesondere bei Anlagenproblemen, die nur sporadisch auftreten. Die zyklische Aufzeichnung in Kombination mit dem DYNMEM Speicher und TESTSTART 0 erlaubt eine maximale Anzahl von Daten permanent endlos im Hintergrund zu erfassen. Sobald ein Fehler, ein definierter Anlagenzustand oder ein äußeres Ereignis (= Interrupt) auftritt, kann die Aufzeichnung gezielt im Programmcode beendet werden. Die bis zu diesem Zeitpunkt erfassten Daten können mit <i>Tools</i> → <i>Oszilloskop</i> (TESTSETP) ausgelesen, analysiert und für die Dokumentation gespeichert werden.</p>
Portabilität	Der Befehl ist ab MCO 5.00 verfügbar.
Befehlsgruppe	SYS
Querverweise	TESTSETP, TESTSETINDEX, TESTSETDEST, TESTSETTIME, TESTSTART, TESTSTOP

__ Befehlsreferenz __

```

Syntax-Beispiel  SYNCP                // Positionssynchronisation starten
                    // Systemvariablen für die Datenaufzeichnung festlegen:
                    // Slave-Pos., Master-Pos., Schleppfehler, Synchronisationsfehler
                    // Slave Geschwindigkeit, Skalierte Differenz von Master- zu Slave-Geschwindigkeit
                    TESTSETINDEX 4096, 4105, 4101, 4207, 4185, 4236
                    TESTSETTYPE 1          // Zyklische Aufzeichnung konfigurieren
                    WAITI 1 ON            // Warten bis Signal an Eingang 1
                    TESTSTART 0          // Aufzeichnung starten (bis TESTSTOP)
                    WAITI 1 OFF          // Warten bis Signal an Eingang 1 nicht mehr anliegt
                    TESTSTOP 0 0        // Aufzeichnung stoppen

Syntax-Beispiel // Eine Fehler-Behandlungsroutine definieren
                    ON ERROR GOSUB errhandler
                    // Alle notwendigen Konfigurationen für die Datenaufzeichnung ausführen:
                    TESTSETDEST DYNMEM    // DYNMEM für die Datenaufzeichnung nutzen
                    TESTSETTYPE 1        // Zyklische Aufzeichnung konfigurieren
                    // Systemvariablen für die Datenaufzeichnung festlegen:
                    // Aktuelle Pos., Schleppfehler, Eingang 1-8, Ausgang 1-8
                    TESTSETINDEX 4096, 4101, 0x01220202, 0x0122020A
                    TESTSTART 0          // Aufzeichnung starten (bis TESTSTOP)

                    // Programm-Hauptschleife
                    endless:
                        // Eigentlichen Anwendungscode ausführen
                    GOTO endless

                    // Unterprogramm-Bereich
                    SUBMAINPROG
                    SUBPROG errhandler    // Fehlerbehandlungsroutine
                        TESTSTOP 0 0      // Aufzeichnung stoppen
                        EXIT // Programmabarbeitung abbrechen
                    RETURN // Ende der Fehlerbehandlungsroutine
                    ENDPROG // Ende des Unterprogramm-Bereichs

```



□ TESTSTART

Kurzinfo	Datenaufzeichnung starten.
Syntax	TESTSTART n
Parameter	<p>n = 0: Die maximale Anzahl von Abtastpunkten wird aufgezeichnet, die in dem zur Verfügung stehenden Speicher (Array oder DYNMEM) Platz findet.</p> <p>n > 0: Anzahl der aufzuzeichnenden Abtastpunkte</p>
	<p> ACHTUNG! Sollte in dem definierten Speicher (Array oder DYNMEM) nicht genügend Platz für die definierte Anzahl Abtastpunkte (= Parameter n > 0) zur Verfügung stehen, tritt zur Programmlaufzeit der Fehler 171 „Array zu klein“ bzw. der Fehler 194 „DYNMEM zu klein“ auf. Es wird deshalb empfohlen mit TESTSTART 0 zu arbeiten, welches den zur Verfügung gestellten Speicher maximal auffüllt.</p> <p> ACHTUNG! Wenn eine Aufzeichnung angehalten wurde (durch TESTSTOP, durch Drücken der [Esc]-Taste, oder bei einer Single-Shot-Aufzeichnung oder durch Erreichen der definierten Anzahl von Abtastungen) und ein TESTSETINDEX Befehl vor einem weiteren TESTSTART ausgeführt wird, dann wird der konfigurierte Speicher (Array oder DYNMEM) wieder von Anfang gefüllt. Dies beutet, dass die letzte(n) Aufzeichnungen im identischen Datenspeicher überschrieben werden. Falls ein weiterer TESTSTART jedoch ohne erneute TESTSETINDEX Definition ausgeführt wird, so werden die neu erfassten Daten an die letzte Datenaufzeichnung angehängt. Es stehen somit in dem konfigurierten Speicher mehrere Aufzeichnungen zur späteren Auswertung zur Verfügung.</p>
Beschreibung	<p>Mit diesem Befehl wird die Datenaufzeichnung entsprechend den zuletzt, mit den Befehlen TESTSETP, TESTSETINDEX, TESTSETDEST, TESTSETTIME, TESTSETTYPE definierten konfigurierten Einstellungen gestartet. Die aufgezeichneten Daten können anschließend mit Tools → <i>Oszilloskop (TESTSETP)</i> ausgelesen und grafisch dargestellt werden.</p> <p>Falls vor dem TESTSTART Befehl nicht alle Parameter vollständig konfiguriert wurden, kommen die folgenden Grundeinstellungen für die Datenaufzeichnung zum Einsatz:</p> <ul style="list-style-type: none"> – Abtastperiode: 1 ms – Aufzeichnungstyp: Einmalige Aufzeichnung – Datenspeicher: DYNMEM – Aufgezeichnete Systemvariablen: 4096 (ACTPOS), 4097 (COMPOS), 4324 (ACTCURR) <p>Es kann auch mehr als eine Aufzeichnung in einem Array gesammelt werden. Zum Beispiel ist es möglich, eine Aufzeichnung von 1000 Proben zu starten, dann die Aufzeichnung zu stoppen und danach eine weitere mit 500 Proben starten. Falls dann die Daten mit APOSS ausgelesen werden, sind zwei Aufzeichnungen zu sehen und können betrachtet werden. Wenn ein Neustart ganz von vorne gewünscht wird, muss ein neuer TESTSETINDEX Befehl ausgeführt werden.</p>
Portabilität	Erweiterungen, die die Oszilloskop Funktionen betreffen sind ab MCO 5.00 verfügbar.
Befehlsgruppe	SYS
Querverweis	<p>Tools → <i>Oszilloskop (TESTSETP)</i> TESTSETP, TESTSETINDEX, TESTSETDEST, TESTSETTIME, TESTSETTYPE, TESTSTOP</p>

__ Befehlsreferenz __

Syntax-Beispiel	<pre> SYNCP // Positionssynchronisation starten // Systemvariablen für die Datenaufzeichnung festlegen: // Slave-Pos., Master-Pos., Schleppfehler, Synchronisationsfehler TESTSETINDEX 4096, 4105, 4101, 4207 WAITI 1 ON // Warten bis Eingang 1 aktiv TESTSTART 200 // Aufzeichnung starten (200 Messungen) </pre>
Syntax-Beispiel	<pre> NOWAIT ON // Nicht warten bis Position erreicht ist DEFORIGIN // Nullpunkt auf aktuelle Position setzen // Erfassungsdaten konfigurieren: // Istpos., Sollpos., Schleppfehler, Geschwindigkeit, Index-Pos., Eing.1-8 TESTSETINDEX 4096, 4097, 4101, 4186, 4098, 0x01220202 TESTSTART 0 // Aufzeichnung starten (bis TESTSTOP oder DYNMEM voll) VEL 50 // 50% der Maximalgeschwindigkeit POSA 100000 // Positionierung mit Geschwindigkeit 50% starten WAITP 20000 // Warten bis Position 20000 erreicht ist VEL 100 // Geschwindigkeit auf 100% erhöhen POSA 100000 // Positionierung mit neuer Geschwindigkeit fortsetzen NOWAIT OFF // Warten bis Positionierung zu Ende DELAY 200 // 200 ms warten TESTSTOP 0 0 // Aufzeichnung stoppen </pre>




□ TESTSTOP

Kurzinfo	Datenaufzeichnung beenden.
Syntax	TESTSTOP methode param
Parameter	<p>methode Verfahren zum Beenden der Datenaufzeichnung 0 = sofortiges Ende der Datenaufzeichnung Aktuell werden keine weiteren Stopp-Methoden unterstützt.</p> <p>param Parameter in Abhängigkeit von der Stopp-Methode. Für die implementierten Methoden aktuell nicht verwendet.</p>
	<p> ACHTUNG! Aktuell wird nur die Methode 0 unterstützt. Es existiert somit nur die folgende zulässige Befehls- /Parameterkombination: TESTSTOP 0 0</p>
Beschreibung	Dieser Befehl stoppt die Datenaufzeichnung.
	<p> ACHTUNG! Falls eine Aufzeichnung durch TESTSTOP beendet wurde und ein TESTSETINDEX Befehl vor einem weiteren TESTSTART ausgeführt wird, wird der konfigurierte Speicher (Array oder DYNMEM) wieder von Anfang gefüllt. Dies beudet, dass die letzte(n) Aufzeichnungen im identischen Datenspeicher überschrieben werden.</p>
	<p> ACHTUNG! Der Befehl TESTSTOP ist bei zyklischen Aufzeichnungen in der Regel vor dem Programmende empfohlen oder Programmabbruch im Fehlerfall empfohlen. Ansonsten läuft die Aufzeichnung auch nach Programmende weiter.</p>
Portabilität	<p>Der Befehl ist ab MCO 5.00 verfügbar.</p> <p>Wenn Werte mit einer neueren APOSS-Version ausgelesen werden, wird die Aufzeichnung korrekt dargestellt (time wise).</p> <p>Ab MCO 5.00 wird die Aufzeichnung auch gestoppt, wenn die [Esc]-Taste in APOSS gedrückt wird.</p>
Befehlsgruppe	SYS
Querverweise	TESTSETP, TESTSETINDEX, TESTSETDEST, TESTSETTIME, TESTSETTYPE, TESTSTART
Syntax-Beispiel	<pre>// Eine Fehler-Behandlungsroutine definieren ON ERROR GOSUB errhandler // Systemvariablen für die Datenaufzeichnung festlegen: // Slave-Pos., Master-Pos., Schleppfehler, Synchronisationsfehler // Slave-Geschwindigkeit, Skalierte Differenz Master- zu Slave Geschwindigkeit TESTSETINDEX 4096, 4105, 4101, 4207, 4185, 4236 TESTSETTYPE 1 // Zyklische Aufzeichnung konfigurieren WAITI 1 ON // Warten bis Signal an Eingang 1 SYNCP // Positionssynchronisation starten TESTSTART 0 // Aufzeichnung starten (bis TESTSTOP) WAITI 1 OFF // Warten bis Signal an Eingang 1 entfernt MOTOR STOP // Antrieb abbremsen TESTSTOP 0 0 // Aufzeichnung stoppen</pre>

__ Befehlsreferenz __

Syntax-Beispiel

```
// Unterprogramm-Bereich
SUBMAINPROG
SUBPROG errhandler // Fehlerbehandlungsroutine
  TESTSTOP 0 0 // Sicherstellen, dass die Aufzeichnung stoppt
  EXIT // Programm beenden
RETURN // Ende der Fehlerbehandlungsroutine
ENDPROG // Ende des Unterprogramm-Bereichs
NOWAIT ON // Nicht warten bis Position erreicht
DEFORIGIN // Nullpunkt auf aktueller Position setzen
// Systemvariablen für die Datenaufzeichnung festlegen:
// IST-Pos., SOLL-Pos., Schleppfehler, Geschwindigkeit, Index-Pos., Eing.1-8
TESTSETINDEX 4096, 4097, 4101, 4186, 4098, 0x01220202
TESTSTART 0 // Aufzeichnung starten (bis TESTSTOP oder DYNMEM
voll)
VEL 20 // 20% der Maximalgeschwindigkeit
POSA 100000 // Positionierung starten
WAITP 50000 // Warten bis Position 50000 erreicht
VEL 100 // Geschwindigkeit auf 100% setzen
POSA 100000 // Positionierung mit neuer Geschwindigkeit fortsetzen
NOWAIT OFF // Warten bis Positionierung beendet
DELAY 200 // 200 ms warten
TESTSTOP 0 0 // Aufzeichnung stoppen
```

Beispiel Dieses Beispiel sammelt Daten kontinuierlich bis eine besondere Bedingung erreicht ist und die Aufzeichnung gestoppt wird. Wenn die Daten ausgelesen werden, erhält man die letzten abgetasteten Werte bevor gestoppt wurde. Die Menge der Werte hängt vom verfügbaren dynamischen Speicher ab.

```
#define PFG_ACTPOS 0x01250001
#define PFG_COMPOS 0x01250002
#define PFG_VCMDSIGNED 0x012500B5
...
TESTSETINDEX PFG_ACTPOS, PFG_COMPOS, PFG_VCMDSIGNED
TESTSETDEST DYNMEM // dynamischen Speicher für die Aufzeichnung wählen
TESTSETTIME 5 // Aufzeichnungsintervall von 5 ms wählen
TESTSETTYPE 1 // zyklisches Aufzeichnen wählen
TESTSTART 0
// Aufzeichnung starten und den ganzen verfügbaren dynamischen Speicher
nutzen
...
IF(whatever) THEN
  TESTSTOP 0 0 // Aufzeichnung stoppen
ENDIF
```

Diese Art der Aufzeichnung könnte auch benutzt werden ohne dass das Programm überhaupt geändert wird, durch Einsatz der Oszilloskop-Funktionen von APOSS. (See Single Shot Oszilloskop).



□ TIME

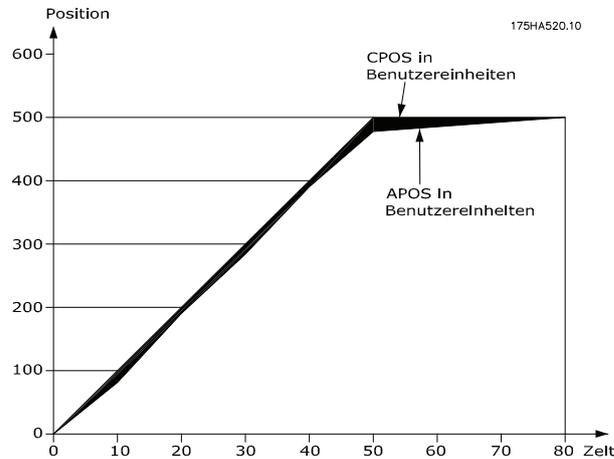
Kurzinfo	Systemzeit auslesen.	
Syntax	erg = TIME	
Rückgabewert	erg = Systemzeit in Millisekunden seit Einschalten	
	ACHTUNG!	Bitte beachten Sie, dass der Wert, wenn er MLONG erreicht hat, auf -MLONG wechselt.
Beschreibung	Mit dem TIME Befehl kann die interne Systemzeit ausgelesen werden. Der TIME Befehl eignet sich vor allem, um die Ausführungszeit einer Befehlssequenz oder Maschinentzykluszeiten zu berechnen.	
Befehlsgruppe	SYS	
Syntax-Beispiel	PRINT TIME	/* aktuelle Systemzeit ausgeben */
	timestop1 = TIME	/* aktuelle Systemzeit zwischenspeichern */
Programmbeispiel	ACC_01.M, DELAY_01.M, EXIT_01.M, GOSUB_01.M	

□ TRACKERR

Kurzinfo Aktuellen Schleppabstand einer Achse abfragen.

Syntax erg = TRACKERR

Rückgabewert erg = aktueller Schleppabstand der Achse n in BE

Beschreibung

Befehlsgruppe SYS

Querverweise APOS, CPOS, Par. 32-67 *max. tolerierter Positionsfehler*

Syntax-Beispiel PRINT TRACKERR /* aktuellen Schleppabstand von Achse 1 abfragen */

Beispiel POSA 500

```

WHILE(1) DO
{
PRINT "Sollposition", CPOS
PRINT "Istposition", APOS
PRINT "Fehler", TRACKERR
WAITT 10
}
ENDWHILE
Ausgabe:
Sollposition 100
Istposition 98
Fehler 2
Sollposition 200
Istposition 199
Fehler 1
Sollposition 300
Istposition 297
Fehler 3      ...
Sollposition 500
Istposition 500
Fehler 0
... und so weiter

```

Der dunkel markierte Bereich zwischen CPOS und APOS im Zeitintervall kann so mit TRACKERR ausgelesen werden. Um alle Fehler während der gesamten Positionierung auszulesen, sollte TRACKERR in einer Schleife ständig die Fehler verfolgen.

__ Befehlsreferenz __

□ USRSTAT

Kurzinfo	Setzt den CAN-User-Status.
Syntax	USRSTAT val
Parameter	val = Wert der gesetzt werden soll
Beschreibung	Setzt den User-Status (long), der über den CAN-Bus abgefragt werden kann.
Portabilität	Der Befehl ist ab MCO 5.00 verfügbar.
Befehlsgruppe	CAN
Syntax-Beispiel	USRSTAT 5 /* User-Status auf 5 setzen */

□ VEL

Kurzinfo	Geschwindigkeit für relative und absolute Bewegungen setzen.
Syntax	VEL v
	$\text{Sollgeschwindigkeit [U/Min]} = v * \frac{\text{Par. 32 - 80 Maximalgeschwindigkeit}}{\text{Par. 32 - 83 Geschwindigkeitsteiler}}$
Parameter	v = Normierter Geschwindigkeitswert
Beschreibung	<p>Mit dem VEL Befehl wird die Geschwindigkeit für die nächsten absoluten und relativen Positioniervorgänge und die maximal zulässige Geschwindigkeit für Synchronisationsvorgänge bestimmt. Der Wert bleibt solange gültig, bis mit einem weiteren VEL Befehl eine neue Geschwindigkeit gesetzt wird. Der zu übergebende Geschwindigkeitswert wird zu den Parametern 32-80 <i>Maximalgeschwindigkeit</i> und 32-83 <i>Geschwindigkeitsteiler</i> in Bezug gesetzt. Wenn der übergebene Geschwindigkeitswert gleich dem <i>Geschwindigkeitsteiler</i> ist, wird mit der in Parameter <i>Maximalgeschwindigkeit</i> festgelegten Drehzahl gefahren.</p> <p>ANMERKUNG: Die Geschwindigkeit des Slaves wird im Synchronisationsmodus auch durch den Befehl VEL begrenzt.</p>
	<p> ACHTUNG! Wurde vor einem Positionier- oder Synchronisierbefehl noch keine Geschwindigkeit definiert, wird mit dem in Par. 32-84 <i>Default-Geschwindigkeit</i> festgelegten Wert gefahren.</p> <p>Soll während des Positioniervorgangs die Geschwindigkeit geändert werden, ist dies bei NOWAIT ON möglich, wenn dem VEL Befehl noch einmal ein POSA auf die gewünschte Zielposition folgt.</p> <p>Die maximal zulässige Geschwindigkeit kann jederzeit mit dem Befehl VEL geändert werden, wenn dem Befehl VEL nochmals ein SYNCV, SYNCP oder SYNCM folgt.</p>
Befehlsgruppe	REL, ABS
Querverweise	ACC, POSA, POSR, NOWAIT Parameter: 32-80 <i>Maximalgeschwindigkeit</i>
Syntax-Beispiel	VEL 100 /* Geschwindigkeit 100 */
Programmbeispiel	VEL_01.M

□ VLTALARMSTAT

Kurzinfo	Gibt an, ob ein Alarm vorliegt oder nicht.
Syntax	VLTALARMSTAT
Beschreibung	Der Befehl VLTALARMSTAT gibt an, ob ein oder mehrere Alarmmeldungen vorhanden sind. Dabei gibt es zwei Möglichkeiten: 1<<3 oder 1<<6 abhängig davon, ob der Motor mit einem Reset wieder gestartet werden muss (Trip) oder nicht. Bit 3 = Alarm (oder mehrere) vorhanden Bit 6 = Trip Lock Alarm (oder mehrere) vorhanden
Befehlsgruppe	CON
Querverweise	VLERRCLR
Syntax-Beispiel	IF (VLTALARMSTAT) THEN PRINT " Alarm aktiv ", VLTALARMSTAT VLERRCLR ENDIF

□ VLTCONTROL

Kurzinfo	Setzt das VLT Steuerwort im Status MOTOR OFF
Syntax	VLTCONTROL Wert Steuerwort
Parameter	wert
Beschreibung	VLTCONTROL kann das VLT Steuerwort (STW) im MOTOR OFF Status setzen. Dieser Befehl kann benutzt werden, um das Steuerwort auf einen beliebigen Wert zu setzen. Der Anwender ist verantwortlich für den richtigen Wert (besonders beim Bit DATA VALID). Die Befehle verhalten sich wie folgt: Wenn VLTCONTROL zum ersten Mal benutzt wird, wird in den vom Anwender gesteuerten Modus gewechselt. In diesem Modus wird das Steuerwort überhaupt nicht beeinflusst. Der Anwender ist verantwortlich für das Steuerwort. So lange das System in diesem Modus ist, beeinflusst OUTAN nur den Sollwert und nicht das Steuerwort. Es kann dann nur mit einem MOTOR ON Befehl oder durch Starten eines neuen APOSS Programms in den normalen Modus zurückgekehrt werden.
Befehlsgruppe	CON
Querverweise	MOTOR OFF, MOTOR ON, OUTAN
Syntax-Beispiel	MOTOR OFF ... VLTCONTROL 0x047C // Antrieb einschalten OUTAN 0x1000 ... MOTOR ON // Anwendersteuerung des Steuerworts abschalten


□ VLTERRCLR

Kurzinfo	Löscht einen VLT-Alarm.
Syntax	VLTERRCLR
Beschreibung	Der Befehl VLTERRCLR löscht einen VLT-Alarm ohne Auswirkung auf einen vorhandenen Fehler der Optionskarte. Dieser Befehl kann an jeder Stelle im APOSS Programm benutzt werden.
Befehlsgruppe	CON
Querverweise	VLTALARMSTAT
Syntax-Beispiel	<pre>IF (VLTALARMSTAT) THEN PRINT " Alarm aktiv ", VLTALARMSTAT VLTERRCLR ENDIF</pre>

□ WAITAX

Kurzinfo	Warten bis Zielposition erreicht ist.
Syntax	WAITAX
Beschreibung	Der WAITAX Befehl ist für die Verwendung bei aktivem NOWAIT Modus vorgesehen. Damit wird im NOWAIT ON der Zustand erreicht, dass nach einem Positionierbefehl mit der weiteren Abarbeitung des Programms gewartet wird, bis die abgefragte Achse ihre Sollposition erreicht hat.
Befehlsgruppe	CON
Querverweise	NOWAIT ON/OFF, POSA, POSR, AXEND, STAT, WAITI
Syntax-Beispiel	<pre>WAITAX /* Warten bis die Achse die Bewegung beendet hat */ WAIT AX /* Alternative Schreibweise */</pre>
Programmbeispiel	WAIT_01.M, VEL_01.M

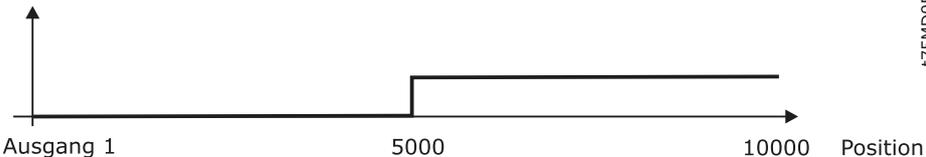
□ WAITI

Kurzinfo	Warten auf bestimmten Eingangszustand.	
Syntax	WAITI n s	
Parameter	n = Nummer des Eingangs	1 – 8 oder 16 – 33
	s = erwarteter Zustand:	ON = High-Signal anliegend OFF = Low-Signal anliegend
Beschreibung	Der WAITI Befehl wartet mit der weiteren Programmausführung, bis der entsprechende Eingang den gewünschten Signalzustand aufweist.	
	ACHTUNG!:	Wenn der erwartete Eingangszustand nie auftritt, bleibt das Programm an diesem Befehl „hängen“.
		Für das sichere Erkennen eines Signalzustandes ist eine minimale Signallänge notwendig!
		Informieren Sie sich im FC 300 Produkthandbuch und FC 300 Projektierungshandbuch über die Beschaltung und technischen Daten der Eingänge.
Befehlsgruppe	CON	
Querverweise	ON INT .. GOSUB, DELAY, WAITT, WAITAX	
Syntax-Beispiel	WAITI 4 ON	/* Warten bis an Eingang 4 High-Pegel anliegt */
	WAITI 4 1	/* 3 alternative Schreibweisen */
	WAIT I 4 ON	
	WAIT I 4 1	
Syntax-Beispiel	WAITI 6 OFF	/* Warten bis an Eingang 6 Low-Pegel anliegt */
	WAITI 6 0	/* 3 alternative Schreibweisen */
	WAIT I 6 OFF	
	WAIT I 6 0	
Programmbeispiel	WAIT_01.M	

□ WAITNDX

Kurzinfo	Warten bis die nächste Indexposition erreicht ist.	
Syntax	WAITNDX t	
Parameter	t = Timeout (maximale Wartezeit) in ms	
Beschreibung	Warten auf Index mit Überprüfung des Timeouts. Es wird solange gewartet, bis entweder der Index der Achse n gefunden oder die Zeit (Timeout) überschritten wurde.	
	ACHTUNG!:	Wenn die Zeit überschritten wurde, wird ein Fehler ausgelöst, der zum Beispiel mit einer ON ERROR Funktion ausgewertet werden kann.
	ACHTUNG!:	Der Befehl WAITNDX kann bei Einsatz von Absolutgebern (siehe Par. 32-00 <i>Inkrementalgeber Signaltyp</i>) nicht verwendet werden.
Befehlsgruppe	CON	
Querverweise	WAITI, WAITP, INDEX	
Syntax-Beispiel	CVEL 1 CSTART WAITNDX 10000 /* Wartet max. 10 s, dass die Achse die Indexposition erreicht */ OUT 1 1	

□ WAITP

Kurzinfo	Warten bis eine bestimmte Position erreicht ist.	
Syntax	WAITP p	
Parameter	p = absolute Position auf die gewartet wird	
Beschreibung	Der WAITP Befehl bewirkt, dass mit der weiteren Programmausführung gewartet wird, bis die Position p erreicht ist. Wenn aus der Geschwindigkeit und der Istposition hervorgeht, dass der Punkt p bereits überschritten wurde, wird der Befehl ebenfalls beendet.	
	ACHTUNG!: Aktive ON INT oder ON PERIOD Befehle können die Genauigkeit und Reproduzierbarkeit beeinflussen.	
Befehlsgruppe	CON	
Querverweise	DELAY, WAITI, WAITAX	
Syntax-Beispiel	<pre>NOWAIT ON POSA 10000 WAITP 5000 /* bis Position 5000 warten */ OUT 1 1 /* Ausgang 1 setzen */ NOWAIT OFF</pre>	
		

t75MD052

□ WAITT

Kurzinfo	Zeitverzögerung	
Syntax	WAITT t	
Parameter	t = Verzögerung in Millisekunden (maximal MLONG)	
Beschreibung	Mit dem WAITT Befehl können Sie eine definierte Programmverzögerung erzielen. Der Übergabeparameter gibt dabei die Verzögerungszeit in Millisekunden an.	
	ACHTUNG!: Wenn während der Zeitverzögerung ein Interrupt auftritt, wird nach der Abarbeitung der Interrupt-Prozedur der komplette Wartevorgang von neuem begonnen. Daher sollte im Allgemeinen statt WAITT der Befehl DELAY wegen seines konstanten Zeitverhaltens verwendet werden.	
Befehlsgruppe	CON	
Querverweise	DELAY, WAITI, WAITAX	
Syntax-Beispiel	<pre>WAITT 5000 /* 5 Sekunden warten */ WAIT T 5000 /* Alternative Schreibweise */</pre>	
Programmbeispiel	WAIT_01.M	

__ Befehlsreferenz __

□ WHILE .. DO .. ENDWHILE

Kurzinfo	Bedingte Schleife mit Überprüfung des Abbruchkriteriums am Schleifenanfang (Während Bedingung erfüllt, wiederhole ...)
Syntax	WHILE (Bedingung) DO ENDWHILE
Parameter	Bedingung = Abbruchkriterium
Beschreibung	Mit der WHILE .. DO .. ENDWHILE Konstruktion kann man den eingeschlossenen Programmbereich in Abhängigkeit von einem beliebigen Kriterium ein- oder mehrfach wiederholen. Das Schleifenkriterium setzt sich aus einer oder mehreren Vergleichsoperationen zusammen und wird stets am Schleifenanfang überprüft. Das kann bei einem negativen Ergebnis bereits bei der ersten Überprüfung dazu führen, dass die Befehle innerhalb der Schleife nicht ausgeführt werden und das Programm sofort nach der ENDWHILE Anweisung fortgesetzt wird.
	ACHTUNG! In Abhängigkeit von dem Schleifenkriterium kann es vorkommen, dass der Schleifeninhalt nicht abgearbeitet wird. Um eine Endlosschleife zu vermeiden, müssen die innerhalb der Schleife abgearbeiteten Befehle direkt oder indirekt Einfluss auf das Ergebnis der Abbruchüberprüfung haben.
Befehlsgruppe	CON
Querverweise	LOOP, REPEAT .. UNTIL
Syntax-Beispiel	WHILE (A != 1 AND B == 0) DO Befehlszeile 1 Befehlszeile n ENDWHILE
Programmbeispiel	WHILE_01.M, INKEY_01.M

□ _GETVEL

Kurzinfo	Abtastzeit für AVEL und MAVEL verändern.
Syntax	var = _GETVEL t Anzeige der Werte in BE/s bei AVEL bzw. qc/s bei MAVEL.
Parameter	t = Abtastrate in Millisekunden
Beschreibung	Mit dem _GETVEL Befehl können Sie die Abtastzeit für AVEL und MAVEL verändern. AVEL und MAVEL arbeiten standardgemäß mit einer Abtastzeit von 20 ms, dadurch ist die Auflösung besser. Allerdings liegt nur alle 20 ms ein neuer Wert vor. Der Befehl _GETVEL dauert genauso lange wie der zugewiesene Wert, z.B. dauert _GETVEL 200 ca. 200 ms.
Befehlsgruppe	SYS
Querverweise	AVEL, MAVEL
Syntax-Beispiel	var = _GETVEL 200 Damit wird die Messauflösung wesentlich besser, allerdings erhält man Veränderungen immer erst mit einer Verzögerung von 200 ms.

□ #INCLUDE

Kurzinfo	Einfügen des Inhalts einer Datei an der aktuellen Progammposition.
Beschreibung	Diese #INCLUDE Anweisung wurde durch die #include Anweisung der Preprozessor-Befehle ersetzt.
Portabilität	Die Syntax der #include Anweisung für APOSS IDE Versionen vor Version MCO 5.00 enthielten keine Anführungszeichen. Wenn ein „altes“ Programm mit einer aktuellen APOSS IDE geöffnet wird, werden die Anführungszeichen automatisch hinzugefügt.
Befehlsgruppe	Preprozessor
Programmbeispiel	INCL_01.M + INCSTA01.M + INCPOS01.M + INCIN01.M



Anhang



▣ Neues in der aktuellen Version ab MCO 5.00

Mit MCO 5.00 wurden zur APOSS IDE erhebliche neue Funktionalität hinzugefügt und viele der vorhandenen Funktionen wurden wesentlich erweitert. Die wichtigsten Änderungen sind:

▣ APOSS Tools

Oszilloskop Tool

Das neue *Oszilloskop Tool* ist eine grafische Anzeige mit leistungsstarken und umfangreichen Funktionen zum Optimieren der Steuerungsparameter und zum Austesten der Programme. Es erlaubt dem Anwender alle internen Steuerungsparameter und Variablen sowie den Status zu überwachen, während die Steuerung läuft.

Das Oszilloskop Tool ersetzt die bisherige *Testfahrt* vollständig; diese wurde daher entfernt.

Array-Editor Tool

Der neue *Array-Editor* ermöglicht dem Anwender einen komfortablen Überblick aller Parameter und Arrays der Steuerung und deren Update mittels einer Liste. Diese Liste kann kundenspezifisch eingerichtet werden, so dass sie für die Konfiguration von Kundenanwendungen benutzt werden kann.

Der Array-Editor kann entweder innerhalb von APOSS oder als Stand-alone Anwendung genutzt werden.

CAM Editor Erweiterungen

Der *CAM-Editor* wurde wie folgt erweitert:

- Die Default Dateierweiterung für Konfigurations-Dateien wurde von „.cnf“ zu „.zbc“ geändert. „.cnf“ Dateien werden jedoch weiterhin akzeptiert. Durch diese Änderung werden Konflikte mit der Microsoft-Dateierweiterung „.cnf“ vermieden.
Mit „.zbc“ kann APOSS ein Datei-Symbol für den Windows-Explorer erstellen, damit Konfigurationsdateien durch Doppelklick geöffnet werden können.
- Das CAM-Editor Fenster wurde so verändert, dass es sich wie ein „normales“ Fenster verhält. Zum Beispiel arbeiten Menüs und Symbolleiste wie im Standard und das CAM-Editor-Fenster „geht in den Hintergrund“ wenn andere APOSS-Fenster ausgewählt werden. Konfigurationsdateien werden nun mit dem Menü Datei bzw. den Symbolen geöffnet und gesichert.
- Die Farbe der Linien und Punkte wurde verändert, so dass der Typ leichter erkannt werden kann. Tangentiallinien und -Punkte werden Grün, Kurvenlinien und -punkte Rot dargestellt werden.

__ Anhang __

- Der neue *Kurventyp* (GRAD - 3) ermöglicht die Spezifikation des Start- und Ende-Gradienten einer Kurve.
- Die zwei neuen Segmenttypen Trapez und 3. Ordnung können für benachbarte Tangenten-Segmente benutzt werden.

Überwachungsfenster

Das *Überwachungsfenster* wurde wie folgt geändert und erweitert:

- Das Überwachungsfenster ist kein separates Fenster („Überwachung anzeigen“) mehr, sondern in das APOSS-Fenster eingebunden und ständig zu sehen. Es wird wie üblich mit den Menü- und Symbol-Funktionen gesteuert.
- Überwachte Variablen werden in der Anwender-Programmdatei gepflegt und werden wieder zum Überwachungsfenster hinzugefügt, sobald ein Programm geöffnet wird.
- „Double“ Variablen und 2-dimensionale Arrays werden nun unterstützt.
- Verschiedene Beschränkungen des bisherigen Überwachungsfenster (z.B. die Begrenzung auf 10 Variablen) wurden entfernt.

□ **APOSS Compiler Erweiterungen**

Der APOSS-Compiler wurde um die folgenden Punkte weiter entwickelt:

- Der kompilierte Code wird für spezifische Steuerungen (und deren Prozessoren) optimiert. Dies führt zu kürzeren Befehlsausführungszeiten und einer schnelleren Programmabarbeitung.
- Die Stack-Größe kann im Menü *Einstellungen* → *Compiler* konfiguriert werden.
- Die Konstrukte SWITCH, CASE und BREAK werden unterstützt.
- Zweidimensionale Arrays werden unterstützt.
- Das Kopieren von Arrays wird unterstützt.
- Variablen können als „double“ oder „constant“ deklariert werden.
- Der Post-Inkrement (++) und Post-Dekrement (--) Operator werden unterstützt.
- Die Deklaration und der Aufruf von Funktionen werden unterstützt.
- Funktionen besitzen eine Parameterliste und liefern einen Rückgabewert.
- Lokale Variablen werden unterstützt.
- Casting (long <-> double) von Variablentypen wird unterstützt.
- Erweiterung der Befehle „ROUND“ und „FABS“ für double-Datentypen.
- Neue Floating-Point Befehle für double-Datentypen:
sqrt(), sin(), grad(), rad(), ln(), exp, pi

□ **Erweiterungen in der APOSS Benutzeroberfläche**

Die wichtigsten Änderungen in der APOSS-Benutzeroberfläche sind:

- Die Tools CAM Editor, Oszilloskop, etc. sind im neuen Menü Tools zusammen gefasst und können auch via Symbole geöffnet werden.
- Die Beispielprogramme können nun direkt über das Menü *Datei* → *Beispiel* geöffnet werden.
- Mit *Bearbeiten* → *Suchen in Dateien* kann man einen String in einer Datei auf der Festplatte suchen.
- *Nächstes Lesezeichen*, *Vorheriges Lesezeichen* und *Lesezeichen ein/aus* im Menü *Datei* bieten eine leichtere Benutzung der Funktion im Editor.
- Einzelne Programme können nun in der Steuerung gelöscht werden (*Steuerung* → *Programme*).
- „Position anfahren“ wurde aus der *Befehlsliste [F12]* entfernt.

□ Neue Befehle

Die folgenden neuen Befehle werden von der APOSS Programmiersprache unterstützt:

Befehl	Kurzinfo
APOSDIFF	Overflow-Handling von Inkrementalgebern in Anwendungen.
CANDEL	Löscht alle oder einzelne CAN-Objekte
CANIN	Liest ein Objekt über den CAN-Bus
CANINI	Initialisiert die notwendigen Objekte (PDOs) für den Datenaustausch mit CANopen-Teilnehmern oder aktiviert die erweiterte CANINI, CANIN Funktion.
CANOUT	Nachricht mit interner Nummer abschicken
CPOSDIFF	Overflow-Handling von Inkrementalgebern in Anwendungen.
DEFCANIN	Definiert ein Empfangsobjekt
DEFCANOUT	Definiert ein Sendeobjekt im CAN-Controller
INGLB	Liest eine globale CAN-Nachricht über CAN-Bus
INMSG	CAN-Nachricht aus dem Puffer lesen
IPOSDIFF	Overflow-Handling von Inkrementalgebern in Anwendungen.
JERKFINVAL	Berechnet die Endgeschwindigkeit für einen ruckbegrenzten Stopp mit maximaler Beschleunigung/Verzögerung.
JERKSTOPDIST	Berechnet den notwendigen Abstand für einen ruckbegrenzten Stopp mit maximaler Verzögerung.
LINKPDO	Systemvariable mit RxPDO verknüpfen und in die internen Parameter kopieren oder Teil eines Arrays in das PDO verlinken.
LINKSDO	TxPDO mit interner Systemvariable verknüpfen oder Teil eines Arrays aus dem PDO verlinken.
MAPOSDIFF	Overflow-Handling von Inkrementalgebern in Anwendungen.
MIPOSDIFF	Overflow-Handling von Inkrementalgebern in Anwendungen.
MSGVAL	Enthält den zweiten Teil der zuletzt gelesenen CAN-Nachricht.
ON CANINPUT	Unterprogramm aufrufen, wenn ein CAN-Telegramm vom Typ 'id' ankommt.
ON CANMSG GOSUB	Aufruf eines Unterprogramms bei einer Nachricht im Puffer.
ON DELETE..SETOUT	Löscht einen Positions-Interrupt.
ON KEYPRESSED GOSUB	Unterprogramm aufrufen, wenn eine LCP Taste gedrückt oder losgelassen wird.
ON posint .. GOSUB	Unterprogramm aufrufen, wenn ein Positions-Interrupt auftritt.
ON posint .. SETOUT (TOIN)	Simuliert ein Nockenschaltwerk (CAM-Box) (alle POSINT Typen).
OUTMSG	Sendet eine CAN-Nachricht.
PDO	Pseudo-Array für den direkten Zugriff auf die CANopen-PDO.
SDOREAD	Liest SDO eines angeschlossenen CANopen-Gerätes.
SDOREADSEG	Segmentiertes Lesen von SDOs (ungepackt).
SDOREADSEGP	Segmentiertes Lesen von SDOs (gepackt).
SDOSTATE	Ergebnis einer aktiven Kommunikation prüfen.
SDOWRITE	Setzt SDO eines angeschlossenen CANopen-Gerätes.
SYNCMARKERSTART	Setzt einen Marker oder die Markerbehandlung zurück.

__ Anhang __

Befehl	Kurzinfo
SWAPMENC	Dieser Befehl wird nicht mehr unterstützt. Die Funktion wurde mit SET ENCODERTYPE und SET MENCODERTYPE realisiert; siehe Par. 32-00 und 32-30.
TESTSETDEST	Definiert den Speicherbereich in dem eine Datenaufzeichnung abgelegt wird.
TESTSETINDEX	Systemvariablen für die Datenaufzeichnung festlegen.
TESTSETTIME	Konfiguriert die Abtastperiode für die Datenaufzeichnung.
TESTSETTYPE	Definiert, ob eine einmalige oder zyklische Datenaufzeichnung ausgeführt werden soll.

□ Neue und erweiterte Parameter

Folgende neuen Achsparameter werden unterstützt:

Parameter		Content
32-13 Enc.2 Steuerung	ENCCONTROL	Konfiguration, wie die Position nach einem Wechsel der Drehgeberquelle aufgeholt werden soll.
32-14 Enc.2 Teilnehmer ID		Feedback CAN Encoder Node ID.
32-15 Enc.2 CAN Überwachung		Feedback CAN Encoder Überwachung.
32-43 Enc.1 Steuerung	MENCCONTROL	Konfiguration, wie die Master-Position nach einem Wechsel der Drehgeberquelle aufgeholt werden soll.
32-44 Enc.1 Teilnehmer ID		Master CAN Encoder Node ID.
32-45 Enc.1 CAN Überwachung		Master CAN Encoder Überwachung.
32-73 Integralwert Filterzeit	KILIMTIME	Zeitspanne, über welche die Integrationsgrenze des Positionsreglers auf den definierten KILIM-Wert erhöht bzw. auf Null reduziert wird.
32-74 Schleppfehler Filterzeit	POSERRTIME	Zeitfenster [ms] für das Auslösen eines Positionsfehlerstatus.
32-86 Beschl.rampe für Ruckbegrenzung	JERKMIN	Zeitspanne zum Erreichen der definierten Maximalbeschleunigung.
32-87 Beschl.dauer für Ruckbegrenzung	JERKMIN2	Zeitspanne zum Erreichen der konstanten Maximalgeschwindigkeit.
32-88 Bremsrampe für Ruckbegrenzung	JERKMIN3	Zeitspanne zum Erreichen der definierten Maximalverzögerung.
32-89 Bremsdauer für Ruckbegrenzung	JERKMIN4	Zeitspanne bis zum Stillstand.
33-32 Geschw.-Feedforward Anpassung	SYNCFVVEL	Geschwindigkeits-Feedforward [per mall von VCMD] für den Synchronisations-Modus.
33-33 Synchronisationsfehler-Fenster	SYNCVFLIMIT	Synchronisationsfehler-Fenster [qc] für die automatische Deaktivierung von SYNCVFTIME.
33-90 X62 MCO CAN Teilnehmer ID	CANNR	CAN node ID
33-91 X62 MCO CAN Baudrate	CANBAUD	CAN Baudrate
33-94 X60 MCO RS485 Serieller Abschluss		RSTERMINATION
33-95 X60 MCO RS485 Serielle Baudrate		RSBAUDRATE

Im Zuge der Einführung der neuen Parameter wurden die folgenden Parameter überarbeitet:

32-01 *Inkrementalgeber Auflösung* ENCODER, 32-60 *Proportionalfaktor* KPROP, 32-61 *Differentialwert für PID-Regelung* KDER, 32-62 *Integralfaktor* KINT, 32-69 *Abtastzeit für PID-Regelung* TIMER und 32-63 *Grenzwert für die Integralsumme* KILIM.

□ Technische Referenz

Dieser Abschnitt dokumentiert Datenstrukturen und Compilerdetails, die der Anwender nur in Ausnahmefällen benötigt. Zum Beispiel wenn eine automatisch erzeugte Programmierung, wie ein CAM-Profil verändert werden soll.

Da der Abschnitt für erfahrene Programmier vorgesehen ist, wird die Referenz nur in Englisch dokumentiert.

□ Array Structure of CAM Profiles

Header

The header contains general information like

- Identification for curve array
- Version number for curve structure
- Type of curve
- Name of curve
- Index to curve information section
- Index to start/stop point section
- Index to fixed point section
- Index to interpolation point section
- Index to start/stop point indices (in interpolation section)
- Index to start/stop velocities (times 100000)
- Index to start path interpolation points
- Index to stop path interpolation points

Curve Information Section

This section of the array contains all information about the type of curve like

- Length of curve (master)
- Length of curve (slave)
- Number of fix points
- Number of Interpolation points (this gives the resolution)
- Type of interpolation
- Slave stop point, point where slave is positioned, when Synchronisation is stopped
- Correction start point (only valid for marker synchronization)
- Correction end point (only valid for marker synchronization)
- Maximum correction which is allowed (only valid for marker synchronization)
- Maximum start/stop path length (Size of start/stop path area)(min. 2)
- Number of start/stop point pairs
- Maximum number of cycles per minute (Application information)

Curve Start/Stop Point Section

This section contains the start/stop points. Because the use of this point is up to the user, we just speak of a path, which can be a start or a stop sequence. Every path consists of 2 points. If we are moving forward, the path starts (start or stop) with the a-point and ends with the b-point. If we are moving backward, the path starts with the b-point and ends with the a-point. So the user is able to tell us in the program, which pair of points to use for starting or stopping, when he uses a STARTCURVE or STOPCURVE command.

- Path 1 (a – point)
- Path 1 (b – point)
- Path 2 (a – point)
- - Path 2 (b – point), ...

__ Anhang __

These points have to lie on interpolation points, so possibly the PC software has to adjust them according to the interpolation resolution. This should not be a real restriction, because the interpolation points are normally very dense. So for example if we have rotating master which makes one revolution per cycle and we choose a cycle length of 3600 MU (1 MU = 1/10 degree). Let us further assume, that we choose the number of interpolation points as 1200, than you have a resolution of 3 MU = 3/10 degree for defining your start and stop points.

Fixed Point Section

This section contains the fix points, which were the basis for the interpolation calculation. These points always consist of the following triple

- Master coordinate
- Slave coordinate
- Type of point (tangent, curve)

These points are defined by the user in MU units (see internal description). If you want to avoid, that the real interpolation curve misses your fix points, you have to choose them in such a manner that they lay on an interpolation point (see above). This can be forced through a snap function within the PC software.

Interpolation Point Section

This section contains a list of slave coordinates. They belong to master coordinates which are of equal distance, given by the interpolation resolution.

Indices of Start/Stop Points

Here we have the indices of the start/stop points (see above) within the interpolation array. These are necessary for the ease of start and stop recognition. We are waiting until start index for example equals the actual index and direction of movement is correct. If both are true synchronization will be started. The same is true for stopping.

Start Stop Velocities

To be able to calculate an appropriate starting or stopping path, we need the velocity we have to reach at end (start) or we will have at the beginning (stop) in UU/MU units (Slave units per Master units).

Start / Stop Paths

This is the place for the interpolation points of the actual start and stop path. These points are calculated when a SYNCCSTART or SYNCCSTOP command is executed, but we have to reserve the room right now.

□ CAM Array Definition

Index	Name	Unit	Value	Description
General				
1	Identification	(dec)	999.000.001	Number to identify array
2	VersioNumber	(dec)	100	Version as decimal (1.00 = 100)
3	CurveType	(dec)	0	0 = symmetrical; 1 = compatible
4	CurveName 1	(4char)	Nona	Name of curve total 16 char.
5	CurveName 2	(4char)	meCu	default is:
6	CurveName 3	(4char)	rve0	NonameCurve00001
7	CurveName 4	(4char)	0001	
8	IndexCIF	(dec)	16	Index to Curve Information Part
9	IndexSTP	(dec)	27	Index to Start/Stop point Part
10	IndexFIP	(dec)	IndexSTP + STPno*2	Index to Fix point Part

__ Anhang __

Index	Name	Unit	Value	Description
11	IndexINP	(dec)	IndexFIP + FixPointNo * 3	Index to Interpolation Point Part
12	IndexSTPInd	(dec)	IndexINP + InterpolPointNo	Index to StartStop Interpolation Indices
13	IndexSTPVel	(dec)	IndexSTPInd +STPno*2	Index to StartStop Velocities
14	IndexSTIP	(dec)	IndexSTPVel +STPno*2	Index to Startpath interpolation points
15	IndexSTPIP	(dec)	IndexSTIP + MaxStartStopLen	Index to Stoppath interpolation points
Curve Information				
1	MasterCycleLen	MU	-	Length of Curve in CurveMaster units
2	SlaveCycleLen	UU	-	Slave max. travel distance in CurveSlave units
3	FixPointNo	(dec)	4	Number of fix points (minimum 4)
4	InterpolPointNo	(dec)	-	Number of interpolation points (including first and last, which correspond to the same location)
5	InterpolType	(dec)	0	0 = cubic spline, 1 = periodic cubic spline
6	SlaveStopPosition	UU	0	Position, where slave stands after stopping
7	CorrectionStartPoint	MU	0	Position, where Correction may start
8	CorrectionStopPoint	MU	MasterCycleLen	Position, where Correction has to be finished
9	MaximumCorrection	UU	-	Maximum Correction which is allowed in one cycle
10	MaxStartStopLen	(dec)	0	Maximum length of start/stop path (number of int. points)
11	StartStopNo	(dec)	0	Number of start stop point pairs (n) (see below)
12	MMaxCycles	(dec)	0	Max. number of cycles per minute (application info)
13	MMarkerPos	CM	0	Master Marker Position in curve
14	SMarkerPos	CS	0	Slave Marker Position in curve
Start/Stop Point				
1	STPoint_1.a	MU	0	Start (forward) / Stop (backward) point no. 1
2	STPoint_1.b	MU	0	Stop (forward) / Start (backward) point no. 1
3	STPoint_2.a	MU	0	Start (forward) / Stop (backward) point no. 2
4	STPoint_2.b	MU	0	Stop (forward) / Start (backward) point no. 2
5	...	MU	0	
6	...	MU	0	
2*n-1	STPoint_n.a	MU	0	Start (forward) / Stop (backward) point no. n
2*n	STPoint_n.b	MU	0	Stop (forward) / Start (backward) point no. n
Fix Point				
1	FixPoint_1.master	MU	0	Fix point no. 1 - master coordinate

__ Anhang __

Index	Name	Unit	Value	Description
2	FixPoint_1.slave	UU	-	Fix point no. 1 - slave coordinate
3	FixPoint_1.type	(dec)	C	Fix point no. 1 - type of point (C = Curve Point, T = Tangent Point)
4	...			
5	...			
6	...			
3*n-2	FixPoint_n.master	MU	MasterCycleLen	Fix point no. n - master coordinate
3*n-1	FixPoint_n.slave	UU	-	Fix point no. n - slave coordinate
3*n	FixPoint_n.type	(dec)	C	Fix point no. n - type of point (C = Curve Point, T = Tangent Point)
Interpolation Point				
1	IntPoint_1	UU	0	Interpolation Point no. 1 - slave coordinate
...				
n	IntPoint_n	UU	-	Interpolation Point no. n - slave coordinate
StartStop Indices				
1	STPoint_1.a-index	(dec)	0	Index in Interpolation Array, corresponding to Start point
2	STPoint_1.b-index	(dec)	0	Index in Interpolation Array, corresponding to Start point
3	..			
StartStop Velocities				
1	STPoint_1.a-veloc.	(dec)	(*100000)	Velocity (UU/MU * 100000) in Start point
2	STPoint_1.b-veloc.	(dec)	(*100000)	Velocity (UU/MU * 100000) in Start point
...				
StartPath Interpolation Points				
1	StartPoint_1	UU	0	Interpolation Point no. 1 - for start path
...				
n				
StopPath Interpolation Points				
1	StopPoint_1	UU	0	Interpolation Point no. 1 - for stop path
...				
n				

□ Curve Arrays and Curve Types

Starting with MCO 5.00 no interpolation points are used anymore. So only the fix points are relevant for the curve. When a SETCURVE is executed (or when the curve is really started), the coefficients for the corresponding polynomials are calculated. Then, when the curve is running, the polynomials are calculated on the fly, while driving.

This procedure allows the user to modify a curve on the fly within the application program. This can be done by overwriting some of the values within the curve array. (See description of curve array in Illustrations.). After that a SETCURVE must be executed to activate the modified curve. This curve is then started as soon as the active curve ends, or immediately by replacing the active curve, if the new curve is of type INTRPT_GRAD (see array description).



NB!:

The curve array is used by the internal SYNCC procedures as long as the curve is running. So you should never modify the curve array of a running curve. To solve this problem you must have two arrays which are used alternatively. That means while one curve is active, the next one can be prepared and started. As soon as the new one is active (PFG_CWRAP changes), the old curve can be modified again.

Type of Fix points

Starting with MCO 5.00 there are new types of fix points. The way tangent points are used is changed. Now the first fix point always tells what type of segment is following. So the last point is always of same type as the first point.

There are new points like Poly3 and Trapezoidal which allow special segments within the curve to be used.

For new curves only the bold point types should be used.

```
#define CU_CPOINT      1      // Curve point (next segment is 3'rd or 5'th order polynomial).
#define CU_T1POINT     2      // Tangent start point (replaced by CU_TPOINT).
#define CU_T2POINT     3      // Tangent end point (replaced by CU_CPOINT).
#define CU_TPOINT      4      // Next segment is a tangent segment.
#define CU_ZPOINT      5      // Next segment is a trapezoidal segment.
#define CU_3POINT      6      // Next segment is a 3'rd order polynomial.
```

So you can, for example, create sequences like .. 4 – 5 – 4 – 5 which means that you will have two straight lines (tangents) which are connected by two parabolas. Those two parabolas meet each other in the middle between the fix points and at all three points (start, middle, end) of the segments, the velocity is the same as the adjacent segment.

The following new curve types can be used by changing the array.

New Curve type 3 – CU_GRAD

Starting with MCO 5.00 another type of curve (3) is supported. This curve consists of only 2 fix points and is calculated as a polynomial of 5th order. Therefore, the following values were added at the end of the fix point area in the curve array (G_CFPIdx is the index of the fix point area):

```
RestartCurve[3] =          3          // Curve type
RestartCurve[G_CFPIdx+0] =    0          // Master start coordinate
RestartCurve[G_CFPIdx+1] =    0          // Slave start coordinate
RestartCurve[G_CFPIdx+2] =    1          // Fix Point Type = curvepoint
RestartCurve[G_CFPIdx+3] = G_ShingleDistance * 4 // Master end coordinate
RestartCurve[G_CFPIdx+4] = G_ShingleDistance * 2 // Slave end coordinate
RestartCurve[G_CFPIdx+5] =    1          // Fix Point Type = curvepoint
RestartCurve[G_CFPIdx+6] =    0          // Velocity v0
RestartCurve[G_CFPIdx+7] =    1          // Divisor v0
RestartCurve[G_CFPIdx+8] =    0          // Acceleration a0
RestartCurve[G_CFPIdx+9] =    1          // Divisor a0
```

__ Anhang __

```

RestartCurve[G_CFPIdx+10] = 0 // Jerk j0
RestartCurve[G_CFPIdx+11] = 1 // Divisor j0
RestartCurve[G_CFPIdx+12] = 1 // Velocity v1
RestartCurve[G_CFPIdx+13] = 1 // Divisor v1
RestartCurve[G_CFPIdx+14] = 0 // Acceleration a1
RestartCurve[G_CFPIdx+15] = 1 // Divisor a1
RestartCurve[G_CFPIdx+16] = 0 // Jerk j1
RestartCurve[G_CFPIdx+17] = 1 // Divisor j1

```

That means we have the possibility to define start and end gradients and acceleration for the polynomial. (Jerk is ignored at the moment. Designed for future use.)

In this array, the start coordinates are only for display purposes because they are replaced by the actual values when the curve is started. (see below)

As a result of this behavior of type 3 curves (predefined end values and calculated start values), they normally cannot be continued when they reach the end (since typically the velocities do not match). Therefore, they are normally continued by another standard curve. If for any reason they are not continued by another curve, they will just try to continue with the actual velocity. This is done with a poly5 looking more or less like a straight line.



NB!:

This continuation overwrites the original curve array with this continuation curve.

Continuation did not work for curves with more than 2 fix points prior to 6.7.11. In those versions, there was also an error when CU_GRAD curves with more than 2 fix points were used as a continuation curve.

New Curve type 4 CU_GRAD_INTRPT

Type 4 is available starting with MCO 5.00. This type is nearly identical to type 3 (CU_GRAD).

The big difference with curves of type CU_GRAD_INTRPT is that they are started immediately when the SETCURVE is executed. When this is done, the actual values for velocity and acceleration are used for the calculation. The actual values of the MCPOS and CURVEPOS are subtracted from the end coordinates of the curve before it is calculated (curves must always internally start at 0,0). This guarantees, that the original end coordinates are absolute to the start of the interrupted curve.

For example, assume that a curve is running which starts at 0,0 and ends at 2000,2000 (master,slave). Now we define a curve of type CU_GRAD_INTRPT, which starts anywhere and ends at 4000,4000. If this curve is now set by SETCURVE at the moment when the original curve passes 1500,1800, for example, then the new curve is calculated in such a manner that it starts at this point (1500,1800) and ends at 4000,4000. To realize this, it uses the velocity and acceleration in the actual point, sets MCPOS and CURVEPOS to 0 and reduces the end coordinates to (4000-1500, 4000-1800) = (2500,2200). It will have the defined velocity (v1) and acceleration (a1) defined in the curve array.

These types of curve are used for processes where the standard curve looks more or less like a straight line (SYNCP / SYNCM behavior) and where the poly5 curves are used to align start or stop or restart processes to defined points.



NB!:

The responsibility for the correctness of poly5 curve lies with the user / application. The firmware does not do any plausibility test.

To allow readability by CAM-Editor, the CurveVersion (index 2) should be 102. Otherwise, the CAM editor will not accept those new curves.

Curve type 3 – CU_GRAD with SYNCSTART

Such curves (Poly 5) can now also be started with SYNCSTART. This means the curve starts immediately and does not wait for next marker. (In previous versions, it was only possible to start such a curve with SYNCMSTART 2001.)

If such a curve is started now with SYNCSTART, the user is responsible for the correct setting of the end-points. Startpoint is 0 which will be internally set to the actual command position. Hence, the curve must be defined from 0 .. endpoint.

Example:

```
G_CFPIdx = StartCurve[10]
  // Array index, where fix point definition starts
  // 1. Fix Point must be 0/0,
  // because curve always must start at this position
StartCurve[G_CFPIdx+0] = 0    // Master start coordinate
StartCurve[G_CFPIdx+1] = 0    // Slave start coordinate
StartCurve[G_CFPIdx+2] = 1    // Fix Point Type = curvepoint
StartCurve[G_CFPIdx+3] = G_ShingleDistance * 4
  // Master end coordinate
StartCurve[G_CFPIdx+4] = G_ShingleDistance * 2
  // Slave end coordinate
StartCurve[G_CFPIdx+5] = 1    // Fix Point Type = curvepoint
StartCurve[G_CFPIdx+6] = 0    // Velocity v0
StartCurve[G_CFPIdx+7] = 1    // Divisor v0
StartCurve[G_CFPIdx+8] = 0    // Acceleration a0
StartCurve[G_CFPIdx+9] = 1    // Divisor a0
StartCurve[G_CFPIdx+10] = 0   // Jerk j0
StartCurve[G_CFPIdx+11] = 1   // Divisor j0
StartCurve[G_CFPIdx+12] = 1   // Velocity v1
StartCurve[G_CFPIdx+13] = 1   // Divisor v1
StartCurve[G_CFPIdx+14] = 0   // Acceleration a1
StartCurve[G_CFPIdx+15] = 1   // Divisor a1
StartCurve[G_CFPIdx+16] = 0   // Jerk j1
StartCurve[G_CFPIdx+17] = 1   // Divisor j1

SETCURVE StartCurve
...
SYNCC 0
SYNCCSTART 0
```

In such a case, when SYNCSTART is executed, the 0 point of the curve is mapped onto the actual command position. Also, the actual velocity is calculated (and start acceleration is set to zero).

__ Anhang __

Curve type 3 – CU_GRAD with SYNCCSTART and DEFSYNCORIGIN

In addition to the above mentioned possibility, you have the option to define the endpoints of the curve with absolute values. So the start of such a curve could look like

```
mendpos = MIPOS + mdistqc
  // apos must be in qc
sendpos = (sdistqc % G_SlaveQcProProdukt) * G_SlaveQcProProdukt + SYSVAR[4098]
defsyncorigin mendpos sendpos
  // define target position for master and slave (in qc)
SYNCC 0
SYNCCSTART 0
```

In this case, the curve is again started immediately and the actual command position will be the Curve zero position. However, the end position of master and slave are calculated in such a way that the curve will end at the given absolute positions.

Final velocity is given by the curve (1 in our example above) and start velocity is taken from actual values.

Curve types CU_GRAD with stability check

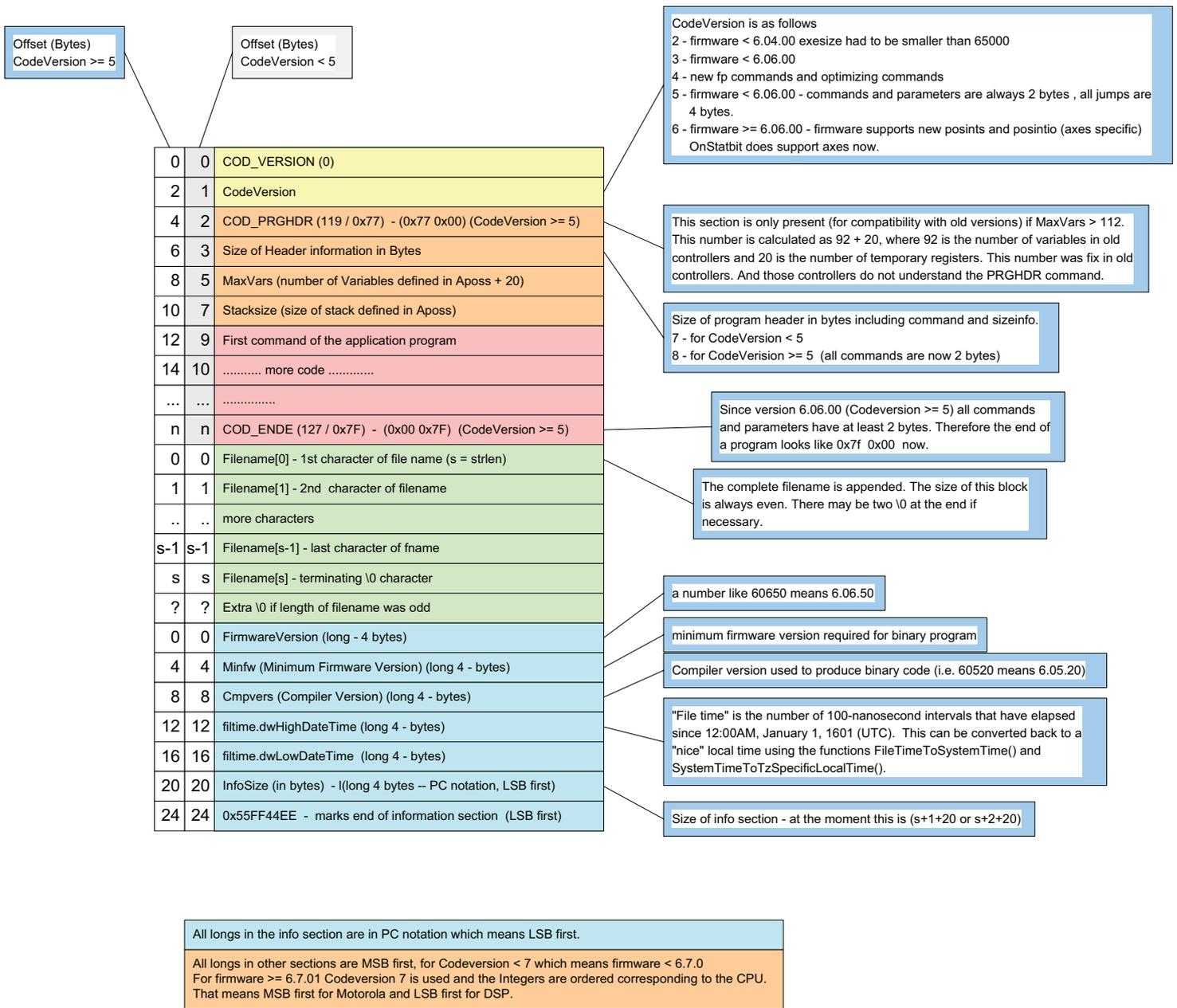
Whenever the new curve types are calculated, a check for extremes within a poly5 is also done if possible. (This only works if start acceleration is 0.) If an extreme within the interval is found, the curve error flag PG_FLAG_CURVE_ERR is set and the error number is stored in the PFG_G_LastError which allows the user to detect this situation. At the same time, the PFG_G_CPOLYMAXVEL [4288] and PFG_G_CPOLYMINVEL [4289] (SU/MU) are stored. Writing to PFG_G_LastError [4258] clears the flag PG_FLAG_CURVE_ERR (Bit 64 <<24) in the STAT.

So a sample could look like.

```
IF((STAT x(1))&(64<<24)) THEN // error bit set
  switch(SYSVAR[4258])
    case 5 : PRINT " Minimum in interval ",SYSVAR[4289] * 100 % 128
             break
    case 6 : PRINT " Maximum in interval ",SYSVAR[4288] * 100 % 128
             break
    case 7 : PRINT " Minimum in interval ",SYSVAR[4289] * 100 % 128
             PRINT " Maximum in interval ",SYSVAR[4288] * 100 % 128
             break
    default : PRINT " Other curve error ",SYSVAR[4258]
  endswitch
ENDIF
```

▣ Abbildungen

Illustrations: Bin File Map (Compiler >= 6.7.0)



BinFileMap_6_5.vsd
2009-08-31-bi

— Anhang —

CurveArray (Long orientiert)

1	1	Identification (999.000.001)
2	2	Version (101, 102)
3	3	CurveType Range is 0 .. 4
4	4	CurveName1 (4char)
5	5	CurveName2 (4char)
6	6	CurveName3 (4char)
7	7	CurveName4 (4char)
8	8	IndexCIF - CurveInformation - Default = 17
9	9	Index STP - Start-Stop Points Default = 31 (32 for Version 102)
10	10	Index FIP - Fixpoint Part Default STP+STPno*2
11	11	Index INP - Interpolation Part Default FIP + FixPointNo * 3
12	12	Index STPI - StartStopInerplnd Default INP + InterPolPointNo
13	13	Index STPV - StartStopVel Default STP + STPno * 2
14	14	Index STPI - StartPathInterpol Default STPV + STPno*2
15	15	Index STPIP - StopPathInterpol Default STIP + MaxStartStopLen
16	16	Extra Info Index (CU_GRAD / LBLINF - Optional Label Info)
17	1	MasterCycleLen (MU) Length of Curve in Master Units
18	2	SlaveCycleLen (UU) - Max. Slave travel dist in UU
19	3	FixPoint number Number of fix points (minimum 2)
20	4	InterpolPointNo Number of interpol points
21	5	Interpolation Type (0 = open, 1 = periodic, 2+3 spec)
22	6	SlaveStopPosition position slave has to be after stop
23	7	CorrectionStartPoint Pos. where correction may start
24	8	CorrectionStopPoint Pos. where correction has to stop
25	9	Maximum Correction maxmal allowed correction
26	10	MaxStartStopLen max length of start stop path
27	11	StartStopNo Number of start stop point pairs
28	12	MMaxCycles Max cycles per minute (info only)
29	13	MMarkerPos Master Marker Pos. in curve
30	14	SMarkerPos Slave Marker Pos in curve (cmd)
31	15	ExtraDataSize Size of extra Data (Version>=102)
32	1	STPoint_1a Start Point Pair 1 - Point A
		STPoint_1b Start Point Pair 1 - Point B
	
		FixPoint_1.master Master Coordinate
		FixPoint_1.slave Slave Coordinate
		FixPoint_1.type Type (C = curve, T = Tangent)
	
		FixPoint_n.master Master Coordinate
		FixPoint_n.slave Slave Coordinate
		FixPoint_n.type Type (C = curve, T = Tangent)
		StartVelocityNum used for Poly5 and Splines
		StartVelocityDen
		StartAccelerationNum (not used for Splines)
		StartAccelerationDen (not used for Splines)
		StartJerkNum (not used at the moment)
		StartJerkDen (not used at the moment)
		EndVelocityNum used for Poly5 and Splines
		EndVelocityDen
		EndAccelerationNum (not used for Splines)
		EndAccelerationDen (not used for Splines)
		EndJerkNum (not used at the moment)
		EndJerkDen (not used at the moment)
		Interpolatio nSection (Interpolation Points, Start Stop Indices, StartStop Velocities, StartPath Interpolation Points, StopPath Interpolation Points)

defines handling of start / end velocities.
 0 = start and end velocity is average,
 1 = end velocity is set to start velocity (Hauser compatible).
 2 = start and end gradients are set to 0 (5th order).
 3 = start and end gradients are user defined (CU_GRAD)
 4 = start and end gradients user defined (CU_GRAD_INTRPT) (starts immediately)

not used any more, because curves are calculated on the fly now

optional Label Info, only used if Interpolation Type (see 21) is >= 2
 Since Version 102 used if CurveType(see 3) is CU_GRAD (3) or CU_GRAD_INTRPT(4). In that case the extra info contains start and stop gradients.

internally overwritten by FixPoint_n.master - FixPoint_1.master

internally overwritten by (FixPoint_n.slave- FixPoint_1.slave)

maximum is 100 at the moment

not used any more, because curves are calculated on the fly now

Interpolation Types are interpreted as follows:
 0 = open (only relevant for CAM - editor)
 1 = periodic (only relevant for CAM - editor)
 2 = Labeling - old version with precalculation
 3 = Labeling - actual development version

wird im Moment nicht übergeben !! Stop ohne CURVESTOP wird wohl nicht funktionieren

next 3 only relevant for marker correction

not used any more, because curves are calculated on the fly now

next 2 only relevant for marker correction

At the moment length is 12

values are given in MU. Point A is where the stopping begins and B is where it has to be finished. If driving backward, it is vice versa.

Fixpoints are give in MasterUnits (master coordinate) or UserUnits (slave coordinate). The type can be either
 C = CurvePoint
 T = TangentPoint
 More then two curvepoints are interpolated by cubic spline functions. That means, that a 3rd order Polynom connects two curvepoints in that way, that the position, velocity and acceleration are identical in intermediate Curvepoints.
 If there is only one CurvePoint followed by an TangentPoint ore a TangentPoint followed by only one CurvePoint (endpoint) or two tangents following each other, then we use 5th order polynoms to connect these points.

Velocity is used, if the CurveType is 3 or 4 (CU_GRAD or CU_GRAD_INTRPT) and curve does not start with a tangent. If curve starts with a tangent, all other values are ignored too.

Acceleration is used, if the CurveType is 3 or 4 (CU_GRAD or CU_GRAD_INTRPT) and curve starts with a Poly5.

Velocity is used, if the CurveType is 3 or 4 (CU_GRAD or CU_GRAD_INTRPT) and curve does not end with a tangent. If curve ends with a tangent, all other values are ignored too.

Acceleration is used, if the CurveType is 3 or 4 (CU_GRAD or CU_GRAD_INTRPT) and curve ends with a Poly5.

not used any more, because curves are calculated on the fly now

□ Stichwortverzeichnis

#

#INCLUDE 148

—

_GETVEL 147

A

Abbildungen 162

ACC 10

APOS 11

APOSDIFF 12

APOSS Benutzeroberfläche Erweiterungen 150

APOSS Compiler Erweiterungen 150

APOSS Tools 149

Array Structure of CAM Profiles 154

AVEL 12

AXEND 13

B

Befehlsreferenz lesen 3

Benutzereinheiten [BE] 7

C

CAM Array Definition 155

CANDEL 14

CANIN 14

CANINI 16

CANOUT 17

COMOPTGET 18

COMOPTSEND 18

CONTINUE 19

CPOS 19

CPOSDIFF 20

CSTART 20

CSTOP 21

Curve Arrays 158

Curve Types 158

CURVEPOS 22

CVEL 23

D

DEC 24

DEFCANIN 25

DEFCANOUT 26

DEFCORIGIN 26

DEFMCPPOS 27

DEFMORIGIN 27

DEFORIGIN 28

DEFSYNCORIGIN 28

DELAY 31

DELETE ARRAYS 31

DIM 32

DISABLE .. interrupts 33

E

ENABLE .. interrupts 35

ENCPOSOFFS 36

ENCTGREAD 37

ENCTGWRITE 38

ERRCLR 38

ERRNO 39

EXIT 39

G

GET 40

GETVLT 40

GETVLTSUB 41

GOSUB 41

GOTO 42

H

HOME 43

I

IF .. THEN .., ELSEIF .. THEN .. ELSE .. ENDIF 44

IN 45

INAD 46

INB 47

INDEX 48

INGLB 48

INKEY 49

INMSG 50

IPOS 51

IPOSDIFF 52

J

JERKFVEL 53

JERKSTOPDIST 53

L

LINKGP 54

__ Anhang __

LINKPDO	55	PID	94
LINKSDO	57	POSA	95
LINKSYSVAR	59	POSA CURVEPOS	96
Literatur	4	POSR	96
LOOP	59	PRINT	97
		PRINT DEV	97
		PULSACC	98
		PULSVEL	99
M		Q	
MAPOS	60	Quadcounts	6
MAPOSDIFF	60		
Master Units [MU]	7	R	
MAVEL	61	REPEAT .. UNTIL	99
MENCPOSOFFS	62	RSTORIGIN	100
MENCTGREAD	62		
MENCTGWRITE	63	S	
MIPOS	64	SAVE part	100
MIPOSDIFF	65	SAVEPROM	101
MLONG	6	SDOREAD	101
MOTOR OFF	66	SDOREADSEG	102
MOTOR ON	66	SDOREADSEGP	103
MOTOR STOP	67	SDOSTATE	104
MOVESYNCORIGIN	67	SDOWRITE	105
MSGVAL	68	SET	105
		SETCURVE	106
		SETMORIGIN	107
		SETORIGIN	107
N		SETVLT	108
Neue Befehle	151	SETVLTSUB	108
Neue und erweiterte Parameter	153	STAT	109
NOWAIT	69	SUBMAINPROG .. ENDPROG	110
		SUBPROG name .. RETURN	111
		Symbole	5
O		SYNCC	112
ON CANINPUT	70	SYNCCMM	114
ON CANMSG GOSUB	70	SYNCCMS	115
ON COMBIT .. GOSUB	71	SYNCCSTART	116
ON DELETE .. GOSUB	71	SYNCCSTOP	117
ON DELETE .. SETOUT	73	SYNCERR	118
ON ERROR GOSUB	74	SYNCM	119
ON INT .. GOSUB	75	SYNCMARKERSTART	121
ON PARAM .. GOSUB	77	SYNCP	122
ON PERIOD	79	SYNCSTAT	123
ON posint .. GOSUB	80	SYNCSTATCLR	124
ON posint .. SETOUT (TOIN)	83	SYNCV	125
ON STATBIT .. GOSUB	84	Systemprozessdaten	126
ON TIME	85	SYSVAR	126
OUT	86		
OUTAN	87	T	
OUTB	88	Technische Referenz	154
OUTDA	89		
OUTMSG	89		
P			
PCD	90		
PDO	91		

__ Anhang __

TESTSETDEST	127
TESTSETINDEX	129
TESTSETP	130
TESTSETTIME	133
TESTSETTYPE	134
TESTSTART	136
TESTSTOP	138
TIME	140
TRACKERR	141

U

USRSTAT	142
---------------	-----

V

VEL	142
VLTALARMSTAT	143
VLTCONTROL	143
VLERRCLR	144

W

WAITAX	144
WAITI	145
WAITNDX	145
WAITP	146
WAITT	146
WHILE .. DO .. ENDWHILE	147