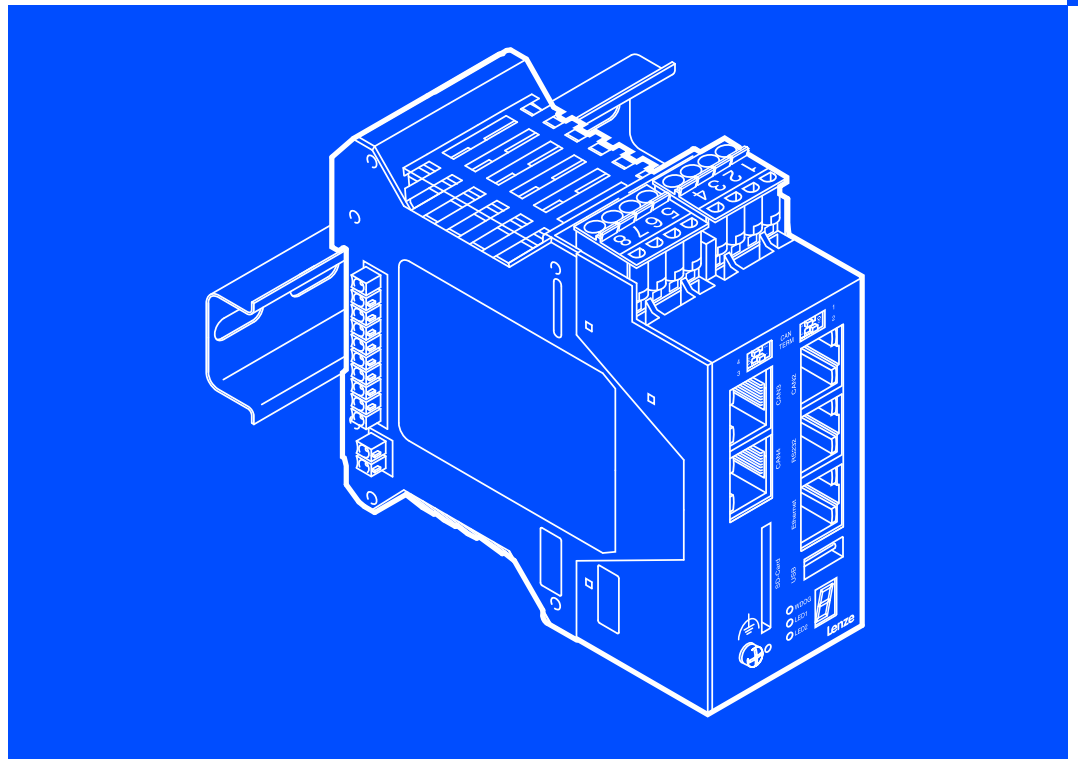


ETC Motion Control



ETCHMxxxBCB

Betriebssystem MC für 2. Gerätegeneration



Tipp!

Aktuelle Dokumentationen und Software-Updates zu Lenze Produkten finden Sie im Internet jeweils im Bereich "Services & Downloads" unter **<http://www.Lenze.com>**

1	Vorwort und Allgemeines	8
1.1	Über dieses Handbuch	8
1.2	Für welche Produkte ist das Handbuch gültig?	9
2	Erste Schritte	10
2.1	Inbetriebnahmeschritte (Übersicht)	11
2.2	Kommunikation zwischen PC und ETCHx herstellen	12
2.2.1	ETCHx und PC verbinden	12
2.2.2	ETCHx starten	12
2.2.3	Terminalprogramm "HyperTerminal" starten und Monitorschnittstelle aktivieren	13
2.2.4	IP-Adresse der ETCHx zuweisen	17
2.3	Antriebe über Maschinenkonstanten parametrieren	19
2.3.1	Übersicht der wichtigsten Maschinenkonstanten	19
2.3.2	Maschinenkonstanten-Datei ETCxM.mk	22
2.4	ETC-SPS-Programmierung mit CoDeSys	23
2.4.1	CoDeSys installieren	23
2.4.2	Steuerung im ETC-CoDeSys konfigurieren	23
2.5	SPS-Beispielprogramm erstellen	31
2.5.1	SPS-Beispielprogramm starten und konfigurieren	31
2.5.2	SPS-Beispielprogramm in die Steuerung laden	34
2.6	SPS-Programm testen	38
2.6.1	Visualisierungsfenster/Hauptfenster	38
2.6.2	Visualisierungsobjekt Statusanzeige	39
2.7	Firmware aktualisieren	40
2.7.1	Aufruf des Bootmonitors in der Steuerung	40
2.7.2	Versionsabfrage der Firmware	41
2.7.3	Updaten der Firmware	41

3	Grundlagen der Motion-Steuerung	43
3.1	Interpolator - Kern des Motion-Systems	43
3.2	Maschinenkonstanten - Steuerung konfigurieren	46
3.2.1	Maschinenkonstanten-Übersicht	49
3.3	Funktionsbibliothek MC.lib - Bewegungsaufträge definieren	61
3.3.1	Funktionsbaustein-Übersicht	62
3.3.2	Allgemeine Schnittstellen der Funktionsbausteine	64
3.3.3	Datentyp AXIS_REF	67
3.3.4	Bewegungen mehrerer Achsen synchronisieren	67
3.3.5	Schnelle Ein-/Ausgänge für Interpolator	71
3.3.6	Software-Endschalter	71
3.3.7	Quickstop (Schnellhalt)	72
3.3.8	Technische Einheiten	72
3.3.9	Virtuelle Achse	72
3.3.10	Rotationsachse	72
3.4	Fehlerhandling	73
3.4.1	Fehlergruppen	73
3.4.2	Achsfehlermeldungen	74
3.4.3	Systemfehlermeldungen	75
3.4.4	Liste aller von Funktionsbausteinen gemeldeten Fehler	77
4	Funktionsbibliothek MC.lib	82
4.1	Funktionsbausteine für die Konfiguration und Initialisierung	82
4.1.1	Funktion MC_WriteMachineConstant	82
4.1.2	Funktion MC_DefineAxisReset	84
4.1.3	Funktion MC_DefineAxis	85
4.1.4	Funktionsbaustein MC_InitAxes	88
4.2	Motion-Funktionsbausteine nach PLCopen-Standard	89
4.2.1	Funktionsbaustein MC_MoveAbsolute	89
4.2.2	Funktionsbaustein MC_MoveRelative	92
4.2.3	Funktionsbaustein MC_MoveAdditive	95
4.2.4	Funktionsbaustein MC_MoveSuperImposed	98
4.2.5	Funktionsbaustein MC_MoveVelocity	101
4.2.6	Funktionsbaustein MC_Home	104
4.2.7	Funktionsbaustein MC_Stop	105
4.2.8	Funktionsbaustein MC_PositionProfile	108
4.2.9	Funktionsbaustein MC_VelocityProfile	112
4.2.10	Funktionsbaustein MC_AccelerationProfile	116
4.2.11	Funktionsbaustein MC_TouchProbe	120
4.2.12	Funktionsbaustein MC_Abort_Trigger	123
4.2.13	Funktionsbaustein MC_TorqueControl	125
4.2.14	Funktionsbaustein MC_DigitalCamSwitch	129

4.3	Administrative Funktionsbausteine nach PLCopen-Standard	132
4.3.1	Funktionsbaustein MC_Power	132
4.3.2	Funktionsbaustein MC_ReadStatus	133
4.3.3	Funktionsbaustein MC_ReadAxisError	135
4.3.4	Funktionsbaustein MC_Reset	137
4.3.5	Funktionsbaustein MC_ReadParameter	138
4.3.6	Funktionsbaustein MC_ReadBoolParameter	139
4.3.7	Funktionsbaustein MC_WriteParameter	140
4.3.8	Funktionsbaustein MC_WriteBoolParameter	141
4.3.9	Funktionsbaustein MC_SetOverride	142
4.3.10	Funktionsbaustein MC_SetPosition	145
4.3.11	Funktionsbaustein MC_ReadActualVelocity	148
4.3.12	Funktionsbaustein MC_ReadActualTorque	149
4.3.13	Funktionsbaustein MC_ReadActualPosition	151
4.4	Achskopplung Funktionsbausteine nach PLCopen-Standard	152
4.4.1	Funktionsbaustein MC_CamTableSelect	152
4.4.2	Funktionsbaustein MC_CamExtTableSelect	155
4.4.3	Funktionsbaustein MC_CamLargeTableSelect	159
4.4.4	Funktionsbaustein MC_CamTableDelete	162
4.4.5	Funktionsbaustein MC_CamIn	163
4.4.6	Funktionsbaustein MC_CamOut	167
4.4.7	Funktionsbaustein MC_CamReadSlavePosition	168
4.4.8	Funktionsbaustein MC_GearIn	169
4.4.9	Funktionsbaustein MC_GearInPos	172
4.4.10	Funktionsbaustein MC_AbortGearInPos	176
4.4.11	Funktionsbaustein MC_GearOut	177
4.4.12	Funktionsbaustein MC_Phasing	179
4.4.13	Funktionsbaustein MC_ResetPhaseShift	182
4.5	Funktionsbausteine für schnelle Ein-/Ausgänge	183
4.5.1	Funktionsbaustein MC_DefineQuickInputs	183
4.5.2	Funktionsbaustein MC_DefineQuickOutputs	185
4.5.3	Funktionsbaustein MC_ReadDigitalInput	187
4.5.4	Funktionsbaustein MC_ReadDigitalOutput	188
4.5.5	Funktionsbaustein MC_WriteDigitalOutput	189
4.6	Support-Funktionsbausteine	190
4.6.1	Funktionsbaustein MC_AttachTaskToInt	190
4.6.2	Funktionsbaustein MC_CalcTimeToMove	193
4.6.3	Funktionsbaustein MC_CalcVelocityToMove	195

4.7	Kundenspezifische Funktionsbausteine	197
4.7.1	Funktionsbaustein MC_TouchProbe_CAN_LH	197
4.7.2	Funktionsbaustein MC_Abort_Trigger_CAN_LH	200
4.8	Funktionsbausteine zum Fehlerhandling	202
4.8.1	Funktionsbaustein MC_ReadSysError	202
4.8.2	Funktionsbaustein MC_ResetSysError	204
4.8.3	Funktionsbaustein MC_ReadEvent	205
5	Parameterliste	206
5.1	Reservierte Parameter nach PLCopen-Standard	206
5.2	Anwenderspezifische Parameter	208
6	SPS-Programmierung	213
6.1	ETC-SPS-Programmierung mit CoDeSys	213
6.2	CoDeSys installieren	214
6.2.1	Systemvoraussetzungen für CoDeSys V2.xx	214
6.2.2	Software installieren	214
6.3	ETC und PC verbinden	215
6.3.1	V.24-Schnittstelle	215
6.3.2	Ethernet-Schnittstelle	215
6.4	Projektierung	216
6.4.1	Zielsystem einstellen	216
6.4.2	SPS-Tasks der ETCxM konfigurieren	216
6.4.3	E/A-Module konfigurieren	218
6.4.4	Adressierung	224
6.4.5	Remanente Variablen	225
6.4.6	Objektverzeichnis (Parametermanager)	225
6.5	Netzwerkvariablen	228
6.5.1	Einstellungen in den Zielsystemeinstellungen	228
6.5.2	Einstellungen in den globalen Variablenlisten	229
6.6	Programm generieren	231
6.7	Schnittstelle zur ETC	232
6.7.1	Datenbausteine	232
6.7.2	Systemvariablen der ETCxM	232
6.7.3	Maschinenkonstanten in der ETCxM verwenden	232
6.7.4	Fehlermeldungen bei der ETCxM auslesen	232

6.8	Bibliothek	233
6.8.1	Allgemeine Funktionen	233
6.8.2	V24-Funktionen	244
6.8.3	FILE IO-Funktionen	249
6.8.4	Speicherzugriffsfunktionen	261
6.8.5	CANopen Funktionen	266
6.8.6	CAN-Funktionen (nur ETCxM)	276
6.8.7	MMI-Kommunikationsfunktionen	281
6.8.8	Echtzeituhr (nur ETCxM)	284
6.9	Bibliothek ServerSDO.lib	285
6.9.1	InitServerSdo	285
7	Stichwortverzeichnis	286

1 Vorwort und Allgemeines

1.1 Über dieses Handbuch

Zielgruppe Dieses Handbuch wendet sich an alle Personen, die das ETC Motion Control-System der 2. Gerätegeneration unter dem Betriebssystem "MC" programmieren und inbetriebnehmen.

Inhalt Im Softwarehandbuch "Betriebssystem MC" finden Sie Informationen zu folgenden Themen:

- ▶ Erste Schritte – eine chronologische Beschreibung der Inbetriebnahmeschritte
- ▶ Aufbau und Parametrierung von Funktionsbausteinen
- ▶ Fehlerhandling
- ▶ SPS-Programmierung

Weitere Information Im hardware-orientierten ETC-Gerätehandbuch für die 2. Gerätegeneration finden Sie Informationen zu folgenden Themen:

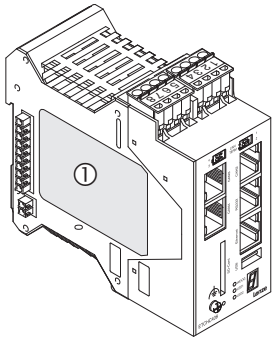
- ▶ Technische Daten
- ▶ Aufbau und Funktion der System-Komponenten einschließlich Schnittstellen-Beschreibung
- ▶ System-Komponenten montieren, anschließen und instandsetzen



Lesen Sie zuerst diese Anleitung und die Dokumentation zum Grundgerät, bevor Sie mit den Arbeiten beginnen!
Beachten Sie die enthaltenen Sicherheitshinweise.

1.2 Für welche Produkte ist das Handbuch gültig?

Typenschlüssel	①					
Steuerung	ETCH	C	x	xx	B	CB
Produkt						
Ausführung M = MC-Kern						
Anzahl CAN-Busse 2 = 2 CAN-Busse 4 = 4 CAN-Busse						
Anzahl Achsen z. B. 04 = 4 Achsen						
Generation B = 2. Gerätegeneration						



ETCH102

Dieses Kapitel vermittelt Grundlagen des ETC-Systems der 2. Generation und beschreibt die Vorgehensweise bei der Realisierung einer Automatisierungsaufgabe.

ETC

Die ETC der 2. Gerätegeneration verfügt über maximal 60 Achsen, wobei maximal 48 Achsen als physikalische CANopen-Achsen und maximal 12 Achsen als Achsen mit analoger Schnittstelle (local bus LBM-ARI) konfiguriert werden können.

Damit die ETC-Steuerung ihre zugeordnete Steuerungsaufgabe ausführen kann, benötigt sie verschiedene Programme, die vom IPC (oder Standard-PC) übertragen werden:

- ▶ Betriebssystem bzw. Firmware der Steuerung
- ▶ SPS-Programme

IPC

Über den IPC (oder Standard-PC) programmieren, bedienen und warten Sie die ETC-Steuerung. Hier laufen z. B. folgende Applikationen:

- ▶ Betriebssystem Windows 2000 oder XP
- ▶ Terminal-Programm (z. B. HyperTerminal) zum Einrichten der Ethernet-Verbindung zwischen IPC und ETC-Steuerung und für die Konfiguration der ETC-Steuerung über die Monitorschnittstelle (z. B. Update der Firmware).
- ▶ CoDeSys-Entwicklungsumgebung zum Erstellen und Testen von SPS-Programmen.

**Hinweis!**

Für die Inbetriebnahme benötigen Sie eine externe Tastatur am IPC. Während des normalen Betriebs ist sie nicht erforderlich.

2.1 Inbetriebnahmeschritte (Übersicht)



Stop!

Beachten Sie die Hinweise im Kapitel "Erstes Einschalten" des ETC-Gerätehandbuchs bevor Sie das System in Betrieb nehmen.

Schritt	Beschreibung	Siehe
1	ETCHM über RS232-Leitung mit PC verbinden und starten.	12
2	Am PC Terminalprogramm starten (ggf. vorher konfigurieren) und Monitor-schnittstelle der ETCHM aktivieren.	13
3	IP-Adresse der ETCHM zuweisen.	17
4	Antriebe über Maschinenkonstanten oder über die Datenstruktur im SPS-Programme parametrieren.	19
5	Ggf. CoDeSys installieren.	23
6	CoDeSys starten und konfigurieren.	23
7	SPS-Programm erstellen.	31
8	SPS-Programm in die ETCHM laden, starten und testen.	34

2.2 Kommunikation zwischen PC und ETCHx herstellen**2.2.1 ETCHx und PC verbinden**

Die Verbindung zwischen der ETCHx und einem PC ist auf 3 Arten möglich.

Verbindungsart	Leitungsausführung	Beschreibung
Serielle Verbindung	Siehe ETC-Gerätehandbuch	Diese Verbindung wird nur für die Inbetriebnahme benötigt! Ein freier COM-Port am PC wird mit der RS232-Schnittstelle der ETC verbunden. Der PC kommuniziert über ein Terminalprogramm mit der Monitorschnittstelle der ETC.
Lokale Verbindung über Ethernet (Peer-to-Peer-Verbindung)	Systemleitung Typ EWL 0065, EWL 0066 oder EWL 0067 (beidseitig RJ45)	Eine Netzwerkkarte im PC wird mit dem Ethernet-Anschluss der ETC über ein Cross-over-Kabel verbunden. Kommuniziert wird über das TCP/IP-Protokoll.
Netzwerk-Verbindung	Ethernet Patch-Kabel STP Cat5 (beidseitig RJ45)	Die ETC wird über ihren Ethernet-Anschluss an ein separates oder bestehendes Netzwerk (z. B. Intranet) angeschlossen. Kommuniziert wird über das TCP/IP-Protokoll.

2.2.2 ETCHx starten

1. Verbinden Sie die seriellen Schnittstellen von PC und ETC.
2. Starten Sie den PC.
3. Starten Sie die ETC durch Netzeinschalten.
Sobald die ETC-Firmware geladen ist, leuchtet an der ETC die grüne Watchdog-LED.

2.2.3 Terminalprogramm "HyperTerminal" starten und Monitorschnittstelle aktivieren

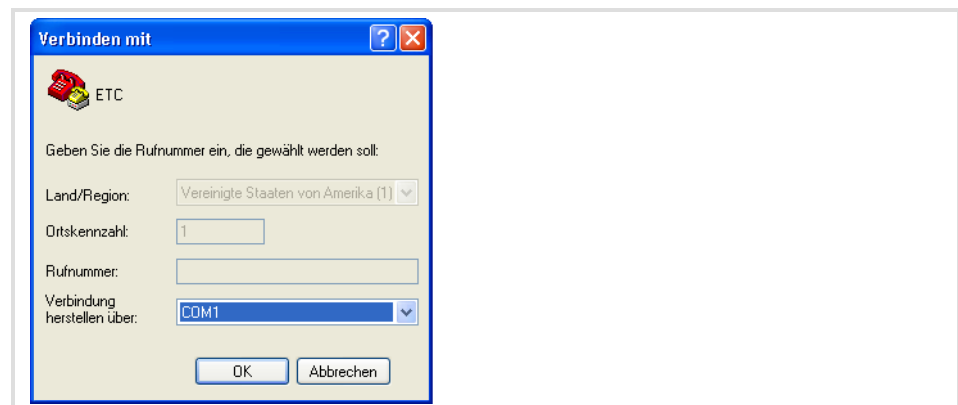
Für die Kommunikation zwischen PC und ETC benötigen Sie ein Terminal-Programm. Das Terminalprogramm "HyperTerminal" steht in jeder Standard-Windows-Installation zur Verfügung.

1. Falls Sie bereits eine Verbindung mit einer ETC hergestellt hatten (d. h. falls bereits eine Verbindung konfiguriert ist), starten Sie die Verbindung mit **<Start> → Programme → Zubehör → Kommunikation → HyperTerminal → Verbindungsname.ht**. Weiter im nächsten Abschnitt.

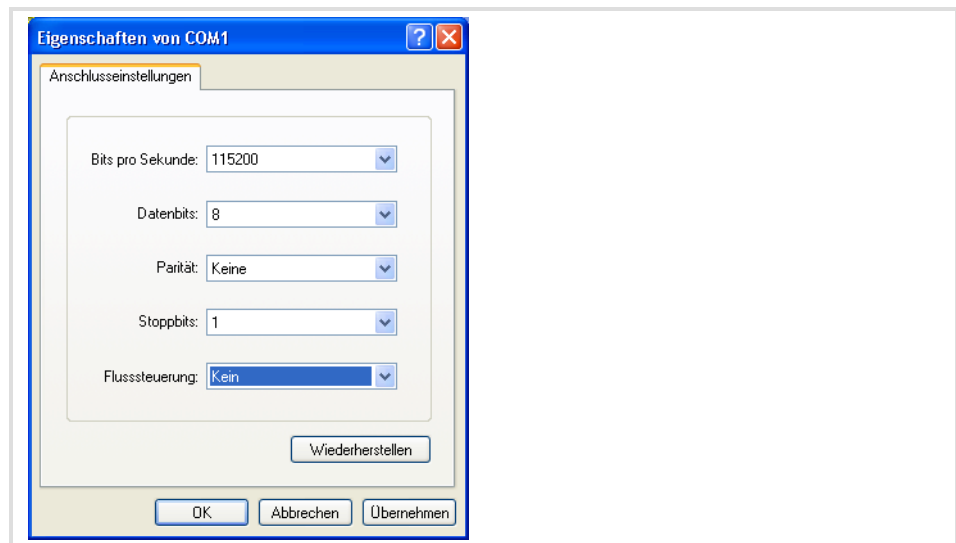
Falls noch keine Verbindung konfiguriert ist, starten Sie das HyperTerminal über **<Start> → Programme → Zubehör → Kommunikation → HyperTerminal**.

Das Programm fragt verschiedene Einstellungen des Telefonanschlusses ab, da es auch für eine Modem-Verbindung gedacht ist. Diese Einstellungen sind hier nicht von Bedeutung.

2. Nach Aufforderung vergeben Sie einen Namen für die Verbindung (zum Beispiel "ETC") und klicken auf **<OK>**.
3. Wählen Sie im Dialog "Verbinden mit" die PC-Schnittstelle, über die Sie die Verbindung herstellen möchten (zum Beispiel "COM1").



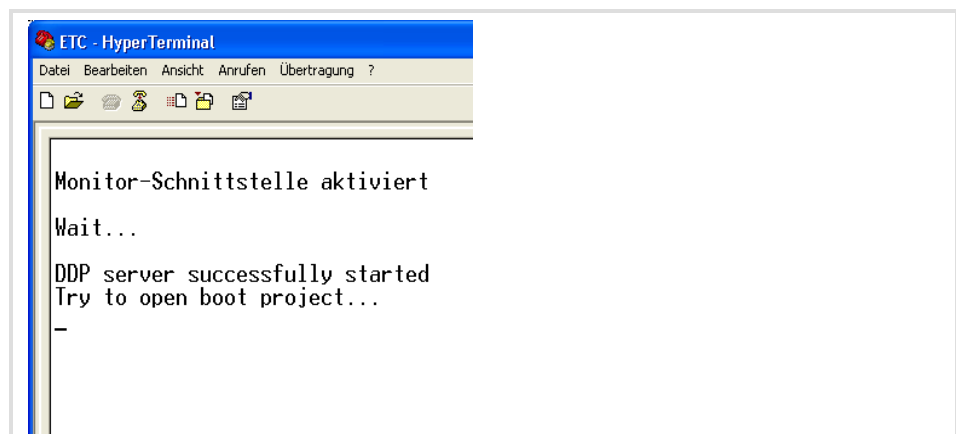
4. Klicken Sie auf **<OK>**.
5. Geben Sie im Dialog "Eigenschaften von COMx" folgende Daten ein:



6. Klicken Sie auf **<OK>**.



Die Meldung "Wait Boot Loader" erscheint und anschließend folgendes Fenster:



7. Drücken Sie die Taste **<?>** bis der Prompt **">"** erscheint.

Kommunikation zwischen PC und ETCHx herstellen Terminalprogramm "HyperTerminal" starten und Monitorschnittstelle aktivieren



Die Monitorschnittstelle ist gestartet. Sie können jetzt über die Tastatur Befehle zur ETC senden.

Wichtige Befehle des Monitorprogramms

Befehl	Bedeutung
dir [dr:][pattern]	Zeigt das Inhaltsverzeichnis des angegebenen Laufwerks an. Dabei bedeutet: ● Flashdisk/Silicon-Disk (sd:) ● Ram-Disk (rd:) ● Programmspeicher (ps:) ● CMOS-RAM (cr:) ● SD-Karte (sc:) ● USB-Massenspeicher (ms:) ● PC-Disk (pc:) ● Net-Disk (nd:) Als Pattern sind die gängigen MS-DOS-Pattern verwendbar, z. B.: *.din
del [dr:][pattern]	Löscht die angegebenen Dateien auf dem angegebenen Laufwerk. Voreingestellt ist auch hier die Flash-Disk (sd:).
cd [dr:]	Wechselt das aktuell ausgewählte Laufwerk bzw. zeigt es an. Voreingestellt ist die Flash-Disk (sd:).
format dr:	Formatiert das aktuelle Laufwerk und löscht damit alle darauf vorhandenen Dateien. Ein Laufwerkname muss angegeben werden.
reboot	Reset der Steuerung. Um den Bootmonitor zu aktivieren, muss anschließend die Taste "!" gedrückt werden, bis sich der Bootmonitor meldet (kann 3-4 Sekunden dauern).
help	Hilfeseite mit den Befehlen des Monitorprogramms anzeigen.
ver	Versionstring anzeigen
ipconf	Netzwerkparameter für Ethernetschnittstelle konfigurieren
hwconf	Hardware-Konfiguration der Steuerung anzeigen (CPU-Typ, Speicher usw.)

Störungsbeseitigung

Falls mit der beschriebenen Vorgehensweise keine Verbindung zur ETCHx aufgebaut wird, gehen Sie wie folgt vor:

1. Kontrollieren Sie die Verkabelung zwischen PC und ETCHx.
2. Kontrollieren Sie die Verbindungsparameter des Terminalprogramms.

Ein erster Test der Kabelverbindung lässt sich mittels einer Brücke zwischen den Pins 2 und 3 an der Buchse des Kabels (Steuerungsseite) durchführen:

- Das Terminalprogramm wird danach wie bisher aufgerufen und beliebige Buchstaben über die Tastatur eingegeben.
- Falls die eingegebenen Zeichen der Tastatur in der Maske des Terminalprogramms korrekt angezeigt werden (die Zeichen werden durch die Brücke als Echo zurückgesandt), ist das Kabel auf eine vertauschte Pinbelegung (Pin2/3), auf Kurzschluss und die Verbindung auf richtige Parametrierung zu kontrollieren.

2.2.4 IP-Adresse der ETCHx zuweisen

Für die Kommunikation über ein Netzwerk oder ein lokales Ethernet-Kabel benötigt die ETCHM eine eindeutige und zu den anderen Teilnehmern passende IP-Adresse (mit Subnet-Maske). Bei Auslieferung der Steuerung hat die Steuerung eine bestimmte, aber zufällige IP-Adresse.

Die IP-Adresse der ETCHM weisen Sie über die Monitorschnittstelle zu, wie nachfolgend beschrieben. Diese IP-Adresse wird bei der Erstellung von Verbindungen im CoDeSys abgefragt (📖 215).

Bilden Sie die IP-Adresse nach den folgenden allgemeinen Regeln für IP-Adressen:

Separates (lokales) Netzwerksegment

Sind Steuerung und PC mit einem lokalen Netzwerksegment oder durch eine direkte Ethernet-Verbindung verbunden (ohne Internet-/ Intranet-Anschluss), ordnen Sie den beiden Netzwerkstationen IP-Adressen der Klasse B oder Klasse C zu.

- ▶ Adressen der Klasse B liegen im Bereich von 128.0.0.0 bis 191.255.255.255. Die entsprechende Subnetz-Maske ist 255.255.0.0 (default).
- ▶ Adressen der Klasse C liegen im Bereich von 192.0.0.0 bis 223.255.255.255. Die entsprechende Subnetz-Maske ist 255.255.255.0.

Die Gateway-Adresse ist hier nicht relevant. Sie können eine freie IP-Adresse des Subnetzes eingeben (aber nicht 0.0.0.0).

Beispiel	IP-Adresse	Subnetz-Maske	Gateway-Adresse
Diese Adresse ist der ETCHM zuzuweisen und als Verbindungsparameter im CoDeSys einzugeben (📖 215):	181.16.1.10	255.255.0.0	181.16.1.1
Diese Adresse ist in der Netzwerkverbindung des PCs einzugeben (TCP/IP-Einstellungen):	181.16.1.11	255.255.0.0	181.16.1.1



Hinweis!

Verwenden Sie in IP-Adressen keine führenden Nullen, da die IP-Adresse sonst als Oktadezimalzahl interpretiert wird.

Existierendes Netzwerksegment

Sollen ETCHM und PC mit einem existierenden Netzwerksegment verbunden werden, müssen die Adressen und die Subnetz-Maske vom Netzwerkadministrator vergeben werden.



Hinweis!

Nur Gerätegeneration 1: Wenn der Ethernet-Adapter 10Base-T-Technologie unterstützt, muss für die "Media Type"-Eigenschaft "Auto Select" oder "10Base-T" eingestellt sein.

MAC –Adresse

Die ETCHM wie auch jedes andere Gerät mit Ethernet-Controller erhält vom Hersteller eine unveränderbare und weltweit eindeutige physikalische Ethernet-Adresse, auch MAC-ID (Media Access Control Identity) genannt. Diese kann zur Adressierung auf Hardware-Ebene verwendet werden. Die Adresse besitzt eine feste Länge von 6 Bytes (48 Bit) und beinhaltet einen Adresstyp, eine Kennzeichnung des Herstellers und eine Seriennummer.

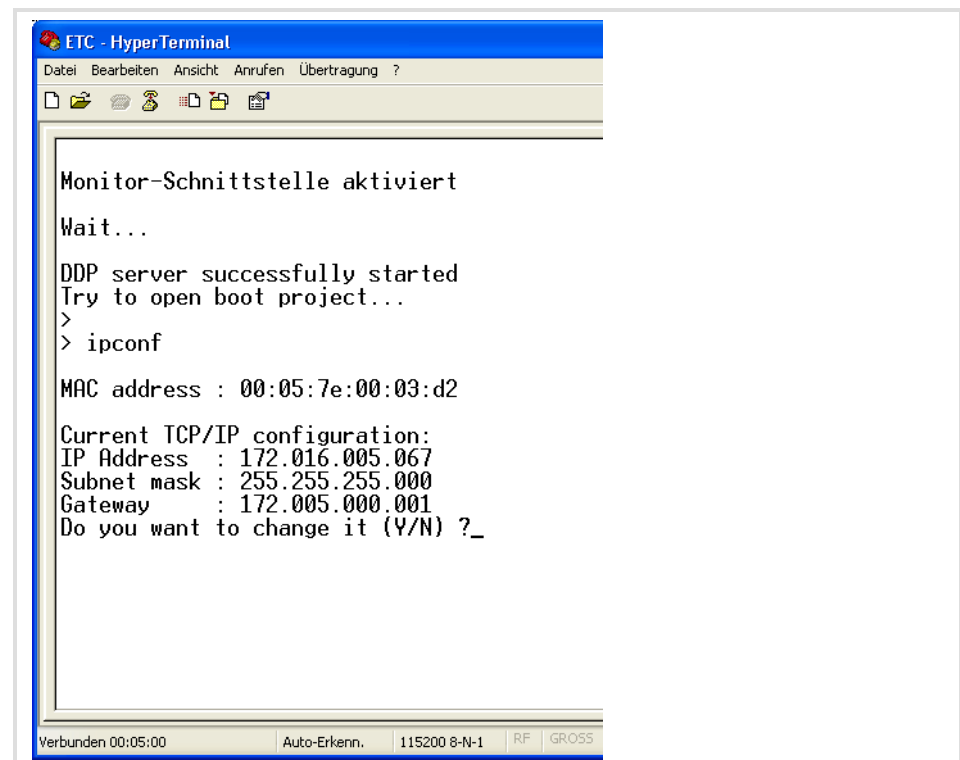
MAC-ID der ETC: 00:05:7e:xx:xx:xx (mit xx:xx:xx = Seriennummer).

IP-Adresse zuweisen

Die IP-Adresse der ETCHM stellen Sie über die Monitorschnittstelle ein.

1. Geben Sie im HyperTerminal den Befehl **ipconf** ein und bestätigen Sie den Befehl mit **<Enter>**.

Die MAC-ID und die aktuelle Adresseinstellung der ETCHM werden angezeigt.



2. Drücken Sie die Taste **<Y>** und ändern Sie die Einstellungen. Folgen Sie dabei den Anweisungen der Monitorschnittstelle.
Wählen Sie eine IP-Adresse gemäß den am Anfang dieses Abschnitts genannten Regeln.
Das Programm HyperTerminal kann jetzt beendet werden.

**Hinweis!**

Notieren Sie die IP-Adresse, da sie bei der Erstellung von Verbindungen im CoDeSys eingegeben werden muss (📖 215).

2.3 Antriebe über Maschinenkonstanten parametrieren

Die Eigenschaften der Antriebe müssen sowohl im Antrieb selbst als auch in der Steuerung parametrieren werden.

In der Steuerung werden die Eigenschaften über Maschinenkonstanten (MKs) zugewiesen. Eine Maschinenkonstante besteht aus einem Schlüsselwort und den zugehörigen Werten.

Die Maschinenkonstanten sind auf dem PC in einer Textdatei mit der Endung .mk (z. B. ETCHM.mk für die ETCHM) gespeichert und werden mit dem Hyperterminal oder einem MMI in die Steuerung geladen.

2.3.1 Übersicht der wichtigsten Maschinenkonstanten

Die im Folgenden aufgeführten Maschinenkonstanten sind zwingend notwendig zum Betrieb eines Antriebs. Sie stellen bezogen auf die Antriebe nur eine Untermenge der notwendigen Parametrierung dar.



Stop!

Die Steuerung initialisiert über ihre Maschinenkonstanten nur die Antriebe und gibt bahnbezogene Parameter vor. Die regelungstechnischen Parameter, die Sicherheitsfunktionen der Antriebe und die Parameter der Referenzfahrt müssen mit Hilfe des GDC (Global Drive Control) im Antrieb selbst parametrieren werden.

Name	Bedeutung
MK_TEST_WITHOUT_MECHANICS	Dient dem Test der Steuerungsfunktionen, ohne dass eine Maschine angeschlossen werden muss. Initialisierungswert: 0
MK_INTERPOL_CYCLE	Zykluszeit des Interpolators. Stellt den internen Interpolationstakt [ms] der Steuerung ein. In jedem solchen Grobinterpolationstakt berechnet die Steuerung neue Positionswerte für die Achsen (siehe folgendes Kapitel). Einheit: [ms] Initialisierungswert: 4 ms
MK_PLC_CODE_SIZE	Reserviert den Programmspeicher für die SPS im internen Arbeitsspeicher der Steuerung und legt damit gleichzeitig die maximale Größe des zu ladenden SPS-Programms in Kilobytes fest. Die Steuerung reserviert das Doppelte des in MK eingegebenen Speichers, um die Funktion Online-Change von CoDeSys zu unterstützen. Einheit: KB Initialisierungswert: 480 KB
MK_PLC_DATA_SIZE	Legt die Größe des Datenspeichers für die Variablen der SPS in Kilobytes fest. Einheit: KB Initialisierungswert: 128 KB

Name	Bedeutung																		
MK_LANGUAGE	Wählt eine Textdatenbank zur Anzeige von länderspezifischen Texten aus. Um eine Sprache auswählen zu können, müssen die jeweiligen Textdateien in das Flash-PROM der Steuerung geladen werden. Die Namen der Textdateien müssen lauten: ● NCRSPCHn.TXT für Fehlertexte ● SPSSPCHn.TXT für SPS-Texte Wobei "n" eine Ziffer von 0 ... 9 darstellt, die über die Maschinenkonstante ausgewählt wird. Initialisierungswert: 0																		
MK_NETDISK_SRVADDR	Legt eine Voreinstellung für die IP Adresse des PC's fest, auf dem der Netdisk-Server läuft. Initialisierungswert: ":" Beispiel: MK_NETDISK_SRVADDR "172.16.5.111";																		
MK_CAN_DRIVE_NODE_ID	Antriebsadresse/Knotennummer. 1. Gerätegeneratuin: 1 ... 12 2. Gerätegeneration: 1 ... 64 Initialisierungswert: -1 für alle Achsen Wenn eine Achse mit der Antriebsadresse -1 mit MC_DefineAxis definiert wird, wird sie als virtuelle Achse betrachtet.																		
MK_AXIS_TYPE	Spezifiziert verschiedene Achseigenschaften. Die MK ist Bit-codiert aufgebaut. <table><tr><th>Bit</th><th>Wert</th><th>Bedeutung</th></tr><tr><td rowspan="2">0</td><td>0</td><td>Linearachse</td></tr><tr><td>1</td><td>Rotationsachse</td></tr><tr><td rowspan="2">1</td><td>0</td><td>Endschalter aktiv</td></tr><tr><td>1</td><td>Endschalter nicht aktiv</td></tr><tr><td rowspan="2">3</td><td>0</td><td>Normale Achse</td></tr><tr><td>1</td><td>Geber</td></tr></table>	Bit	Wert	Bedeutung	0	0	Linearachse	1	Rotationsachse	1	0	Endschalter aktiv	1	Endschalter nicht aktiv	3	0	Normale Achse	1	Geber
Bit	Wert	Bedeutung																	
0	0	Linearachse																	
	1	Rotationsachse																	
1	0	Endschalter aktiv																	
	1	Endschalter nicht aktiv																	
3	0	Normale Achse																	
	1	Geber																	
MK_INCREMENTS	Bestimmt die Anzahl der Impulse, die dem Weg (MK_DISTANCE) in Technischer Einheit [u] äquivalent sind. Einheit: keine Initialisierungswert: 65536 für alle Achsen																		
MK_DISTANCE	Bestimmt die Anzahl der Technischen Einheiten [u], die der Anzahl der Impulse (MK_INCREMENTS) äquivalent sind. Positionen von Rotationsachsen werden modulo MK_DISTANCE berechnet. Einheit: [u] (Technische Einheit) Initialisierungswert: 10 für alle Achsen																		
MK_HOME_OFFSET	Der Versatz des mechanischen Nullpunktes der Maschine zum Nullpunkt des Wegmesssystems. Einheit: [u] Initialisierungswert: 0 für alle Achsen Hinweis: diese Maschinenkonstante kann für die Bestimmung des Eingangsparameters Position des Funktionsbausteins MC_Home eingesetzt werden.																		
MK_SW_LIMIT_NEGATIVE	Definiert die negative Verfahrbereichsgrenze der Mechanik. Einheit: [u] Initialisierungswert: -100000 für alle Achsen																		
MK_SW_LIMIT_POSITIVE	Definiert die positive Verfahrbereichsgrenze der Mechanik Einheit: [u] Initialisierungswert: 100000 für alle Achsen																		

Antriebe über Maschinenkonstanten parametrieren Übersicht der wichtigsten Maschinenkonstanten

Name	Bedeutung
MK_MAX_VELOCITY	Die maximale Geschwindigkeit der einzelnen Achsen. Einheit: [u/s] Initialisierungswert: 100 für alle Achsen
MK_MAX_ACCELERATION	Die maximale Beschleunigung der einzelnen Achsen. Einheit: [u/s ²] Initialisierungswert: 1000 für alle Achsen
MK_MAX_DECELERATION	Die maximale Verzögerung der einzelnen Achsen. Einheit: [u/s ²] Initialisierungswert: 1000 für alle Achsen
MK_MAX_JERK	Der maximale Ruck der einzelnen Achsen. Einheit: [u/s ³] Initialisierungswert: 10000 für alle Achsen
MK_CAN_DRIVE_BAUDRATE	Baudrate für Drive-CAN-Bus Einheit: [kBits/s] Initialisierungswert: 1000
MK_FILTER_TIME_CONST	Zeitkonstante für Positionsfilter eines Gebers. Gilt nur für Achsen mit MK_AXIS_TYPE = 8 (Geber). Der Positionsfilter ist ein Filter 2. Ordnung, der die durch einen Geber gelieferten Positionen glättet. Der Filter führt zu einer Positionsverzögerung, die ca. 2mal so groß ist, wie diese Maschinenkonstante. Einheit: [s] Initialisierungswert: 0 für alle Achsen
MK_TECHNOLOGY_DATA_1 MK_TECHNOLOGY_DATA_2 MK_TECHNOLOGY_DATA_3 MK_TECHNOLOGY_DATA_4	Frei verwendbare Maschinenkonstanten. Dies MK's enthalten jeweils 10 Einträge. Der Wertebereich jedes Eintrags entspricht einem 64-bit Floating point Wert (IEC61131-Datentyp LREAL). Die einzelnen Einträge können über die Funktion GetUserParam() von der SPS ausgelesen werden. Einheit: Benutzerabhängig Initialisierungswert: 0 für alle Einträge
MK_REGLER_MODE	Zur Zeit nur für interne Verwendung

2.3.2 Maschinenkonstanten-Datei ETCxM.mk

Das folgende Beispiel von Maschinenkonstanten für 12 Achsen ist ein Ausschnitt aus der Datei ETCHM.mk mit voreingestellten Maschinenkonstanten, die für den konkreten Anwendungsfall angepasst werden müssen. Die Datei ETCHM.mk wird per Hyperterminal in die Steuerung geladen.

```

MK_TEST_WITHOUT_MECHANICS 0;
MK_PLC_CODE_SIZE 400;
MK_PLC_DATA_SIZE 512;
MK_CAN_DRIVE_NODE_ID 5,7,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1;
MK_CAN_DRIVE_BAUDRATE 1000;
MK_AXIS_TYPE 0,0,0,0,0,0,0,0,0,0,0,0;
MK_LANGUAGE 0;
MK REGLER_MODE 2,2,2,2,2,2,2,2,2,2,2,2;
MK_INTERPOL_CYCLE 4;
MK_INCREMENTS 65536,65536,65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, ...
MK_DISTANCE 10,10,10,10,10,10,10,10,10,10,10,10;
MK_HOME_OFFSET 0,0,0,0,0,0,0,0,0,0,0,0;
MK_SW_LIMIT_NEGATIVE -10000,-10000,-10000,-10000,-10000,-10000,-10000,-10000,-10000,...
MK_SW_LIMIT_POSITIVE 10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000,...
MK_FILTER_TIME_CONST 0,0,0,0,0,0,0,0,0,0,0,0;
MK_MAX_VELOCITY 1000, 1000, 1000, 10000, 10000, 10000, 10000, 1000, 1000, 1000, 1000,...
MK_MAX_ACCELERATION 15000,15000,15000,10000,10000,10000, 10000, 10000, 10000, 10000, 10000,...
MK_MAX_DECELERATION 15000,15000,15000,10000,10000,10000, 10000, 10000, 10000, 10000, 10000,...
MK_MAX_JERK 100000, 100000, 100000, 100000, 100000, 100000, 100000, 100000, 100000, 100000,...
MK_MAX_TORQUE 1,1,1,1,1,1,1,1,1,1,1,1;
MK_MAX_TORQUE_RAMP 10,10,10,10,10,10,10,10,10,10,10,10;
MK_TORQUE_TECH_UNITS_DENOM 1,1,1,1,1,1,1,1,1,1,1,1;
MK_TORQUE_TECH_UNITS_NUM 1,1,1,1,1,1,1,1,1,1,1,1;
MK_VELOCITY_TECH_UNITS_DENOM 1,1,1,1,1,1,1,1,1,1,1,1;
MK_VELOCITY_TECH_UNITS_NUM 1,1,1,1,1,1,1,1,1,1,1,1;
MK_TECHNOLOGY_DATA_1 0,0,0,0,0,0,0,0,0,0,0,0;
MK_TECHNOLOGY_DATA_2 0,0,0,0,0,0,0,0,0,0,0,0;
MK_TECHNOLOGY_DATA_3 0,0,0,0,0,0,0,0,0,0,0,0;
MK_TECHNOLOGY_DATA_4 0,0,0,0,0,0,0,0,0,0,0,0;
MK_NETDISK_SRVADDR " ";

```

2.4 ETC-SPS-Programmierung mit CoDeSys

Das Werkzeug ETC-CoDeSys ist eine komplette Entwicklungsumgebung zum Erstellen und Testen von SPS-Programmen für die ETC.

Es basiert auf dem weit verbreiteten Programmpaket CoDeSys mit den spezifischen Erweiterungen für die ETC-Steuerungen.



Hinweis!

Weitere Informationen erhalten Sie im Kapitel "SPS-Programmierung" (📖 213).

2.4.1 CoDeSys installieren

1. Legen Sie die CoDeSys-Setup-CD in Ihr CD-ROM-Laufwerk.
Wenn das Autostart-Feature des CD-ROM-Laufwerks in Windows aktiviert ist, startet das Setup automatisch.
2. Sollte das Setup nicht automatisch starten:
 - Wählen Sie im Startmenü den Menüpunkt "Ausführen".
 - Geben Sie in der Befehlszeile den Laufwerksbuchstaben Ihres CD-ROM-Laufwerks gefolgt von "setup.exe" ein (z. B. "d:\setup.exe") und bestätigen Sie mit "OK".
3. Folgen Sie den Anweisungen des Setup-Programms.
CoDeSys wird im Verzeichnis "c:\programme\lenze\CoDeSys2 V2.3" installiert.



Stop!

Die Datei "codesys.ini" im Installationsverzeichnis kann nach der Installation von CD schreibgeschützt sein. Der Schreibschutz muss aufgehoben werden, da sonst die CoDeSys-Einstellungen nicht gespeichert werden.

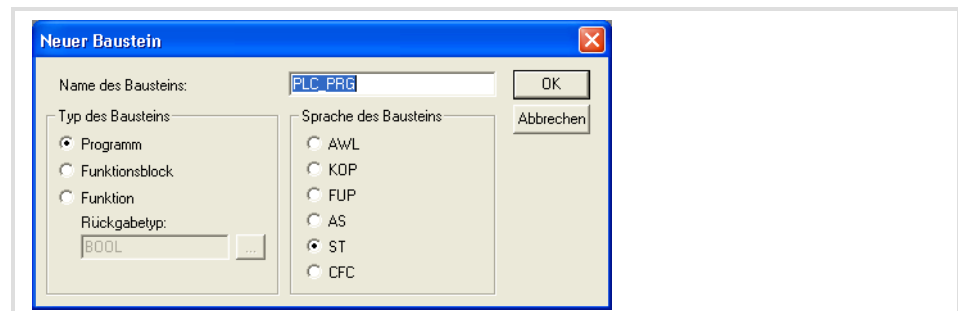
2.4.2 Steuerung im ETC-CoDeSys konfigurieren

Dieses Kapitel beschreibt die generelle Vorgehensweise bei der Steuerungskonfiguration als Voraussetzung zum Erstellen eines Programms mit dem ETC-CoDeSys am Beispiel einer ETCHx.

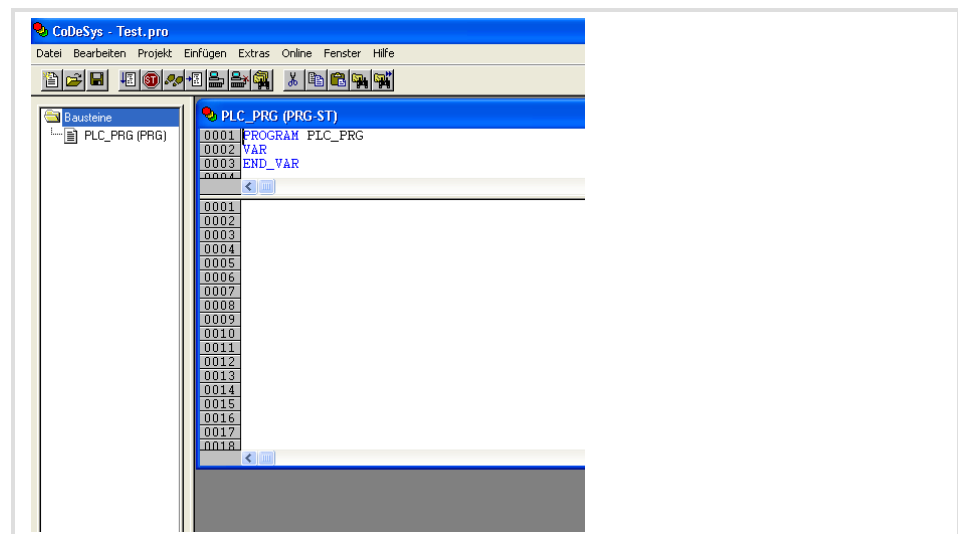
Dieses Kapitel kann übersprungen werden, wenn das Beispielprogramm (📖 31) aufgerufen wird, da die beschriebenen Einstellungen dort schon vorhanden sind.

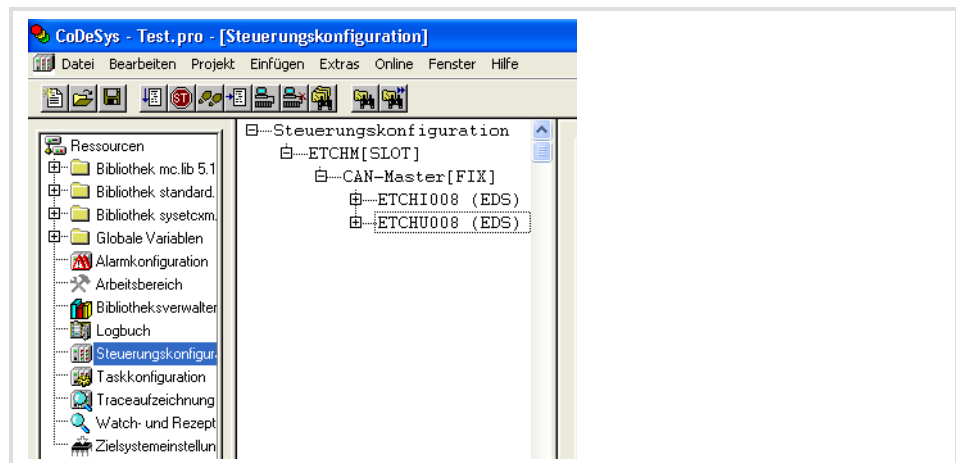
ETC-CoDeSys starten und
neues Projekt anlegen

1. Starten Sie das ETC-CoDeSys über **<Start> → Programme → Lenze → CoDeSys**.
2. Bei Bedarf können Sie die Sprache im CoDeSys umschalten:
 - **Projekt → Optionen → Arbeitsbereich → Sprache = englisch**
 - **Project → Options → Desktop → Language = german**
3. Erstellen Sie ein neues Projekt über **Datei → Neu**.
4. Wählen Sie im Dialog "Zielsystemeinstellungen" die entsprechende Steuerung.
5. Akzeptieren Sie die Voreinstellung.
Ein Dialog für den ersten Baustein erscheint. Der Name ist frei wählbar (z. B. PLC_PRG). Als erster Baustein wird in der Voreinstellung ein Programm in der Sprache ST (strukturierter Text) erstellt.



6. Drücken Sie **<OK>**
7. Speichern Sie das neue Projekt über **Datei → Speichern**.





Hinweis!

Neue Module werden an das Ende der Liste gestellt. Um ein neues Modul vor ein bestehendes einzufügen, das bestehende anwählen und über die rechte Maustaste mit **Element einfügen** einfügen.

Ein Modul kann aus der Liste über die rechte Maustaste und **Löschen** entfernt werden.

Beschreibung des Moduls in der EDS-Datei

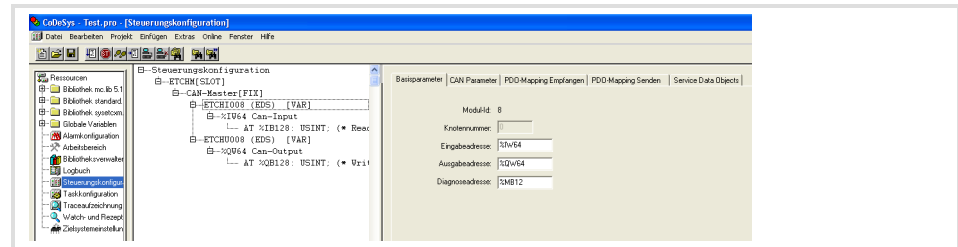
Jedes in der Steuerung zu konfigurierende Modul benötigt eine EDS-Datei. Die zum Modul gehörige EDS-Datei muss in dem jeweiligen Target-Verzeichnis der ausgewählten Steuerung (hier Lenze) liegen.

Eine EDS-Datei enthält Einträge von Schlüsselworten, die mit den entsprechenden Werten die Eigenschaft des Moduls beschreiben.

Bei Modulen von Fremdherstellern müssen die entsprechenden EDS-Dateien in das Target-Verzeichnis der Steuerung kopiert werden.

Module parametrieren

Nach Anklicken der einzelnen Module sieht das Bild wie folgt aus:



Dem Programm muss mitgeteilt werden, an welchen Adressen des Speicherbereichs die Eingabe- und Ausgabedaten der Module liegen. Die Voreinstellung im Programm sind Byte-Adressen (%IB, %QB), die in Wort-Adressen (%IW, %QW) geändert werden müssen, da das Prozessabbild der ETC wortweise organisiert ist. Der Anfang des Adressbereichs liegt dabei bei Wort 64. Bei der Änderung in Wort-Adressen ist wie folgt vorzugehen:

- Beginnend mit dem ersten Modul in der Liste (also hier ETCHI008) in der Reihenfolge der Module (also ETCHU008) den Typ der Variablen %IB, %QB in %IW, %QW ändern. Dabei erhält das erste Modul als Adresse Wort 64 und die folgenden Module erhalten als Wortadresse Wort 64 plus der Byte-Adresse minus 128.

Modul	angezeigte Eingabe- adresse	angezeigte Ausgabe- adresse		Eingabe- Eingabe- adresse	Eingabe- Ausgabe- adresse	Diagnose- adresse
ETCHI008	%IB128	%QB128	→	%IW64	%QW64	unverändert
ETCHU008	%IB129	%QB128	→	%IW65	%QW64	
Nächstes Modul	%IB129	%QB130	→	%IW65	%QW66	

Die Diagnoseadresse muss nicht geändert werden.

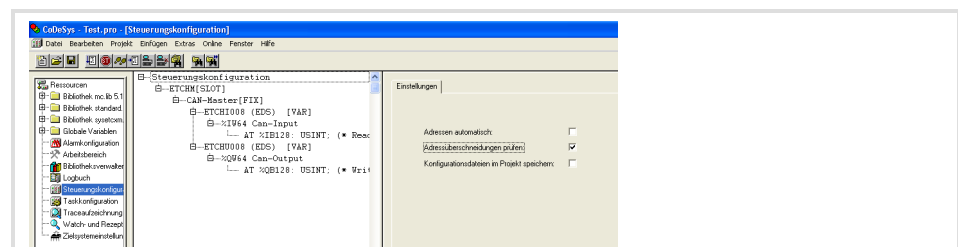
**Hinweis!**

Die beschriebene Methode der Adressvergabe vermeidet Adressüberschneidungen der einzelnen Module, vergibt aber die Adressen nicht platzoptimal.

Für eine platzoptimale Vorgabe der Adressen folgender Module müssen diese errechnet werden, worauf hier aus Vereinfachungsgründen verzichtet wird.

Adressen automatisch

In der Steuerungskonfiguration sollte das Häkchen "Adressen automatisch" nicht gesetzt werden, da sonst CoDeSys die Adressen bei Änderungen der Steuerungskonfiguration neu vergibt.



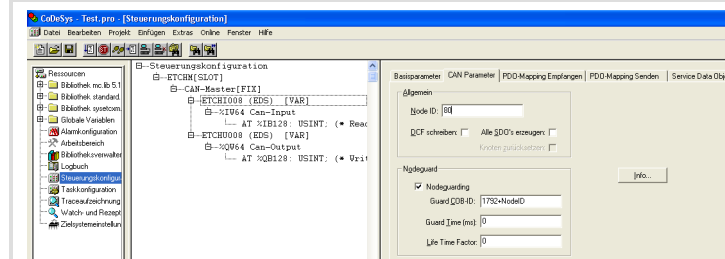
CAN-Adresse eingeben

Jedes Modul hat eine eigene CAN-Adresse (Node-ID). Eine Adresse setzt sich aus der typspezifischen Basisadresse und der individuell eingestellten Adresse am Hex-Schalter auf der Frontseite zusammen.



Wenn für die eingesetzten Feldbus-Module der HEX-Schalter auf Null eingestellt ist (Pfeil zeigt nach links), müssen bei den einzelnen Modulen folgende resultierende Node-ID eingetragen werden:

Modul	CAN-Basisadresse	Eingestellt am Hex-Schalter	Resultierende Node-ID
ETCHI0xx	80 _{dez}	0	80 _{dez}
ETCHU0xx	16 _{dez}	0	16 _{dez}
ETCHA022	48 _{dez}	0	48 _{dez}



Nodeguarding

Ist die Option **Nodeguarding** aktiviert, wird im Intervall, das bei **Guard Time** in Millisekunden angegeben ist, eine Nachricht an das Modul verschickt. Meldet sich das Modul daraufhin nicht mit einer Antwort und ist die Anzahl der Versuche (**Life Time Factor**) erreicht, so gilt das Modul als nicht NOK (ausgefallen oder nicht vorhanden).

Die Standard-Einstellung ist bei allen Modulen **Guard Time = 500** und **Life Time Factor = 3**. Diese Einstellung bedeutet ein Erkennen eines Ausfalls nach $500 \text{ ms} \times 3 = 1,5 \text{ s}$. Wenn keine Guard Time und kein Life Time Factor angegeben ist (= 0), wird das Modul nicht überwacht.

Weitere Einstellungen sind nicht erforderlich.

Taskkonfiguration

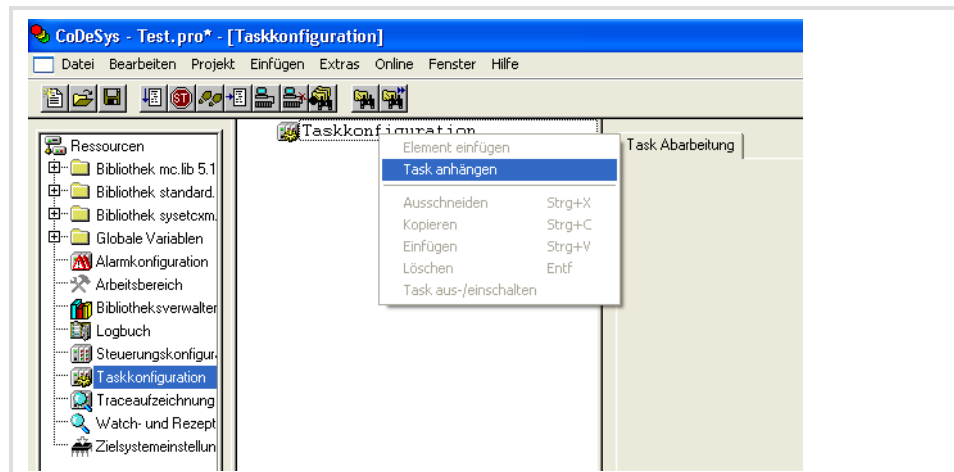
Mit der ETCxM ist es möglich, vier voneinander unabhängige SPS-Tasks zu nutzen. Jede Task kann mit einem Aufruf eines IEC-Programmbausteins (PRG) verknüpft sein. Die mit den Tasks verknüpften Programmbausteine werden von dem Multitaskingbetriebssystem der ETCxM "quasiparallel" bearbeitet. Die Abarbeitungsreihenfolge ergibt sich aus der Intervallzeit (zeitgesteuert) und der Priorität einer Task. Eine Task läuft also solange, bis ein SPS-Zyklus abgearbeitet wurde oder eine andere Task mit höherer Priorität laufen will.

**Stop!**

Bei Anwendung der Motion-Bibliothek "mc.lib" muss vermieden werden, dass mehrere Bewegungen gleichzeitig aus mehreren Tasks für die gleiche Achse gestartet werden. Das kann zu unerwarteten Ergebnissen führen.

Damit ein SPS-Programm durch die ETCxM abgearbeitet wird, muss in der Taskkonfiguration von CoDeSys mindestens eine Task angelegt sein und mit einem IEC-Programmbaustein verknüpft werden.

Die Taskkonfiguration ist im Falle der ETCxM zwingend nötig. Das Anhängen einer Task erfolgt über die rechte Maustaste "Task anhängen".



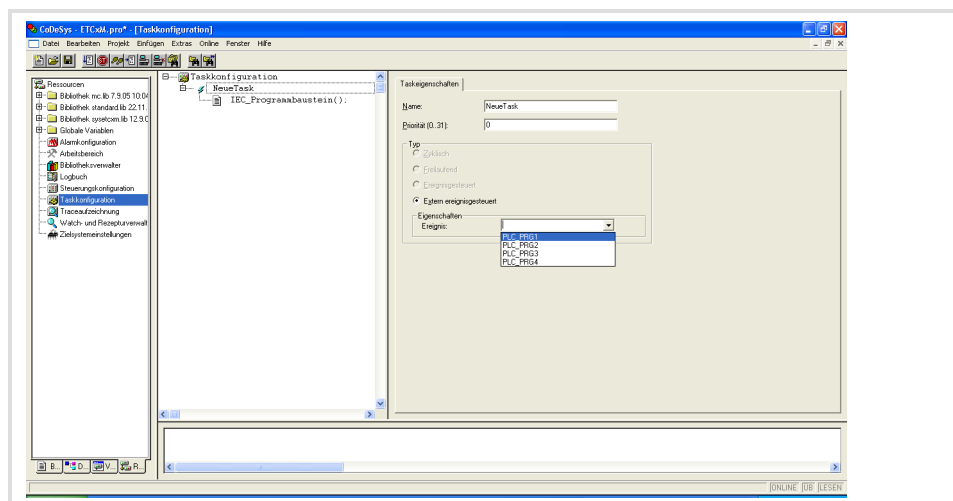
Taskeigenschaften

Die Eigenschaft jeder Task kann individuell eingestellt werden. Beim Anlegen einer zyklischen Task muss der Typ "Zyklisch" und anschließend der Typ "Extern ereignisgesteuert" eingetragen werden.

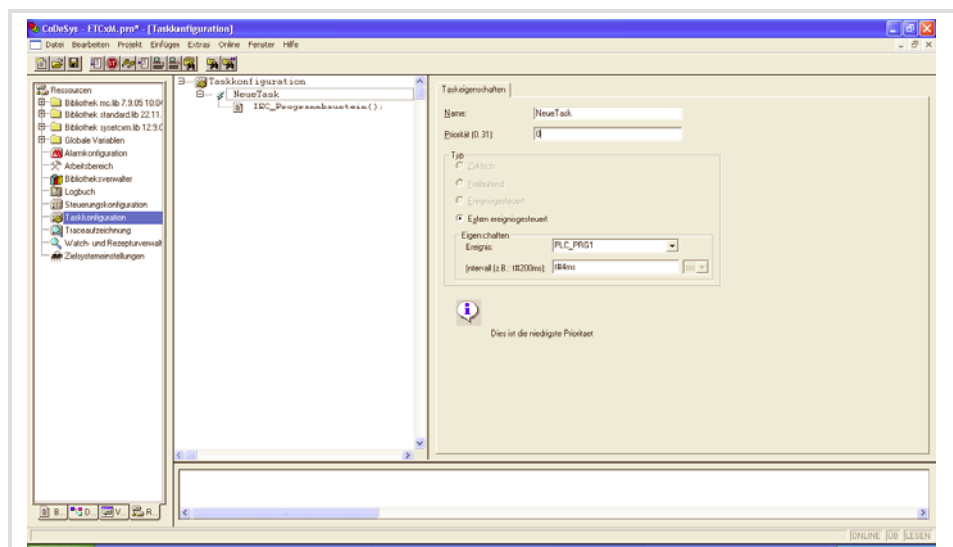
Ereignisvariable	Eigenschaft	Kommentar
PLC_PRG1, PLC_PRG2, PLC_PRG3	Intervall	Task wird mit dem eingestellten Zeitintervall aufgerufen.
PLC_PRG4	Intervall, Ereignis	Beim Auftreten eines Achs- oder Systemfehlers (unabhängig von der Fehlerklasse) wird die Task sofort außerhalb der konfigurierten Zykluszeit aufgerufen.

Beispiel einer zyklischen Task:

1. Auswahl der Ereignisvariable



2. Intervallzeit einstellen, indem das Programm zyklisch aufgerufen wird.



2.5 SPS-Beispielprogramm erstellen

Dieses Kapitel beschreibt die Erstellung eines SPS-Programms am Beispiel eines ETCHx-Systems.

2.5.1 SPS-Beispielprogramm starten und konfigurieren

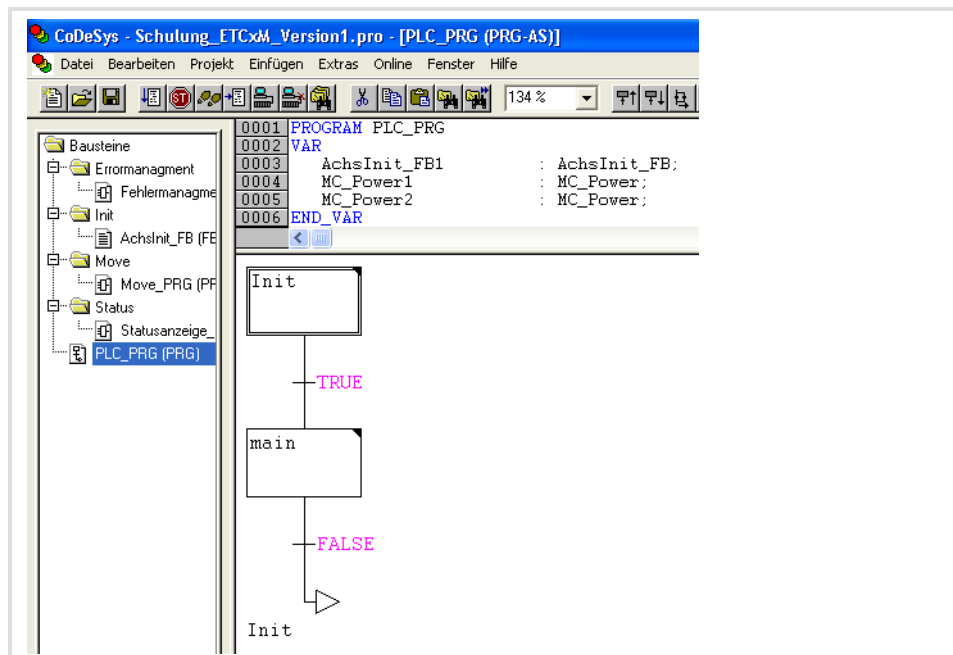
Das Beispielprogramm "Schulung_ETCxC_M_Version1" beinhaltet

- ▶ alle Einstellungen der Steuerung,
- ▶ die Parametrierung,
- ▶ die Aufrufe um mit der aufgeführten HW-Konfiguration zu kommunizieren.

Beispielprogramm starten

1. Klicken Sie im Windows-Explorer doppelt auf die Datei "Schulung_ETCxC_M_Version1".

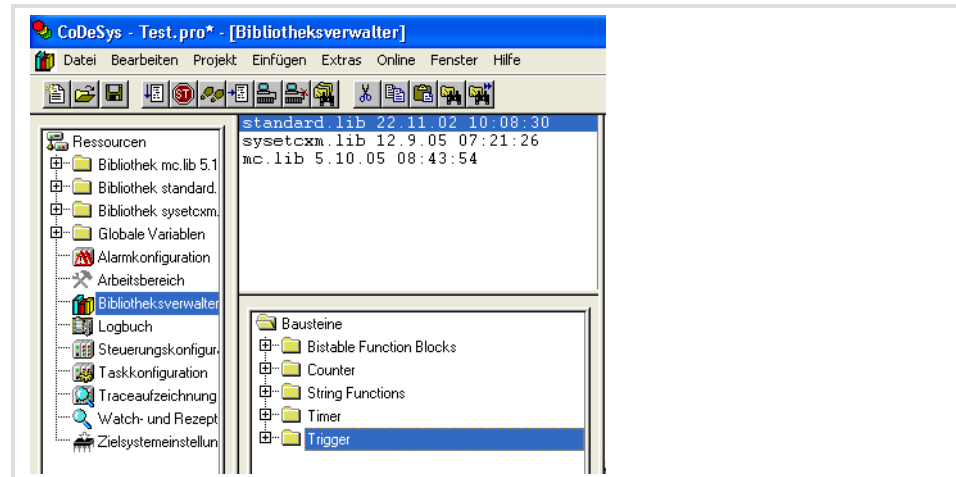
CoDeSys wird gestartet und das Beispielprogramm wird geladen. Es besteht aus einer Ablaufsteuerung (AS) mit den Schritten "Init" und "Main".



Bibliotheken einfügen

Das Beispielprogramm benötigt die Bibliotheken "standard.lib" und "mc.lib" (...\\CoDeSys V2.3\\Targets\\Lenze\\ETCxCM).

2. Fügen Sie, falls die Bibliotheken noch nicht vorhanden sind, diese über **Ressourcenverwaltung** → **Bibliotheksverwalter** ein.



Programmstruktur

Das Programm "Schulung_ETCxCM_Version1" besteht aus einer Task "Task 1". Diese Task ruft zyklisch das Programm "PLC_PRG" auf. Das Programm "PLC_PRG" ist als Ablaufsteuerung geschrieben und besteht aus 2 Schritten und 2 Transitionen. Der erste Schritt ist der Initialisierungsschritt, der einmalig aufgerufen wird. In den fortlaufenden Zyklen verweilt das Programm in dem Schritt "Main", indem weitere Programm und Funktionsbausteine aufgerufen werden.

Programm		PLC_PRG	
Element	Name	Inhalt des Elements	Kommentar
Schritt	Init		Initialisierung, Schritt wird einmalig beim Beginn durchlaufen.
Transition		True	Weiterschaltung auf den nächsten Schritt "Main".
Schritt	Main	Funktionsbaustein AchsInit FB	Initialisieren der Antriebsachse am Drive CAN.
		Funktionsbaustein MC Power	Alle Freigaben für eine Achse setzen oder zurücksetzen.
		Programm Fehlermanage- ment_PRG	Programmaufruf zur Behandlung evtl. Fehler und dem Quit-tieren.
		Programm Statusanzeige PRG	Programmaufruf für die Visualisierung des Achs-Status.
		Programm Move PRG	Programmaufruf zur Behandlung der Achsbewegungen.
Transition		False	Keine Weiterschaltung auf nächsten Schritt, da FALSE. Das Programm bleibt in dem Schritt "Main".

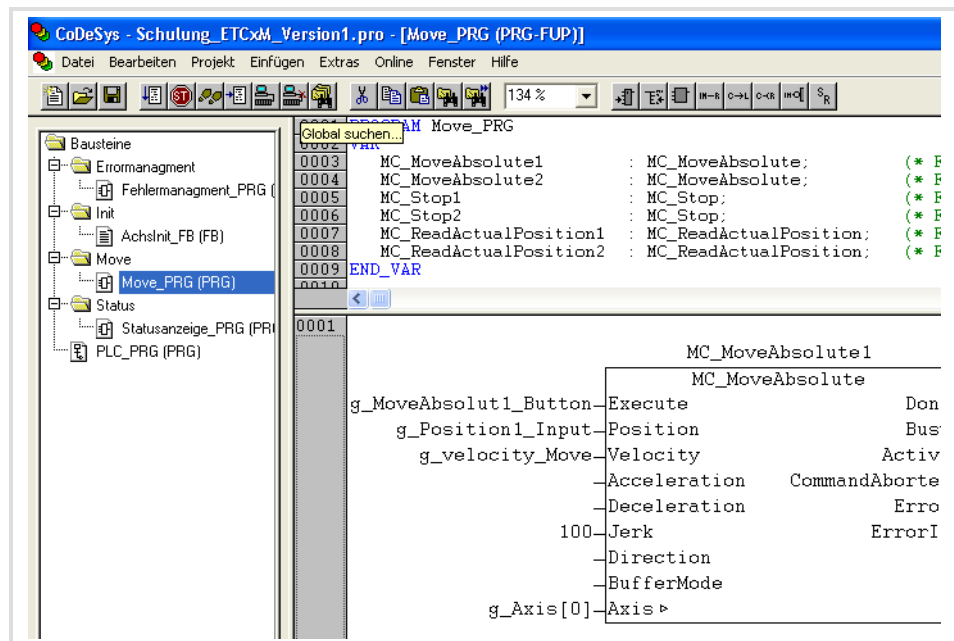
SPS-Beispielprogramm erstellen SPS-Beispielprogramm starten und konfigurieren

Programm		Fehlermanagement_PRG	
	Element	Name	Kommentar
	Funktionsbaustein	MC_ReadAxisError	Anzeigen der im Antrieb aufgetretenen Achsfehler.
	Funktionsbaustein	MC_Reset	Quittierung anstehender Fehlermeldungen und Zustandswechsel von ErrorStop in den Zustand Standstill.

Programm		Statusanzeige_PRG	
	Element	Name	Kommentar
	Funktionsbaustein	MC_ReadStatus	Status-Anzeige des jeweiligen Antriebs.

Programm		Move_PRG	
	Element	Name	Kommentar
	Funktionsbaustein	MC_MoveAbsolute	Status-Anzeige des jeweiligen Antriebs.
	Funktionsbaustein	MC_Stop	Stoppen des jeweiligen Antriebs.
	Funktionsbaustein	MC_ReadActualPosition	Ausgabe der aktuellen Istposition der jeweiligen Achse

Nach Doppelklick auf eines der Programme in der Bausteinstruktur öffnet ein Fenster, in dem die entsprechend hinterlegten Funktionen sichtbar werden. Als Beispiel dient das Programm "Move_PRG" welches in der Programmiersprache "Funktionsplan" (FUP) programmiert wurde.

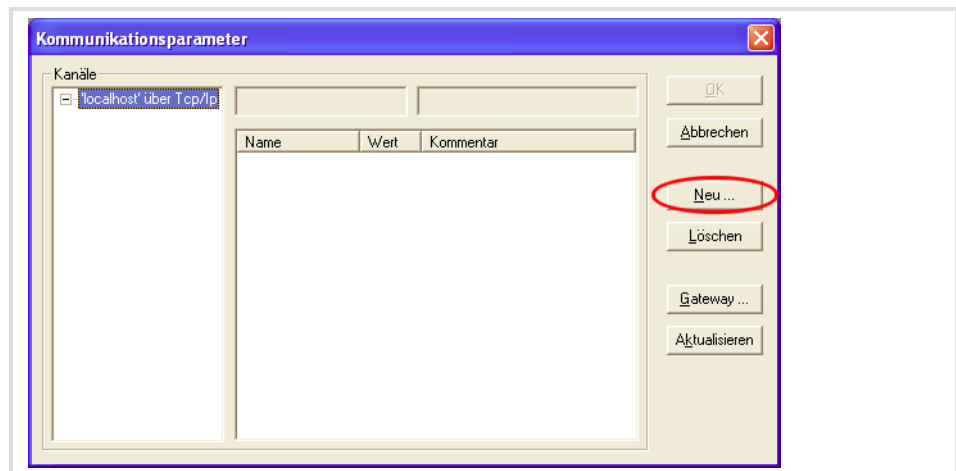


2.5.2 SPS-Beispielprogramm in die Steuerung laden

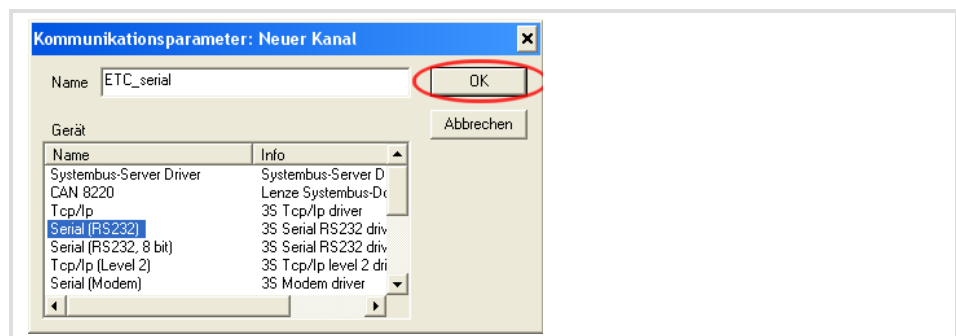
Über das ETC-CoDeSys kann die Verbindung vom PC zur Steuerung eingestellt werden, wahlweise über eine serielle Verbindung (RS232-Schnittstelle) oder über das Netzwerk.

Serielle Verbindung

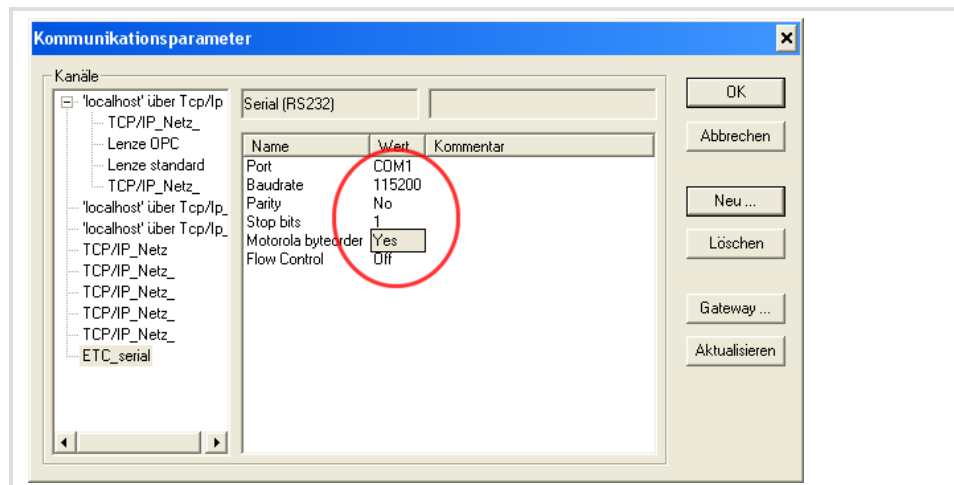
1. Verbinden Sie PC und ETCHC über die serielle Schnittstelle.
2. Wählen Sie **Online → Kommunikationsparameter**.
3. Klicken Sie auf **<Neu>**.



4. Geben Sie einen Verbindungsnamen ein (z. B. ETC_serial) und wählen Sie "Serial RS232".



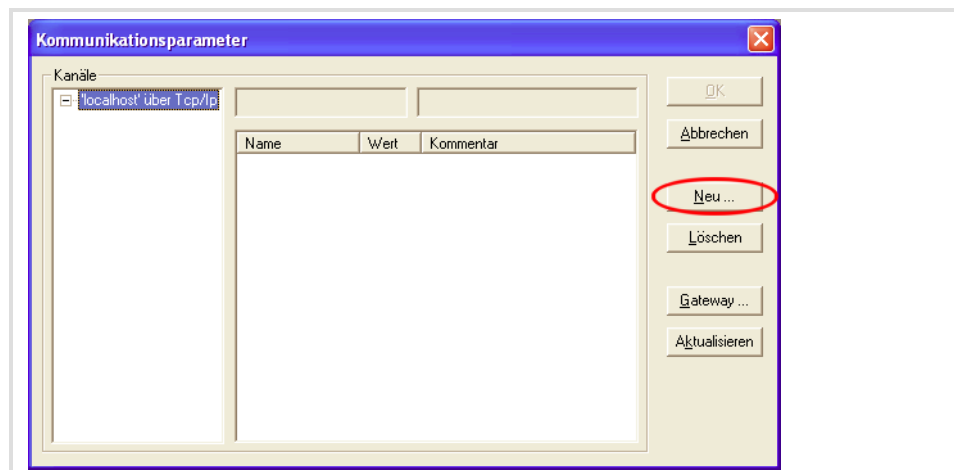
5. Klicken Sie auf **<OK>**.
6. Geben Sie die Parameter der Schnittstelle (z. B. COM1 mit Baudrate, Parity und Stopbit) vor, klicken Sie dazu auf das entsprechende Feld doppelt.



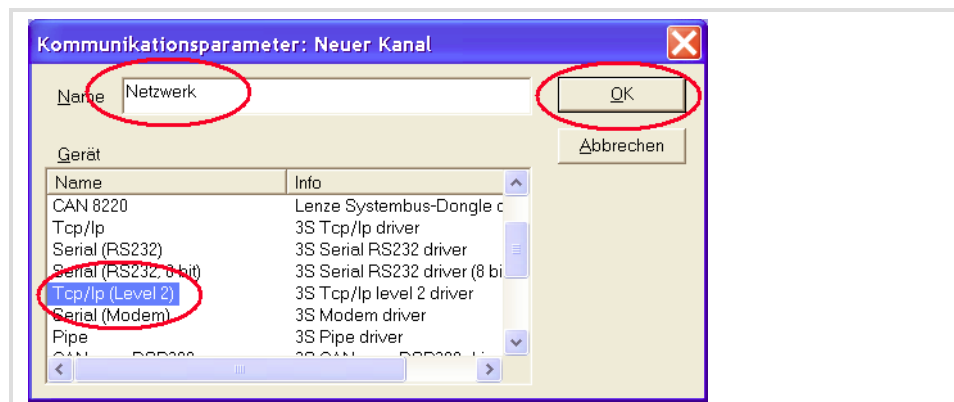
7. Da die Steuerung einen Motorola-Prozessor beinhaltet, muss das Feld "Motorola Byteorder" auf "Yes" stehen. Korrigieren Sie das ggf.
8. Schließen Sie das Fenster mit <OK>.

Verbindung über Netzwerk

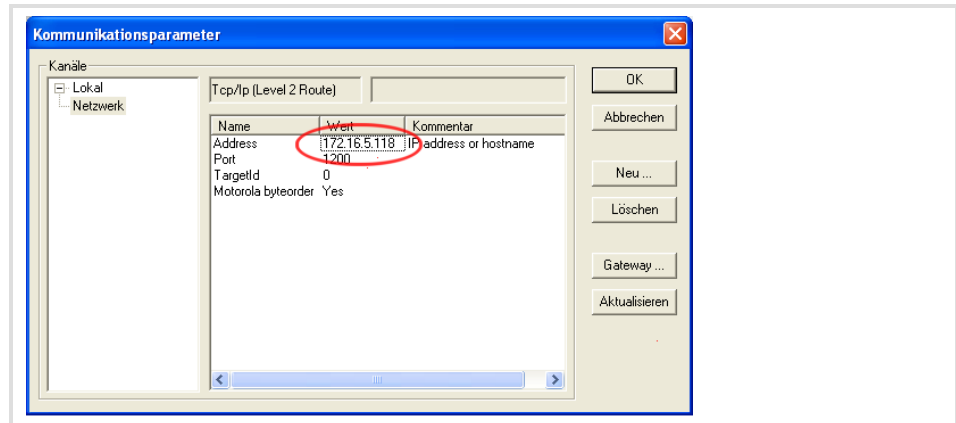
1. Wählen Sie **Online** → **Kommunikationsparameter**.
2. Klicken Sie auf <Neu>.



3. Geben Sie einen Namen vor (z. B. Netzwerk) und wählen Sie als Parameter das TCP/IP-Protokoll. Klicken Sie auf <OK>.



4. Klicken Sie doppelt auf den Wert des Feldes "Adress" und geben Sie die korrekte IP-Adresse vor (📖 17).

**Hinweis!**

Verwenden Sie in IP-Adressen keine führenden Nullen, da die IP-Adresse sonst als Oktadezimalzahl interpretiert wird.

5. Da die Steuerung einen Motorola-Prozessor beinhaltet, muss das Feld "Motorola Byteorder" auf "Yes" stehen. Korrigieren Sie das ggf. durch mehrfaches Doppelklicken.

**Hinweis!**

Im linken Fensterbereich werden die Verbindungsmöglichkeiten angezeigt. In diesem Fall die beiden Verbindungen "Lokal" und "Netzwerk".

Die aktuelle Einstellung ist die Einstellung, die vor Klicken auf <OK> im rechten Fenster angezeigt wird; sie wird für das Einloggen benutzt.

6. Schließen Sie das Fenster mit <OK>.

Programm laden und starten

1. Wählen Sie **Online → Einloggen**.

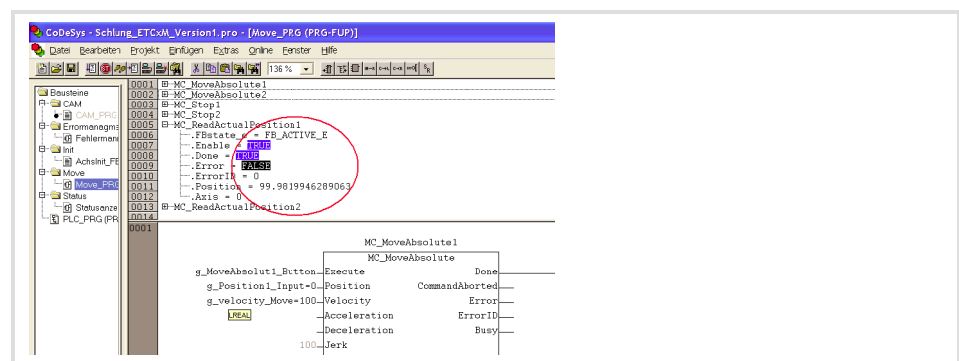
Das Programm wird in die Steuerung geladen.

2. Wählen Sie **Online → Start**.

Das Programm wird gestartet. Nach erfolgreichem Start wechselt in der Statuszeile das Feld "LÄUFT" seine Farbe von grau auf schwarz.



In der Statuszeile wird die aktuelle Verbindung (hier: "Lokal") angezeigt. Das Programm zeigt im rechten Teil des Fensters die aktuellen Werte der Variablen.



Programm bedienen

Nach Einloggen in die Steuerung über **Online → Einloggen**, können folgende Bedienfunktionen aktiviert werden:

- ▶ Programm stoppen: **Online → Stop**
- ▶ Datenspeicher auf Startwerte zurücksetzen: **Online → Reset**
- ▶ Boot-Projekt erstellen: **Online → Bootprojekt erzeugen**

Das Programm wird als "default.prg" in das Flash der Steuerung abgelegt. Beim Starten der Steuerung wird nach einem Programm "default.prg" gesucht. Dieses wird dann automatisch als Boot-Projekt gestartet.

- ▶ Boot-Projekt löschen: **Online → Reset (Ursprung)**
 - ▶ Kommunikation mit der Steuerung beenden: **Online → Ausloggen**
- Das Programm in der Steuerung behält den Status, der zuletzt ausgewählt wurde.

2.6 SPS-Programm testen

Das SPS-Programm wird über ETC-CoDeSys in die Steuerung geladen und gestartet. Über die CoDeSys-Visualisierung können die programmierten Funktionen angesteuert werden.

2.6.1 Visualisierungsfenster/Hauptfenster

Auf diesem Visualisierungsobjekt können folgende Funktionen vorgenommen werden:

- Initialisierung
- Freigabe der Achsen
- Absolute Bewegung der Achsen
- Fehlerinformation der Achsen

Bevor die Freigabe der Achsen erfolgreich durchgeführt werden kann, müssen die Achsen initialisiert werden. Dies geschieht über den Button **<axis init>**. Bei einem Fehler kann auf der rechten Seite der Fehlerinfo-Text angezeigt werden. Nach erfolgreicher Initialisierung können die Achsen freigegeben werden (**<axis1 powerOn>** und **<axis2 powerOn>**). Nach der Freigabe der Achsen besteht die Möglichkeit, die Achsen zu verfahren. In dem Beispiel ist ein absoluter Verfahrbefehl über den Button **<Start>** möglich.

The screenshot displays the 'Hauptfenster' (Main Window) of the ETC-CoDeSys visualization. It is divided into three main sections:

- initialisation:** Contains three buttons: 'axis init', 'axis1 powerON', and 'axis2 powerON'. Each button has a green indicator light and a numeric display showing '0'.
- axis-error-management:** Contains two input fields for 'axis1' and 'axis2' error information, each with an 'errorinfo' label and an 'ID' display showing '0'. A 'reset' button is located below these fields.
- move absolut:** Contains two rows for 'axis1' and 'axis2'. Each row has a 'start' button, a 'position' input field (set to '0'), and a unit indicator '[u]'. Below these rows are 'move-velocity' (set to '100') and 'stop-deceleration' (set to '1000') input fields, both with units '[u/s²]'. A 'stop' button is located at the bottom left of this section.

On the right side of the 'move absolut' section, there is an 'actual position' display showing the current positions: '1.2207 in [u]' for axis1 and '-7.625 in [u]' for axis2.

2.6.2 Visualisierungsobjekt Statusanzeige

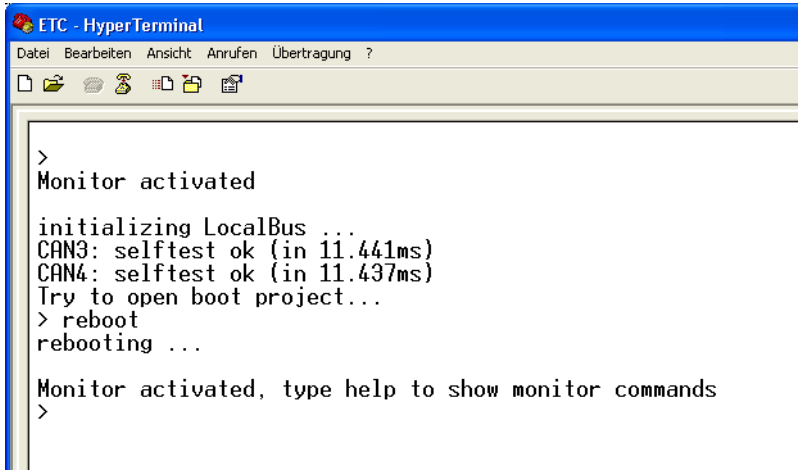
Zusätzlich zu den im Hauptfenster zur Verfügung stehenden Funktionen, wird in dem Visualisierungsobjekt "Statusanzeige" der Status der beiden Achsen angezeigt. Mit dem Betätigen von **<axis1 status>** und/oder **<axis2 status>** werden die Funktionen aktiviert.

status	
axis1 status	axis2 status
Error	Error
ErrorID	ErrorID
ErrorStop	ErrorStop
Stopping	Stopping
SysErrorstop	SysErrorstop
StandStill	StandStill
DiscreteMotion	DiscreteMotion
ContinuousMotion	ContinuousMotion
SynchronizedMotion	SynchronizedMotion
Homing	Homing
SuperImposedMotion	SuperImposedMotion
ConstantVelocity	ConstantVelocity
Accelerating	Accelerating
Decelerating	Decelerating

2.7 Firmware aktualisieren**2.7.1 Aufruf des Bootmonitors in der Steuerung**

Ausgangszustand: Die Kommunikation zwischen PC und ETC ist aufgebaut und die Monitorschnittstelle ist aktiviert (12)

1. Geben Sie im Terminal-Programm den Befehl **reboot** ein.
2. Halten Sie während des laufenden Reboot-Prozess die Tasten **<Shift> + <!>** gedrückt.



```
>
Monitor activated

initializing LocalBus ...
CAN3: selftest ok (in 11.441ms)
CAN4: selftest ok (in 11.437ms)
Try to open boot project...
> reboot
rebooting ...

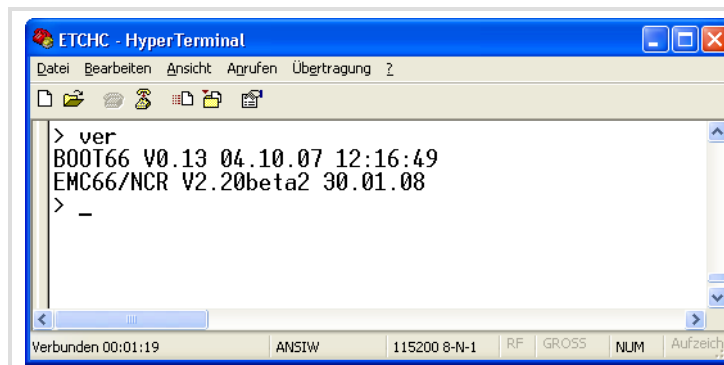
Monitor activated, type help to show monitor commands
>
```

Wenn der Prompt ">" erscheint, ist die Eingabe im Bootmonitor der Steuerung aktiv.

Auf der ETC leuchtet ein äußerer Balken der 7-Segment-Anzeige umlaufend.

2.7.2 Versionsabfrage der Firmware

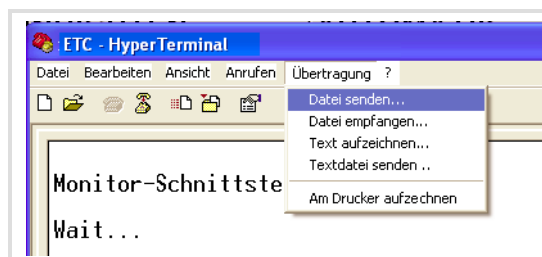
1. Wenn der Prompt ">" des Bootmonitors erscheint, geben Sie den Befehl **ver** ein.
Die Version der Firmware wird abgefragt und angezeigt.



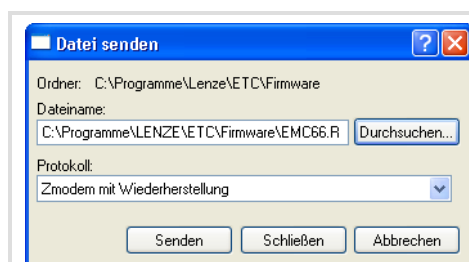
2.7.3 Updaten der Firmware

Ausgangszustand: Die Kommunikation zwischen PC und ETC ist aufgebaut und die Monitorschnittstelle ist aktiviert (12).

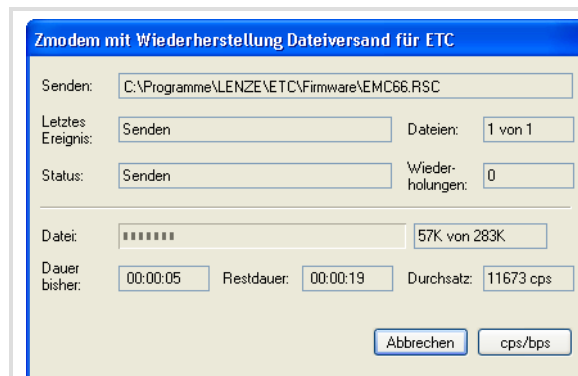
1. Um die aktuelle Firmware auf die ETC zu übertragen, aktivieren Sie im HyperTerminal den Menüpunkt **Übertragung → Datei senden**.



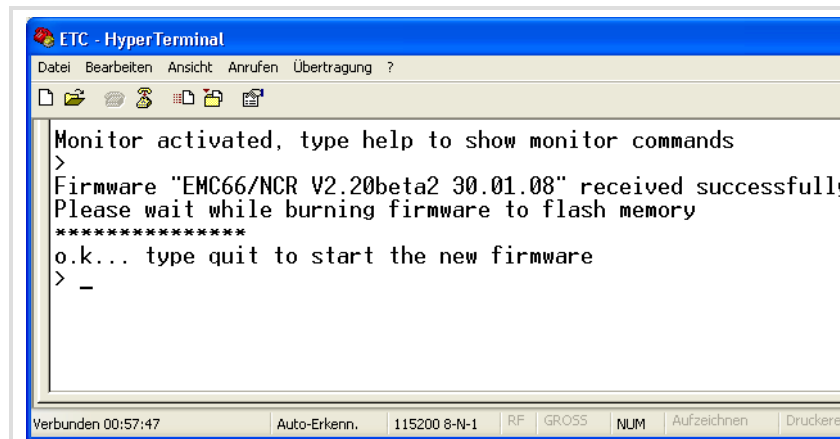
2. Geben Sie den Pfad zur Firmware-Datei vor und wählen Sie als Protokoll "Zmodem mit Wiederherstellung".
Firmware-Dateinamen: EMC55.rsc



3. Klicken Sie auf **Senden**.
Während des Ladens der Datei in die Steuerung erscheint das nachfolgende Bild. Auf der ETC-Frontplatte blinken die LEDs 1 ... 6 umlaufend.



Nach Abschluss des Ladevorgangs erscheint das folgende Bild:



Die Steuerung wartet auf das Laden der Firmware; auf der Frontplatte der ETC blinken die LEDs 1 ... 6 umlaufend.

4. Geben Sie im HyperTerminal-Fenster **quit** ein.

Die Firmware startet. Auf der ETC-Frontplatte leuchtet die grüne Watch-dog-LED.

Wichtige Befehle des Bootmonitors

Befehl	Bedeutung
ver	Zeigt die Versionen des Bootloaders und der aktuell geladenen Firmware an.
sz [dateiname]	Sendet die aktuell geladene Firmware unter dem angegebenen Namen per Z-Modem an den PC.
rz	Dieser Befehl wird implizit durch das angeschlossene Terminalprogramm bei Auswahl von "Dateien Senden" an die Steuerung geschickt und dient zur Übertragung von Firmware oder Bootloader in die Steuerung. Zum Senden muss das Protokoll "ZMODEM" angewählt werden.
quit	Beendet den Bootmonitor und startet die geladene Firmware.
reset	Setzt die Steuerung zurück.
help	Zeigt die Befehle des Bootmonitors an.

3 Grundlagen der Motion-Steuerung

3.1 Interpolator - Kern des Motion-Systems

Der Kern der Motion-Steuerung ist der Interpolator, der für die Ausführung von Bewegungsaufträgen zuständig ist, die mit Motion-Control-Funktionsbausteinen parametrisiert und angestoßen werden.

Ein achsunabhängiges Zustandsdiagramm des Interpolators ist in der folgenden Abbildung dargestellt.

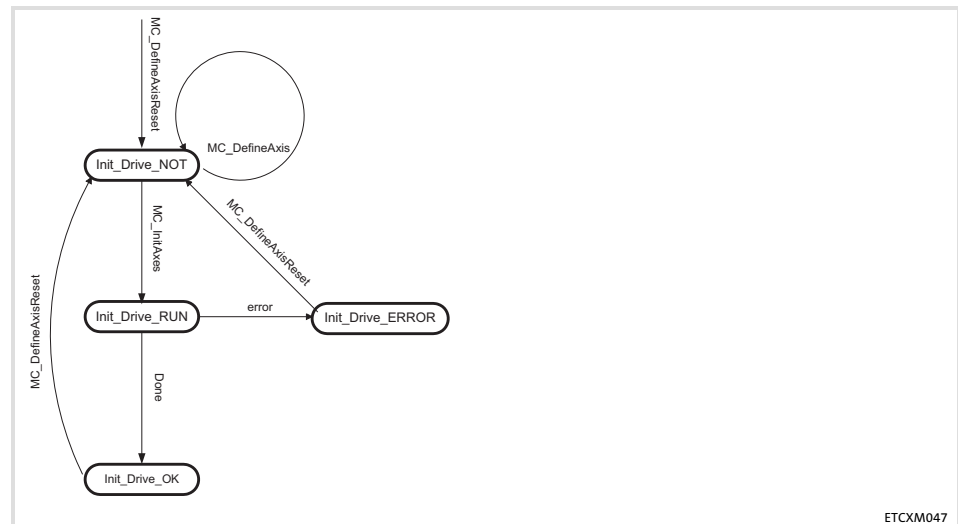


Abb. 3.1-1 Zustandsdiagramm des Interpolators

Der Interpolator kann sich in einem von vier Zuständen befinden:

Zustand	Bedeutung
Init_Drive_NOT	nach Reset der Steuerung
Init_Drive_RUN	Initialisierung läuft
Init_Drive_ERROR	Während der Initialisierung ist ein Fehler aufgetreten.
Init_Drive_OK	Der Interpolator ist betriebsbereit, d. h. die Bewegungsaufträge können ausgeführt werden.

Nach dem Reset der Steuerung befindet sich der Interpolator im Zustand **Init_Drive_NOT**. Dieser Zustand kann auch durch den Aufruf der Funktion **MC_DefineAxisReset** (☞ 84) erzwungen werden.

Die Aufgabe der SPS-Applikation ist, den Interpolator für jede Achse zu initialisieren: Mit dem Funktionsbaustein **MC_DefineAxis** (☞ 85) werden für jede Achse die Konfigurationsdaten an den Interpolator übergeben. Nachdem alle Achsen definiert sind, wird die Initialisierung des Drivehandlers mit der Funktion **MC_InitAxes** (☞ 88) bei dem Interpolator beauftragt.

Während der Initialisierung befindet sich der Interpolator im Zustand **Init_Drive_Run**.

Nach erfolgreicher Initialisierung geht der Interpolator in den Zustand **Init_Drive_OK** über. In diesem Zustand wird im Interpolator für jede definierte Achse eine Zustandsmaschine realisiert.

Wenn während der Initialisierung Fehler aufgetreten sind, geht der Interpolator in den Zustand **Init_Drive_ERROR**.

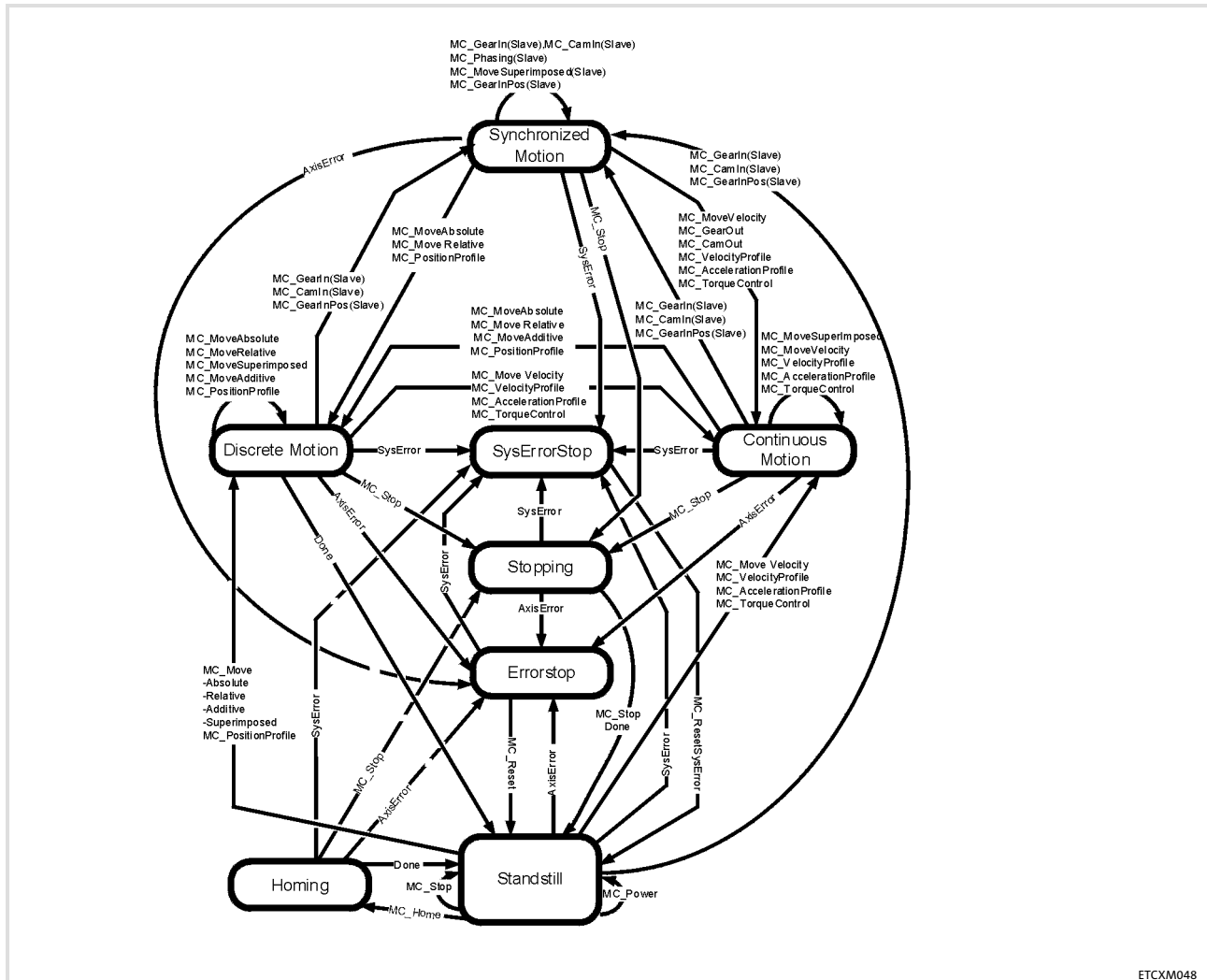


Abb. 3.1-2 Zustandsdiagramm einer Achse im Zustand INIT_DRIVE_OK des Interpolators

Jede Achse befindet sich in einem der folgenden Zustände:

Zustand	Bedeutung
Standstill	Keine Bewegung
Homing	Referenzpunktfahrt läuft
Discrete Motion	Bewegung auf Zielposition
Continuous Motion	Unendliches Fahren (jogging)
Synchronized Motion	Master/Slave synchronisierte Bewegung
Stopping	Achse wird angehalten
Errorstop	Achsfehler ist aufgetreten
SysErrorstop	Systemfehler ist aufgetreten

Ein Zustandswechsel wird durch eine andere Bewegungsart, Achsfehler, Systemfehler, Anhalten der Bewegung oder durch Fehlerquittieren (Reset) verursacht.



Hinweis!

AxisError bezieht sich auf einen Fehler, der von dem Drive-Handler der Steuerung oder von dem Antriebsrechner gemeldet wird.

SysError bezieht sich auf einen Systemfehler, der in der Steuerung generiert wird und achsunabhängig ist. Ein Systemfehler hat gleichzeitige Auswirkung auf alle Achsen.

3.2 Maschinenkonstanten - Steuerung konfigurieren

Die Steuerung wird über Maschinenkonstanten konfiguriert, die in Form einer ASCII-Datei in die Steuerung (ETCxM) geladen werden. Der Name der Datei ist beliebig, er muss lediglich die Endung **".mk"** aufweisen.

Jede Maschinenkonstante hat in der ***.mk**-Datei einen Bezeichner, der zur Identifikation dieser Maschinenkonstante in der Steuerung dient.

Nach dem Laden der Maschinenkonstanten wird die **PLC_PRG4**-Task sofort außerhalb des normalen Zyklus aufgerufen. Durch das Ereignis (Event) **SIG_MK_LOADED** wird die **PLC_PRG4**-Task informiert, dass neue Maschinenkonstanten geladen sind. Das Auslesen des Events kann mit dem Funktionsbaustein **MC_ReadEvent** geschehen.

Nur die Maschinenkonstanten **MK_LANGUAGE**, **MK_NETDISK_SRVADDR**, **MK_CAN_DRIVE_BAUDRATE**, **MK_TECHNOLOGY_DATA_1**, **MK_TECHNOLOGY_DATA_2**, **MK_TECHNOLOGY_DATA_3** und **MK_TECHNOLOGY_DATA_4** sind sofort nach dem Laden der Maschinenkonstanten in die Steuerung wirksam.

Die Maschinenkonstanten **MK_PLC_CODE_SIZE** und **MK_PLC_DATA_SIZE** werden erst nach einem Neustart der Steuerung wirksam (nur relevant bei Gerätegeneration 1).

Die restlichen Maschinenkonstanten sind nach erneuter Initialisierung (Definition) von Achsen wirksam; d. h.:

- ▶ alle Achsen in den Zustand **Standstill** versetzen (Bewegungen anhalten, anstehende Fehlermeldungen quittieren)
- ▶ mit **MC_DefineAxisReset** (📖 84) Interpolator zurücksetzen
- ▶ mit **MC_DefineAxis** (📖 85) alle Achsen definieren (dabei werden erst die geladenen Maschinenkonstanten an Interpolator übergeben)
- ▶ mit **MC_InitAxes** (📖 88) die Initialisierung vom Interpolator und Achshandler anstoßen.



Hinweis!

Bei der Gerätegeneration 2 muss die Maschinenkonstantendatei nicht vorhanden sein. Bei der Gerätegeneration 1 müssen lediglich MK_PLC_CODE_SIZE und/oder MK_PLC_DATA_SIZE in der Maschinenkonstantendatei enthalten sein, wenn die Defaultwerte geändert werden sollten.

Die Konfiguration kann vom SPS-Programm aus erfolgen.

Die allgemeinen (achsunabhängigen) Maschinenkonstanten können mit dem Funktionsbaustein MC_WriteMachineConstant geschrieben werden (82).

Die achsbezogenen Maschinenkonstanten können auch direkt im SPS-Programm festgelegt und an den Interpolator übergeben werden (85).

Alle Maschinenkonstanten werden in der Steuerung vorinitialisiert. Die Maschinenkonstanten-Datei sollte nur die Maschinenkonstanten beinhalten, deren Initialisierungswerte geändert werden.

Eine Maschinenkonstanten-Datei mit allen Maschinenkonstanten und ihren Initialisierungswerten wird in der nachfolgenden Tabelle gezeigt.

MK_TEST_WITHOUT_MECHANICS	0;
MK_PLC_CODE_SIZE	768;
MK_PLC_DATA_SIZE	768;
MK_CAN_DRIVE_NODE_ID	-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1;
MK_CAN_DRIVE_BAUDRATE	1000;
MK_AXIS_TYPE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
MK_LANGUAGE	0;
MK_REGLER_MODE	2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2;
MK_INTERPOL_CYCLE	4;
MK_INCREMENTS	65536,65536,65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536;
MK_DISTANCE	10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10;
MK_HOME_OFFSET	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
MK_SW_LIMIT_NEGATIVE	-10000,-10000,-10000,-10000,-10000,-10000,-10000,-10000,-10000,-10000,-10000,-10000,-10000,-10000,-10000,-10000;
MK_SW_LIMIT_POSITIVE	10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000;
MK_FILTER_TIME_CONST	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
MK_MAX_VELOCITY	1000,1000,1000,10000,10000,10000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000;
MK_MAX_ACCELERATION	10000,10000,10000,10000,10000,10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000;
MK_MAX_DECELERATION	10000,10000,10000,10000,10000,10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000;

MK_MAX_JERK	100000,100000,100000,100000,100000,100000,100000,100000,100000,100000,100000,100000,100000,100000;
MK_MAX_TORQUE	1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
MK_MAX_TORQUE_RAMP	10,10,10,10,10,10,10,10,10,10,10,10,10,10,10;
MK_TORQUE_TECH_UNITS_DENOM	1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
MK_TORQUE_TECH_UNITS_NUM	1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
MK_VELOCITY_TECH_UNITS_DENOM	1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
MK_VELOCITY_TECH_UNITS_NUM	1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
MK_TECHNOLOGY_DATA_1	0,0,0,0,0,0,0,0,0;
MK_TECHNOLOGY_DATA_2	0,0,0,0,0,0,0,0,0;
MK_TECHNOLOGY_DATA_3	0,0,0,0,0,0,0,0,0;
MK_TECHNOLOGY_DATA_4	0,0,0,0,0,0,0,0,0;
MK_NETDISK_SRVADDR	" " ;

3.2.1 Maschinenkonstanten-Übersicht

Name	Bedeutung																		
MK_TEST_WITHOUT_MECHANICS	Dient dem Test der Steuerungsfunktionen, ohne dass eine Maschine angeschlossen werden muss. Initialisierungswert: 0																		
MK_INTERPOL_CYCLE	Zykluszeit des Interpolators. Stellt den internen Interpolationstakt der Steuerung ein. In jedem Grobinterpolationstakt berechnet die Steuerung neue Positionswerte für die Achsen. Einheit: [ms] Initialisierungswert: 4 ms																		
MK_PLC_CODE_SIZE (nur Gerätegeneration 1)	Reserviert den Programmspeicher für die SPS im internen Arbeitsspeicher der Steuerung und legt damit gleichzeitig die maximale Größe des zu ladenden SPS-Programms in Kilobytes fest. Die Steuerung reserviert das Doppelte des in MK eingegebenen Speichers, um die Funktion Online-Change von CoDeSys zu unterstützen. Einheit: KB Initialisierungswert: 480 KB																		
MK_DATA_SIZE (nur Gerätegeneration 1)	Legt die Größe des Datenspeichers für die Variablen der SPS in Kilobytes fest. Einheit: KB Initialisierungswert: 128 KB																		
MK_LANGUAGE	Wählt eine Textdatenbank zur Anzeige von länderspezifischen Texten aus. Um eine Sprache auswählen zu können, müssen die jeweiligen Textdateien in das Flash-PROM der Steuerung geladen werden. Die Namen der Textdateien müssen lauten: ● NCRSPCHn.TXT für Fehlertexte ● SPSSPCHn.TXT für SPS-Texte Wobei "n" eine Ziffer von 0 ... 9 darstellt, die über die Maschinenkonstante ausgewählt wird. Initialisierungswert: 0																		
MK_NETDISK_SRVADDR	Legt eine Voreinstellung für die IP-Adresse des PC's fest, auf dem der Netdisk-Server läuft. Initialisierungswert: ":" Beispiel: MK_NETDISK_SRVADDR "172.16.5.111";																		
MK_CAN_DRIVE_NODE_ID	Antriebsadresse/CAN-Knotennummer 1. ETC-Gerätegeneration: 1 ... 12 2. ETC-Gerätegeneration: 1 ... 64, mit ● Knoten-Nr. 1 ... 16 für CAN 2 ● Knoten-Nr. 17 ... 32 für CAN 3 ● Knoten-Nr. 33 ... 48 für CAN 4 gelten. Initialisierungswert: -1 für alle Achsen Wenn eine Achse mit der Antriebsadresse mit MC_DefineAxis definiert wird, wird sie als virtuelle Achse betrachtet.																		
MK_AXIS_TYPE	Spezifiziert verschiedene Achseigenschaften. Die Maschinenkonstante ist Bit-codiert aufgebaut. <table><tr><th>Bit</th><th>Wert</th><th>Bedeutung</th></tr><tr><td rowspan="2">0</td><td>0</td><td>Linearachse</td></tr><tr><td>1</td><td>Rotationsachse</td></tr><tr><td rowspan="2">1</td><td>0</td><td>Endschalter aktiv</td></tr><tr><td>1</td><td>Endschalter nicht aktiv</td></tr><tr><td rowspan="2">3</td><td>0</td><td>Normale Achse</td></tr><tr><td>1</td><td>Geber</td></tr></table>	Bit	Wert	Bedeutung	0	0	Linearachse	1	Rotationsachse	1	0	Endschalter aktiv	1	Endschalter nicht aktiv	3	0	Normale Achse	1	Geber
Bit	Wert	Bedeutung																	
0	0	Linearachse																	
	1	Rotationsachse																	
1	0	Endschalter aktiv																	
	1	Endschalter nicht aktiv																	
3	0	Normale Achse																	
	1	Geber																	

Name	Bedeutung
MK_INCREMENTS	Bestimmt die Anzahl der Impulse, die dem Weg (MK_DISTANCE) in technischer Einheit [u] äquivalent sind. Einheit: keine Initialisierungswert: 65536 für alle Achsen
MK_DISTANCE	Bestimmt die Anzahl der technischen Einheiten [u], die der Anzahl der Impulse (MK_INCREMENTS) äquivalent sind. Positionen von Rotationsachsen werden modulo MK_DISTANCE berechnet. Einheit: [u] (Technische Einheit) Initialisierungswert: 10 für alle Achsen
MK_HOME_OFFSET	Der Versatz des mechanischen Nullpunktes der Maschine zum Nullpunkt des Wegmesssystems. Diese Nullpunktverschiebung wird nicht automatisch durch die Steuerung ausgeführt. (Es kann bei MC_Home benutzt werden.) Einheit: [u] Initialisierungswert: 0 für alle Achsen. Hinweis: Diese Maschinenkonstante kann mit MC_ReadParameter gelesen und für die Bestimmung des Eingangsparameters Position des Funktionsbausteins MC_Home eingesetzt werden.
MK_SW_LIMIT_NEGATIVE	Definiert die negative Verfahrbereichsgrenze der Mechanik. Einheit: [u] Initialisierungswert: -100000 für alle Achsen Software-Endlagen werden mit Parameter 5 aktiviert/deaktiviert
MK_SW_LIMIT_POSITIVE	Definiert die positive Verfahrbereichsgrenze der Mechanik. Einheit: [u] Initialisierungswert: 100000 für alle Achsen Software-Endlagen werden mit Parameter 4 aktiviert/deaktiviert
MK_MAX_VELOCITY	Die maximale Geschwindigkeit der einzelnen Achsen. Einheit: [u/s] Initialisierungswert: 100 für alle Achsen
MK_MAX_ACCELERATION	Die maximale Beschleunigung der einzelnen Achsen. Einheit: [u/s ²] Initialisierungswert: 1000 für alle Achsen
MK_MAX_DECELERATION	Die maximale Verzögerung der einzelnen Achsen. Einheit: [u/s ²] Initialisierungswert: 1000 für alle Achsen
MK_MAX_JERK	Der maximale Ruck der einzelnen Achsen. Einheit: [u/s ³] Initialisierungswert: 10000 für alle Achsen

Name	Bedeutung
MK_CAN_DRIVE_BAUDRATE	Baudrate für Drive-CAN-Bus/-Busse. Wert 0: Keine CNC-Open-Module angeschlossen Wert 10: Bitrate 10 kBit/s Wert 20: Bitrate 20 kBit/s Wert 50: Bitrate 50 kBit/s Wert 100: Bitrate 100 kBit/s Wert 125: Bitrate 125 kBit/s Wert 250: Bitrate 250 kBit/s Wert 500: Bitrate 50 kBit/s Wert 666: Bitrate 666,7 kBit/s Wert 800: Bitrate 800 kBit/s Wert 1000: Bitrate 1000 kBit/s Einheit: [kBits/s]. Initialisierungswert: 1000
MK_FILTER_TIME_CONST	Diese MK ist die Filterzeitkonstante für den Feininterpolationsfilter in Sekunden. Diese MK beeinflusst nicht direkt die Lageregelung, sondern die Feininterpolation, mit deren Hilfe neue Lagesollwerte im Feininterpolationsraster (MK_FINE_INTERPOL_CYCLE_NUMBER) berechnet werden. Der Feininterpolationsfilter ist ein Filter 2. Ordnung, mit dem die berechneten Lagesollwerte geglättet werden können, um ein sanfteres Fahrverhalten bei bestimmten Achsen zu erzielen. Diese MK wirkt auch auf Achsen mit digitaler Schnittstelle. Der Feininterpolationsfilter führt zu einer zeitlichen Verzögerung bei der Ausgabe der Lagesollwerte, die ca. 2mal so groß ist wie die Filterzeitkonstante. Daher sollte bei Achsen, die gemeinsam an einer Bahninterpolation beteiligt sind, grundsätzlich die gleiche Filterzeitkonstante eingetragen werden. Diese MK ist defaultmäßig auf 0 gesetzt und sollte nur vergrößert werden, wenn eine Achse zum Schwingen neigt. Sinnvolle Werte liegen zwischen 0 und 0.2 s. Je größer der eingestellte Wert ist, desto größer ist die durch den Filter hervorgerufene Bahnabweichung. Bei der Achsen von Geber-Typ (MK_AXIS_TYPE=8) werden die Istpositionen des Gebers gefiltert. Einheit: [s]. Initialisierungswert: 0 für alle Achsen
MK_TECHNOLOGY_DATA_1 MK_TECHNOLOGY_DATA_2 MK_TECHNOLOGY_DATA_3 MK_TECHNOLOGY_DATA_4	Freiverwendbare Maschinenkonstanten. Die MK's enthalten jeweils 10 Einträge. Der Wertebereich jedes Eintrags entspricht einem 64-bit Floating point-Wert (IEC61131-Datentyp LREAL). Die einzelnen Einträge können über die Funktion GetUserParam() von der SPS ausgelesen werden. Einheit: Benutzerabhängig. Initialisierungswert: 0 für alle Einträge
MK REGLER_MODE	Zur Zeit nur für interne Verwendung
MK_MAX_TORQUE	Maximales Drehmoment Einheit: [Nm] bei Rotationsmotoren oder [N] bei Linearmotoren Initialisierungswert: 1
MK_MAX_TORQUE_RAMP	Maximale Rampe (1. Ableitung; Drehmomente nach Zeit) Einheit: [Nm/s] bei Rotationsmotoren oder [N/s] bei Linearmotoren Initialisierungswert: 10

Name	Bedeutung
MK_TORQUE_TECH_UNITS_DENOM MK_TORQUE_TECH_UNITS_NUM	Nenner und Zähler für Umrechnungsfaktor von Drehmoment von Busdaten (Einheit im Achsmodul) in technische Einheiten (Einheit im Interpolator: Nm bei einem Drehmoment oder N bei einer Kraft). Initialisierungswert: 1 für Zähler und Nenner
MK_VELOCITY_TECH_UNITS_DENOM MK_VELOCITY_TECH_UNITS_NUM	Nenner und Zähler für Umrechnungsfaktor von Geschwindigkeit (Drehzahl) von Busdaten (Einheit im Achsmodul) in technische Einheiten (Einheit im Interpolator: u/s). Initialisierungswert: 1 für Zähler und Nenner
MK_ANALOG_DRIVE_CHANNEL (nur Gerätegeneration 2)	Die Nummer des LBM-ARI01-Kannels an dem die Analogachse angeschlossen ist. Die Kanäle werden von links nach rechts von 1 bis 12 ohne Lücken gezählt. Die maximale Anzahl von Analogachsen ist auf 12 begrenzt.
MK_FINE_INTERPOL_CYCLE_NUMBER (nur Gerätegeneration 2)	Die Anzahl der Feininterpolationstakte pro Grobinterpolationstakt (MK_INTERPOL_CYCLE). Damit wird die Dauer eines Lageregelungszyklus für Achsen mit analoger Schnittstelle festgelegt. Bei Achsen mit analoger Schnittstelle (konfiguriert mit MK_ANALOG_DRIVE_CHANNEL) werden in jedem Feininterpolationstakt die Istpositionen der Achsen eingelesen, die Differenz zu den Sollpositionen (Schleppabstand) ermittelt und neue Drehzahlstellgrößen berechnet. Bei Achsen mit digitaler Schnittstelle (MK_CAN_DRIVE_NODE_ID) erfolgt die Soll- und Istwertübergabe unabhängig von MK_FINE_INTERPOL_CYCLE_NUMBER einmal pro Grobinterpolationstakt.
MK_US (nur Gerätegeneration 2)	Überwachung des Signals "Weggeber gestört": 0 - Überwachung ausgeschaltet 1 - Überwachung eingeschaltet 2 - Überwachung eingeschaltet, einmalige Störung übergehen
MK_UA0 (nur Gerätegeneration 2)	Überwachung des Signals "Nullimpuls": 0 - Überwachung ausgeschaltet 1 - Überwachung eingeschaltet 2 - Überwachung eingeschaltet, einmalige Störung übergehen
MK_UA12 (nur Gerätegeneration 2)	Überwachung des Weggebersignals: 0 - Überwachung ausgeschaltet 1 - Überwachung eingeschaltet 2 - Überwachung eingeschaltet, einmalige Störung übergehen
MK_SSI_ENCODER (nur Gerätegeneration 2)	Dient bei Achsen mit analoger Schnittstelle zum Einstellen der Übertragungsparameter von Absolutwertgebern mit SSI-Schnittstelle. Mit dieser MK wird die Anzahl der zu übertragenden Bits eingestellt und zwischen Binär- oder Gray-Codierung umgeschaltet: Bit 0-6 mit Wert 12-32 - Anzahl der zu übertragenden Daten-Bits Bit 7 = 0 - Gray-Code Bit 7 = 1 - Binär-Code

Name	Bedeutung
MK_AXIS_DIGITAL_INPUTS (nur Gerätegeneration 2)	Ordnet die logischen Achsrechner-Eingänge: Endschalter plus, Endschalter minus, Referenznocken und Reserve den physikalischen digitalen LBM-ARI-Eingängen zu. Der Eingang Reserve ist für spezielle Verwendung applikationsspezifisch reserviert. Der einzugebende Wert ist BCD-codiert und benennt den logischen Achsrechnereingang 1-4, indem dessen Index an entsprechender Stelle in der MK eingetragen wird. ABCD Nummern der Eingänge, denen die entsprechende Funktion zugeordnet ist (N=1-4) A - positiver Endschalter B - negativer Endschalter C - Referenznocken D - Reservierung für diverse Anwendungen (z. B. für Touchprobe)
MK_MAX_POSITION_LAG (nur Gerätegeneration 2)	Die MK legt bei Achsen mit analoger Schnittstelle die Grenze in Incrementen fest, ab der die Achsrechnersoftware auf Schleppfehler erkennt. Der maximale Wert ist 32767 (voreingestellt).
MK_MAX_POSITION_LAG_COUNTER (nur Gerätegeneration 2)	Ist bei Achsen mit analoger Schnittstelle die Anzahl von unmittelbar aufeinander folgenden Reglerzyklen, in denen der maximale Schleppabstand überschritten werden muss, bis ein Schleppfehler gemeldet wird. Initialisierungswert: 1
MK_DRIFT_COMPENSATION (nur Gerätegeneration 2)	Kann bei Achsen mit analoger Schnittstelle verwendet werden, um den Offset am Sollwerteingang des Antriebsverstärkers zu kompensieren. Dies ist nur notwendig, falls der Antriebsverstärker selbst keine Möglichkeit bietet diesen Offset zu korrigieren. Die Offsetswerte werden für jede Achse vorzeichenbehaftet in Millivolt eingegeben. Initialisierungswert: 0
MK_QUICKSTOP (nur Gerätegeneration 2)	Bestimmt die Zeit, in der die Achse mit analoger Schnittstelle zum Stillstand gebracht wird, nach dem ein Achsfehler aufgetreten ist. Einheit: ms Initialisierungswert: 0

Name	Bedeutung		
MK_REGLERMODE (nur Gerätegeneration 2)	Bestimmt die Art des Lagereglers bei Achsen mit analoger Schnittstelle.		
	Wert	Bedeutung	Einzustellende MKs
	0	Reglerkreis geöffnet	keine
	2	P-Regler (default)	MK_KP
	3	P-Regler mit Geschwindigkeits- und Beschleunigungsvorsteuerung	MK_KP, MK_KF, MK_KB
	5	PID-Regler mit Geschwindigkeits- und Beschleunigungsvorsteuerung	MK_KP, MK_KF, MK_KB, MK_TV, MK_TN
Die Auswahl des Reglermodos hängt vom Anwendungsgebiet der Steuerung und von der Güte der Drehzahlregelung der angeschlossenen Achsen ab. Der reine P-Regler ist am einfachsten einzustellen, arbeitet aber mit dem größten Schleppabstand (Regelabweichung). Der P-Regler mit Geschwindigkeitsvorsteuerung (Feed-Forward) arbeitet bei konstanter Geschwindigkeit und guten Antrieben mit sehr kleinen Schleppabständen. Mit der Beschleunigungsvorsteuerung kann der Schleppabstand auch in der Beschleunigungs- und Bremsphase stark reduziert werden. Der PID-Regler ist vorzuziehen, wenn Antriebsverstärker ohne I-Anteil in der Drehzahlregelung eingesetzt werden. Er kann auch mit Antrieben ohne Drehzahlregelung, bei denen der Sollwert aus dem Lageregler direkt mit dem Stromregler verbunden ist, verwendet werden. Die überlagerte Geschwindigkeitsvorsteuerung spielt in beiden Fällen eine untergeordnete Bedeutung, da diese nur in Verbindung mit einem guten Drehzahlregler ihre volle Wirkung zeigt.			

Name	Bedeutung
MK_KP	MK_KP ist bei Achsen mit analoger Schnittstelle der Verstärkungsfaktor des P-Reglers, der in allen Reglertypen integriert ist. Er ist der Faktor, mit dem die Regeldifferenz in Inkrementen multipliziert wird. Das heißt, das bei dieser Einstellung auch MK_IM-PULSE und MK_WEG von Bedeutung sind. Bei Achsen mit SERCOS-Schnittstelle wird der Wert dieser MK im Parameter S-0-0104 (KV) des Antriebsverstärkers eingetragen. Für CAN-Achsen ist MK_KP ohne Bedeutung. MK_KP ist nicht der Kv-Faktor als Maß für die Verstärkung im Regelkreis der Maschine. Definitionen für Achsen mit analoger Schnittstelle: Gleichung 1: $K_v = \frac{V_{max}}{S_a}$ $V_{max} = \text{max. Geschwindigkeit der Achse [u/s]}$ $S_a = \text{Schleppabstand [u]}$ $u = \text{technische Einheit}$ Gleichung 2: $U_{max} = \text{Sollwert bei } V_{max} [V]$ $\{0.0 \dots 10.0 V\}$ Gleichung 3: $MK_KP = \frac{3276.7 \times U_{max} \times \frac{MK_DISTANCE}{MK_INCREMENTS}}{S_a}$ Gleichung 4: $S_a = \frac{V_{max}}{K_v} [u]$ Zur Kontrolle der Einstellung kann der Schleppabstand mit MC_ReadParameter (ParameterNumber=ACTUAL_POSITION_LAG) gelesen werden. Einstellen von Antriebsverstärkern mit analoger Schnittstelle: • Vorgabe einer Regelkreis-Verstärkung $K_v = 1$ • Berechnung von MK_KP nach (3) • Fahren der Achse mit G1 F(Vmax) • Justieren des Verstärkers, so dass sich der erforderliche Schleppabstand S_a einstellt. Die Achse kann auch mit niedrigerer Geschwindigkeit als V_{max} eingestellt werden. Es muss sich dann ein entsprechend kleinerer Schleppabstand einstellen. Damit ist sichergestellt, dass sich bei der Sollwertvorgabe U_{max} am Antrieb die Geschwindigkeit V_{max} einstellt und Gleichung (2) stimmt. Danach können Sie die dynamischen Parameter des Antriebsverstärkers, wie in Ihrem Antriebshandbuch beschrieben, optimieren. Optimieren Sie jetzt den Wert von MK_KP, um den Schleppabstand zu verkleinern und die Einschwingzeiten zu verkürzen. Vergewissern Sie sich dabei, dass die Achse auch bei einer stärkeren Anregung (z. B. Bremsen und Beschleunigen) nicht schwingt. Bei Achsen mit digitaler Schnittstelle erfolgt die Lageregelung und Schleppabstandsüberwachung im Antriebsverstärker. Die Beschreibung der notwendigen Parameter finden Sie in der Dokumentation Ihres Antriebsherstellers.

Name	Bedeutung
MK_KF (nur Gerätegeneration 2)	MK_KF ist der Bewertungsfaktor für die Geschwindigkeitsvorsteuerung (Feed-Forward) bei Achsen mit analoger Sollwertschnittstelle. Dieser kann mit nachfolgender Formel berechnet werden. Anschließend sollte der Wert in der Praxis überprüft und ggf. optimiert werden. Im besten Fall kann man damit den Schleppfehler auf Null reduzieren. Formel für MK_KF: (Umax ist hierbei der Sollwert für die Geschwindigkeit Vmax -> Gleichung 2 bei MK_KP-Beschreibung) $MK_KF = \frac{3276700 \times U_{max}}{V_{max} \frac{[u]}{[s]} \times \frac{MK_INCREMENTS}{MK_DISTANCE} [u] \times FIT [ms]}$ mit $FIT = \frac{MK_INTERPOL_CYCLE}{MK_FINE_INTERPOL_CYCLE_NUMBER}$ u - technische Einheit Die Einstellung der Achsen sollte nach folgendem Schema verlaufen (für jede Achse getrennt): • MK_REGLER_MODE auf P-Regler mit Geschwindigkeitsvorsteuerung (3) setzen. • MK_KP auf 1.0, MK_TV auf 0 und MK_TN auf 0 setzen. • Feed-Forward-Faktor nach o. g. Formel ausrechnen und in MK_KF eintragen. • Verfahren der Achse mit max. Geschwindigkeit und Kontrolle des Schleppabstandes mit Hilfe MC_ReadParameter(ParameterNumber= ACTUAL_POSITION_LAG). • Durch Verändern von MK_KF (geringfügig) den Schleppabstand gegen Null optimieren. • Optimieren von MK_KP, um den Schleppabstand in der Beschleunigungsphase zu minimieren. Zur reibungslosen Durchführung der o. g. Einstellung ist es notwendig, dass die Servoverstärker optimal eingestellt sind. Dies muss nach Anweisung des Herstellers erfolgen. Eine Kontrolle/Korrektur erfolgt durch Konfiguration und Einstellung der Achse zunächst als reiner P-Regler, wie zuvor beschrieben.
MK_KB (nur Gerätegeneration 2)	MK_KB ist der Bewertungsfaktor für die Beschleunigungsvorsteuerung für Achsen mit analoger Sollwertschnittstelle. Mit ihm kann der Schleppabstand in der Beschleunigungs- und Bremsphase reduziert werden. Diese MK ist nur wirksam, wenn ein Reglermode mit Geschwindigkeitsvorsteuerung eingestellt und in MK_KF ein Wert ungleich 0 eingetragen ist. Die Einstellung dieser MK sollte nach der Optimierung von MK_KF und MK_KP erfolgen. Sinnvolle Werte für diese MK liegen erfahrungsgemäß zwischen 0.0 und 2.0. Da die Beschleunigungsvorsteuerung nur in der kurzen Zeit einer Geschwindigkeitsänderung wirksam ist, sollte die Einstellung mit einer graphischen Achseinstellungssoftware wie z. B. AXSCOPE erfolgen.

Name	Bedeutung
MK_TV (nur Gerätegeneration 2)	MK_TV dient bei Achsen mit analoger Sollwertschnittstelle zum Einstellen der Vorhaltezeit [s] des PID-Reglers. Die angegebene Zeit ist proportional zum Differentialanteil des Reglers. Der D-Anteil dient dazu, bei trägen Regelstrecken die Schwingungsneigung zu unterdrücken, indem er jeder Änderung des Schleppabstands entgegen wirkt. Sinnvolle Werte für diese MK liegen im Bereich zwischen 0.0 und 0.05. Der eingegebene Wert ist zu groß gewählt, wenn die Achse schwingt oder beim Fahren laute Geräusche macht. Zur Einstellung wird am besten die graphische Achseinstellungssoftware AXSCOPE verwendet.
MK_TN (nur Gerätegeneration 2)	MK_TN ist bei Achsen mit analoger Sollwertschnittstelle die Nachstellzeit [s] des PID-Reglers. Die angegebene Zeit ist umgekehrt proportional zum Integral-Anteil des Reglers. Der I-Anteil dient dazu, den bleibenden Lagefehler bei Drehzahlreglern ohne Drehzahlrückführung oder ohne I-Anteil auszugleichen, indem er den Schleppabstand auf integriert und zum Drehzahlsollwert aus P- und D-Anteil addiert. Sinnvolle Werte für diese MK liegen im Bereich zwischen 1.0 und 0.05. Der eingegebene Wert ist zu klein gewählt, wenn die Achse schwingt. Bei einem Wert von 0 ist der I-Anteil ausgeschaltet. Zur Einstellung wird am besten die graphische Achseinstellungssoftware AXSCOPE verwendet.

Name	Bedeutung																
MK_HOMING_TYPE (nur Gerätegeneration 2)	MK_HOMING_TYPE legt durch den konfigurierten Wert die Art und Weise fest, wie die Achsen referenziert werden sollen. Diese MK wirkt nur auf Achsen mit analoger Schnittstelle. Bei CAN-Achsen muss die Art der Referenzpunktfahrt im Antriebsverstärker konfiguriert werden. <table> <tr> <th>Wert</th><th>Bedeutung</th></tr> <tr> <td>0</td><td>Normal Mode: Die Achse fährt in vorgegebener Richtung (Vorzeichen bei MK_HOMING_MAX_VEL_TO_CAM) bis der Referenznocken erkannt wird. Hierauf erfolgt Richtungsumkehr. Der Istwertzähler wird nach Wiederverlassen des Nockens mit dem nächsten Nullimpuls des Gebers genullt (default).</td></tr> <tr> <td>1</td><td>Wie 0, aber der Istwertzähler wird per Programm beim Verlassen des Nockens genullt.</td></tr> <tr> <td>2</td><td>Der Istwertzähler wird vom Programm direkt ohne Verfahrbewegung genullt. Nur bei Achsen mit analoger Schnittstelle möglich.</td></tr> <tr> <td>3</td><td>Nullen über nächsten Nullimpuls. Bei Achsen mit analoger Schnittstelle wird die Richtung der Nullimpulssuche zusätzlich durch den Zustand des Referenznockens bestimmt.</td></tr> <tr> <td>4</td><td>Der Reserveeingang des Achsrechners dient als zusätzlicher Nocken, der zuerst angefahren wird, bevor eine Referenzpunktfahrt nach Mode 0 ausgeführt wird. Nur bei Achsen mit analoger Schnittstelle möglich.</td></tr> <tr> <td>5</td><td>Referenzpunktfahrt für eine Achse von Typ Geber. Dabei wird nur auf den Nullimpuls gewartet. Die Achse wird nicht aktiv gefahren, sondern muss von Hand oder auf andere Weise bewegt werden. Nur bei Achsen mit analoger Schnittstelle möglich.</td></tr> <tr> <td>7</td><td>Wie Mode 0. Jedoch wird beim Erreichen des Referenznockens der Schleppabstand genullt. Dies ist notwendig, wenn der Referenznocken als Richtungsbegrenzung auf den Antriebsverstärker verdrahtet ist. Nur bei Achsen mit analoger Schnittstelle möglich.</td></tr> </table>	Wert	Bedeutung	0	Normal Mode: Die Achse fährt in vorgegebener Richtung (Vorzeichen bei MK_HOMING_MAX_VEL_TO_CAM) bis der Referenznocken erkannt wird. Hierauf erfolgt Richtungsumkehr. Der Istwertzähler wird nach Wiederverlassen des Nockens mit dem nächsten Nullimpuls des Gebers genullt (default).	1	Wie 0, aber der Istwertzähler wird per Programm beim Verlassen des Nockens genullt.	2	Der Istwertzähler wird vom Programm direkt ohne Verfahrbewegung genullt. Nur bei Achsen mit analoger Schnittstelle möglich.	3	Nullen über nächsten Nullimpuls. Bei Achsen mit analoger Schnittstelle wird die Richtung der Nullimpulssuche zusätzlich durch den Zustand des Referenznockens bestimmt.	4	Der Reserveeingang des Achsrechners dient als zusätzlicher Nocken, der zuerst angefahren wird, bevor eine Referenzpunktfahrt nach Mode 0 ausgeführt wird. Nur bei Achsen mit analoger Schnittstelle möglich.	5	Referenzpunktfahrt für eine Achse von Typ Geber. Dabei wird nur auf den Nullimpuls gewartet. Die Achse wird nicht aktiv gefahren, sondern muss von Hand oder auf andere Weise bewegt werden. Nur bei Achsen mit analoger Schnittstelle möglich.	7	Wie Mode 0. Jedoch wird beim Erreichen des Referenznockens der Schleppabstand genullt. Dies ist notwendig, wenn der Referenznocken als Richtungsbegrenzung auf den Antriebsverstärker verdrahtet ist. Nur bei Achsen mit analoger Schnittstelle möglich.
Wert	Bedeutung																
0	Normal Mode: Die Achse fährt in vorgegebener Richtung (Vorzeichen bei MK_HOMING_MAX_VEL_TO_CAM) bis der Referenznocken erkannt wird. Hierauf erfolgt Richtungsumkehr. Der Istwertzähler wird nach Wiederverlassen des Nockens mit dem nächsten Nullimpuls des Gebers genullt (default).																
1	Wie 0, aber der Istwertzähler wird per Programm beim Verlassen des Nockens genullt.																
2	Der Istwertzähler wird vom Programm direkt ohne Verfahrbewegung genullt. Nur bei Achsen mit analoger Schnittstelle möglich.																
3	Nullen über nächsten Nullimpuls. Bei Achsen mit analoger Schnittstelle wird die Richtung der Nullimpulssuche zusätzlich durch den Zustand des Referenznockens bestimmt.																
4	Der Reserveeingang des Achsrechners dient als zusätzlicher Nocken, der zuerst angefahren wird, bevor eine Referenzpunktfahrt nach Mode 0 ausgeführt wird. Nur bei Achsen mit analoger Schnittstelle möglich.																
5	Referenzpunktfahrt für eine Achse von Typ Geber. Dabei wird nur auf den Nullimpuls gewartet. Die Achse wird nicht aktiv gefahren, sondern muss von Hand oder auf andere Weise bewegt werden. Nur bei Achsen mit analoger Schnittstelle möglich.																
7	Wie Mode 0. Jedoch wird beim Erreichen des Referenznockens der Schleppabstand genullt. Dies ist notwendig, wenn der Referenznocken als Richtungsbegrenzung auf den Antriebsverstärker verdrahtet ist. Nur bei Achsen mit analoger Schnittstelle möglich.																
MK_HOMING_MAX_VEL_TO_CAM (nur Gerätegeneration 2)	MK_HOMING_MAX_VEL_TO_CAM bestimmt bei Achsen mit analoger Schnittstelle die Geschwindigkeit bei der Suche des Nockens.																
MK_HOMING_MAX_ACC_TO_CAM (nur Gerätegeneration 2)	MK_HOMING_MAX_ACC_TO_CAM bestimmt bei Achsen mit analoger Schnittstelle die Beschleunigung bei der Suche des Nockens.																
MK_HOMING_MAX_VEL_TO_ZERO (nur Gerätegeneration 2)	MK_HOMING_MAX_VEL_TO_ZERO bestimmt bei Achsen mit analoger Schnittstelle die Geschwindigkeit bei der Suche des Nullimpulses.																
MK_HOMING_MAX_ACC_TO_ZERO (nur Gerätegeneration 2)	MK_HOMING_MAX_ACC_TO_ZERO bestimmt bei Achsen mit analoger Schnittstelle die Beschleunigung bei der Suche des Nullimpulses.																

3.2.1.1 Maschinenkonstante MK_INTERPOL_CYCLE

MK_INTERPOL_CYCLE stellt den internen Grobinterpolationstakt in [ms] der Steuerung ein. In jedem Grobinterpolationstakt berechnet die Steuerung aus der programmierten Bahn neue Positionswerte für die beteiligten Achsen.

Je kleiner dieser Takt ist, desto dichter liegen die berechneten Positionen beieinander; desto höher ist jedoch auch die Rechenbelastung für die Steuerung und die Busbelastung für Antriebsverstärker mit digitaler Schnittstelle.

Bei Veränderungen dieser MK müssen in Abhängigkeit von Anzahl und Typ der angeschlossenen Antriebsverstärker zusätzlich Grenzen beachtet werden.

Die ETCHM der 1. Gerätegeneration kann mit einem minimalen Grobinterpolationstakt von 1 ms arbeiten.

Bei der ETCHM der 2. Gerätegeneration werden alle drei Drive-CAN-Bus parallel bearbeitet. Den minimalen Interpolationstakt bestimmt der Bus, an dem die meisten Achsen angeschlossen sind.

Bei Verstärkern mit CAN-Schnittstelle ist die begrenzte Bandbreite der CAN-Bus-Übertragung von entscheidender Bedeutung. Die folgende Tabelle zeigt den minimalen Grobinterpolationstakt und die mittlere CAN2-Buslast in Abhängigkeit von der Anzahl der konfigurierten CAN-Achsen und der eingestellten Bitrate.

Bei der Berechnung der Buslast wurden nur die zyklischen Telegramme berücksichtigt und nicht eventuell zusätzlich anfallende asynchrone Telegramme von MMI oder SPS. Der angegebene Wert kann durch unterschiedliche Telegrammlängen bei verschiedenen Antriebsverstärkern leicht variieren.

Achsenanzahl	1000 kBit	800 kBit	666 kBit	500 kBit	250 kBit
1	1 ms / 29%	1 ms / 36%	1 ms / 43%	2 ms / 29%	3 ms / 38%
2	1 ms / 52%	2 ms / 32%	2 ms / 39%	2 ms / 52%	4 ms / 52%
3	2 ms / 38%	2 ms / 47%	2 ms / 56%	3 ms / 50%	5 ms / 60%
4	2 ms / 49%	2 ms / 62%	3 ms / 49%	3 ms / 66%	6 ms / 66%
5	2 ms / 61%	3 ms / 51%	3 ms / 61%	4 ms / 61%	7 ms / 70%
6	2 ms / 73%	3 ms / 60%	3 ms / 73%	4 ms / 73%	8 ms / 73%
7	3 ms / 56%	3 ms / 70%	4 ms / 63%	5 ms / 67%	9 ms / 75%
8	3 ms / 64%	4 ms / 60%	4 ms / 72%	5 ms / 77%	10 ms / 77%
9	3 ms / 72%	4 ms / 67%	5 ms / 65%	6 ms / 72%	11 ms / 78%
10	3 ms / 79%	4 ms / 74%	5 ms / 72%	6 ms / 79%	12 ms / 79%
11	4 ms / 65%	4 ms / 82%	5 ms / 79%	7 ms / 75%	13 ms / 80%
12	4 ms / 71%	5 ms / 71%	6 ms / 71%	7 ms / 81%	14 ms / 82%
13	4 ms / 77 %	5 ms / 77 %	6 ms / 76 %	8 ms / 77 %	15 ms / 82 %
14	4 ms / 82 %	5 ms / 83 %	6 ms / 83 %	8 ms / 83 %	16 ms / 83 %
15	5 ms / 71 %	6 ms / 74 %	7 ms / 76 %	9 ms / 79 %	17 ms / 84 %
16	5 ms / 76 %	6 ms / 79 %	7 ms / 81 %	9 ms / 84 %	18 ms / 84 %

Bitte beachten Sie, dass die Bitraten 800 kBit und 666 kBit nicht von allen Antriebsherstellern unterstützt werden.

Wenn sichergestellt ist, dass im Betriebszustand keine asynchronen Telegramme von MMI oder SPS übertragen werden, kann die zur Verfügung stehende Übertragungsbandbreite besser ausgenutzt werden, s. folgende Tabelle (ohne Gewähr):

Achsenanzahl	1000 kBit	800 kBit	666 kBit	500 kBit	250 kBit
1	1 ms / 29%	1 ms / 36%	1 ms / 43%	1 ms / 57%	2 ms / 57%
2	1 ms / 52%	1 ms / 65%	1 ms / 78%	2 ms / 52%	3 ms / 69%
3	1 ms / 75%	1 ms / 94%	2 ms / 56%	2 ms / 75%	4 ms / 75%
4	2 ms / 49%	2 ms / 62%	2 ms / 74%	3 ms / 66%	5 ms / 79%
5	2 ms / 61%	2 ms / 76%	2 ms / 91%	3 ms / 81%	6 ms / 81%
6	2 ms / 73%	2 ms / 91%	3 ms / 73%	4 ms / 73%	7 ms / 83%
7	2 ms / 84%	3 ms / 70%	3 ms / 84%	4 ms / 84%	8 ms / 84%
8	2 ms / 96%	3 ms / 80%	3 ms / 96%	4 ms / 96%	8 ms / 96%
9	3 ms / 72%	3 ms / 90%	4 ms / 81%	5 ms / 86%	9 ms / 96%
10	3 ms / 79%	4 ms / 74%	4 ms / 89%	5 ms / 95%	10 ms / 95%
11	3 ms / 87%	4 ms / 82%	5 ms / 79%	6 ms / 87%	11 ms / 95%
12	3 ms / 95%	4 ms / 89%	5 ms / 86%	6 ms / 95%	12 ms / 95%
13	4 ms / 77 %	5 ms / 77 %	5 ms / 93 %	7 ms / 88 %	13 ms / 95 %
14	4 ms / 83 %	5 ms / 83 %	6 ms / 83 %	7 ms / 95 %	14 ms / 95 %
15	4 ms / 89 %	5 ms / 89 %	6 ms / 89 %	8 ms / 89 %	15 ms / 95 %
16	4 ms / 95 %	5 ms / 95 %	6 ms / 95 %	8 ms / 95 %	16 ms / 95 %

3.3 Funktionsbibliothek MC.lib - Bewegungsaufträge definieren

Die Motion Control-Bibliothek "MC.lib" beinhaltet verschiedene IEC-61131-3 konforme SPS-Funktionsbausteine, die das Verfahren von Achsen aus einem SPS-Programm ermöglichen.

Mit dem Einsatz dieser Bibliothek können komplexe Bewegungsführungen und einfache, bewegungsnahe SPS-Funktionen, wie das Verknüpfen von Eingangssignalen und das Berechnen und Setzen von Ausgangssignalen im selben System, realisiert werden.

Die ETC-Steuerung kann damit auch in Maschinen eingesetzt werden, bei denen die Steuerung von komplexen, schnellen und genauen Bewegungen eine entscheidende Rolle spielt.

Der Einsatz der SPS-Plattform hat den Vorteil, dass die SPS- und MC-Funktionalität in einer Programmierhand liegt. Damit können neben den üblichen SPS-Aktionen mehrere Bewegungen koordiniert, parallel oder überlagert ausgeführt werden.

Die Bibliothek beinhaltet Motion Control-Bausteine nach PLCopen-Standard: Teil 1 Version 1.1 und Teil 2 Version 1.0.

3.3.1 Funktionsbaustein-Übersicht

Ein Überblick über alle Funktionen und Funktionsbausteine der Motion-Control-Bibliothek zeigt die nachfolgende Tabelle.

Bausteine nach
PLCopen-Standard

Administrative	
Einzelachsen	Mehrachsen
MC_Power (📖 132)	MC_CamTableSelect (📖 152)
MC_Reset (📖 137)	
MC_ReadStatus (📖 133)	
MC_ReadAxisError (📖 135)	
MC_ReadParameter (📖 138)	
MC_ReadBoolParameter (📖 139)	
MC_WriteParameter (📖 140)	
MC_WriteBoolParameter (📖 141)	
MC_ReadActualPosition (📖 151)	
MC_TouchProbe (📖 120)	
MC_AbortTrigger (📖 123)	
MC_SetPosition (📖 145)	
MC_ReadDigitalInput (📖 187)	
MC_ReadDigitalOutput (📖 188)	
MC_WriteDigitalOutput (📖 189)	
MC_SetOverride (📖 142)	
MC_ReadActualVelocity (📖 148)	
MC_ReadActualTorque (📖 149)	
Motion	
Einzelachsen	Mehrachsen
MC_MoveAbsolute (📖 89)	MC_CamIn (📖 163)
MC_MoveRelative (📖 92)	MC_CamOut (📖 167)
MC_MoveAdditive (📖 95)	MC_GearIn (📖 169)
MC_MoveSuperImposed (📖 98)	MC_GearOut (📖 177)
MC_MoveVelocity (📖 101)	MC_Phasing (📖 179)
MC_Home (📖 104)	MC_GearInPos (📖 172)
MC_Stop (📖 105)	MC_AbortGearInPos (📖 176)
MC_PositionProfile (📖 108)	MC_ResetPhaseShift (📖 182)
MC_VelocityProfile (📖 112)	
MC_AccelerationProfile (📖 116)	
MC_TorqueControl (📖 125)	
MC_DigitalCamSwitch (📖 129)	

Weitere Bausteine

Administrative	
Einzelachsen	Mehrachsen
MC_DefineAxisReset (📖 84)	MC_CamLargeTableSelect (📖 159)
MC_DefineAxis (📖 85)	MC_CamTableDelete (📖 162)
MC_InitAxes (📖 88)	MC_CamReadSlavePosition (📖 168)
MC_ReadSysError (📖 202)	MC_CamExtTableSelect (📖 155)
MC_ResetSysError (📖 204)	
MC_ReadEvent (📖 205)	
MC_DefineQuickInputs (📖 183)	
MC_DefineQuickOutputs (📖 185)	
MC_AttachTaskToInt (📖 190)	
MC_CalcTimeToMove (📖 193)	
MC_CalcVelocityToMove (📖 195)	
MC_WriteMachineConstant (📖 82)	

3.3.2 Allgemeine Schnittstellen der Funktionsbausteine

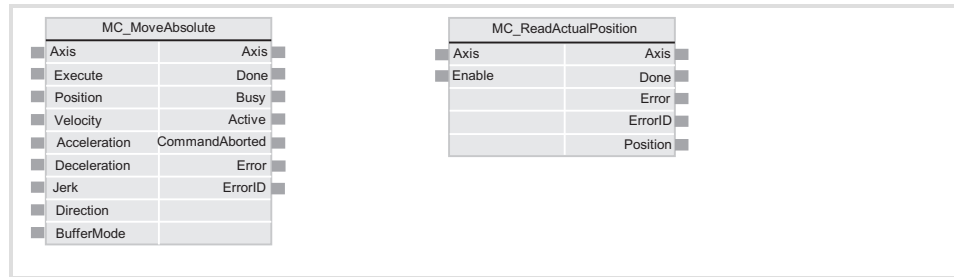


Abb. 3.3-1 Beispiele für Funktionsbausteine

Allgemeine Eingangsparameter

Alle Funktionsbausteine verfügen entweder über einen Eingangsparameter **Execute** oder **Enable**.

Mit steigender Flanke von **Execute** werden alle Werte der Eingangsparameter übernommen und die Ausführung des Bausteins gestartet. Der Zustand der Eingangsparameter muss mindestens für einen Bausteinaufruf anliegen.

Eine fallende Flanke von **Execute** setzt die Ausgänge zurück. Es ist garantiert, dass diese Ausgänge für mindestens einem Aufruf vom Funktionsbaustein gesetzt werden, wenn die entsprechende Situation auftritt. Das ist auch dann der Fall, wenn **Execute** früher auf FALSE gesetzt wird.

Funktionsbausteine, die Informationen zyklisch über einen Ausgangsparameter "liefern", verfügen über den Eingangsparameter **Enable**. Solange der Eingangsparameter **Enable** = TRUE ist, wird der entsprechende Ausgangsparameter mit dem aktuellen Wert versorgt.

Alle Parameter, die in den Parametertabellen von Funktionsbausteinen mit "P" (Pflichtparameter) bezeichnet sind, müssen vom Anwender versorgt werden. Alle Parameter, die in den Parametertabellen von Funktionsbausteinen mit "O" (optional) bezeichnet sind, sind mit Defaultwerten vorinitialisiert. Erfordert eine Anwendung andere Werte, so müssen die Parameter beim Aufruf geändert werden. Einmal übergebene Parameterwerte bleiben erhalten, wobei die neuen Werte erst mit steigender Flanke von **Execute** übernommen werden.

Einige Funktionsbausteine haben den Eingangsparameter **BufferMode** mit dem der Übergang zwischen zwei Bewegungen bestimmt werden kann:

- ▶ Wenn die Bewegung nicht gepuffert werden soll, wird die laufende Bewegung sofort abgebrochen und die neue gestartet.
- ▶ Wenn die Bewegung gepuffert werden soll, wird die laufende Bewegung solange fortgesetzt, bis der Zustand **Done**, **InSync**, **InVelocity** usw. erreicht wird. Wenn während einer laufenden Aufgabe ein Fehler gemeldet wird, wird die im Puffer wartende Aufgabe gelöscht.
- ▶ Wenn eine neue Aufgabe mit steigender Flanke am **Execute** gestartet wird, wird die im Puffer auf Ausführung wartende Aufgabe überschrieben und der für überschriebene Aufgaben zuständige Funktionsbaustein meldet das mit **CommandAborted=TRUE**.

Die folgende Tabelle gibt einen Überblick über die Eigenschaften des Eingangsparameters **BufferMode** bei verschiedenen Funktionsbausteinen:

Funktionsblock	Kann gepuffert werden	Kann durch einen gepufferten FB gefolgt werden	Relevantes Signal für Aktivieren eines gepufferten FB
MC_MoveAbsolute (📖 89)	Ja	Ja	Done
MC_MoveRelative (📖 92)	Ja	Ja	Done
MC_MoveAdditive (📖 95)	Ja	Ja	Done
MC_MoveSuperImposed (📖 98)	Ja	Ja	Done: bei MC_MoveSuperImposed (verfügt über eigenen Puffer)
MC_MoveVelocity (📖 101)	Ja	Ja	InVelocity
MC_Home (📖 104)	Nein	Nein	
MC_Power (📖 132)	Nein	Nein	
MC_Stop (📖 105)	Nein	Nein	
MC_PositionProfile (📖 108)	Ja	Ja	Done
MC_VelocityProfile (📖 112)	Ja	Ja	Done
MC_AccelerationProfile (📖 116)	Ja	Ja	Done
MC_CamIn (📖 163)	Ja	Ja	EndOfProfile
MC_CamOut (📖 167)	Nein	Ja	Done
MC_GearIn (📖 169)	Ja	Ja	InGear
MC_GearOut (📖 177)	Nein	Ja	Done
MC_Phasing (📖 179)	Ja	Ja	Done: bei MC_Phasing (verfügt über eigenen Puffer)
MC_TorqueControl (📖 125)	Ja	Ja	InTorque

Allgemeine
Ausgangsparameter

Parameter **Done** = TRUE, wenn der Befehl erfolgreich beendet, z. B. Zielposition erreicht wird.

Parameter **CommandAborted** = TRUE, wenn der Befehl durch einen anderen Bewegungsbefehl abgebrochen wird.

Der Ausgangsparameter **Busy** wird mit steigender Flanke von dem Eingangsparameter **Execute** gesetzt und zurückgesetzt, wenn einer von den Ausgangsparametern **Done**, **CommandAborted** oder **Error** gesetzt wird.

Den Ausgangsparameter **Active** haben die Funktionsbausteine, die auch den Eingangsparameter **BufferMode** haben. Der Ausgang wird gesetzt, wenn der jeweilige Funktionsbaustein die Bewegungskontrolle über entsprechender Achse übernimmt.

Bei einer nicht gepufferten Aufgabe haben die Ausgangsparameter **Busy** und **Active** den gleichen Wert.

Die Ausgänge **Done**, **Busy**, **Active**, **InVelocity**, **InGear** und **InSync** sind logisch exklusiv zum Ausgang **Error**; d. h. bei **Error** = TRUE werden diese zurückgesetzt.

Mit einer Ausnahme: Falls bei aktiver Bewegung neue Eingangsparameter mit der gleichen FB-Instanz übergeben werden und diese Eingangsparameter falsch sind, wird dieser Fehler über die Ausgangsparameter **Error** und **ErrorID** bis zum Ende der laufenden Bewegung oder bis zu einer fallenden Flanke des Eingangsparameter **Execute** angezeigt.

Somit werden die Ausgangsparameter **Done**, **Busy**, **Active**, **CommandAborted**, **inVelocity**, **inGear** und **inSync** zusammen mit **Error** und **ErrorID** angezeigt.

Über den Ausgang **Error** werden folgende Fehler gemeldet:

- ▶ Instanzfehler beim Aktivieren eines Funktionsbausteins. Dieses bedeutet, dass die Parameterwerte außerhalb des zulässigen Bereichs sind oder die Achse befindet sich in einem für diesen Befehl unerlaubten Zustand.
- ▶ Achsfehler (AxisError), wenn eine Achse während der Ausführung des Befehls einen Fehler gemeldet hat.
- ▶ Systemfehler, wenn während der Ausführung des Befehls ein Systemfehler aufgetreten ist.
- ▶ Der Ausgangsparameter **ErrorID** zeigt die aktuelle Fehlernummer, wenn **Error** = TRUE.

Position oder Distanz

Position ist ein absoluter Wert in einem Koordinatensystem. **Distanz** ist ein relativer Wert und als Differenz zwischen zwei Positionen zu verstehen ist.

Vorzeichenregel

Velocity, **Acceleration**, **Deceleration** und **Jerk** haben immer positive Werte. **Position** und **Distance** können positiv oder negativ sein.

3.3.3 Datentyp **AXIS_REF**

Der Datentyp **AXIS_REF** wird folgendermaßen definiert:

```
TYPE
  AXIS_REF : WORD;
END_TYPE
```

Damit ist jede Achse in einem SPS-Programm über ihre Nummer (Index) identifizierbar.

Die erlaubte Axisnummer: 0 ... MaxAxis; wobei

MaxAxis=15 bei der ETCHM 1. Gerätegeneration

MaxAxis=59 bei der ETCHM 2. Gerätegeneration

Nachfolgend zwei Beispiele, wie die Variablen für den Eingang **AXIS** in einer Applikation deklariert werden können:

Beispiel 1

```
VAR
  AXIS_REF :          ARRAY[0..11] OF AXIS_REF:=0,1,2,3,4,5,6,7,8,9,10,11;
  MCMoveAbsoluteX:    MC_MoveAbsolute;
  MCMoveAbsoluteX:    MC_MoveAbsolute;
END_VAR
MCMoveAbsolute[0](Axis:=Axis[0],Execute:=FALSE);
MCMoveAbsolute[1](Axis:=Axis[1],Execute:=FALSE);
```

Beispiel 2

```
VAR
  Axis0:              WORD:=0;
  Axis1:              WORD:=1;
  MCMoveAbsolute:     ARRAY[0..1] OF MC_MoveAbsolute;
END_VAR
MCMoveAbsolute[0](Axis:=Axis0,Execute:=FALSE);
MCMoveAbsolute[1](Axis:=Axis1,Execute:=FALSE);
```

3.3.4 Bewegungen mehrerer Achsen synchronisieren

Eine synchronisierte Bewegung von zwei oder mehreren Achsen kann über ein elektronisches Getriebe oder über eine so genannte Kurvenscheibentabelle (CAM-Tabelle) durchgeführt werden.

Das elektronische Getriebe ist eine lineare Kopplung mit einem festen Übersetzungsverhältnis.

Eine komplexe analytisch nicht beschreibbare Kopplung von Master und Slave lässt sich oft nur in Form einer Tabelle darstellen, in der zu jeder Master-Position eine zugehörige Slaveposition festgehalten wird.

Über die Funktionsbausteine **MC_GearIn** (169), **MC_GearInPos** (172) und **MC_GearOut** (177) lässt sich ein elektronisches Getriebe realisieren.

Die Funktionsbausteine **MC_CamTableSelect** (152), **MC_CamIn** (163) und **MC_CamOut** (163) bieten die Möglichkeit, eine Slave-Achse über eine Tabelle an eine Master-Achse zu koppeln. Die Master-Achse kann eine virtuelle Achse sein.

Eigenschaften CAM-Tabelle

- ▶ Die Tabellen enthalten die Daten im Binärformat.
- ▶ Die Positionen in der Tabelle sind einheitenneutral. Erst nach der Zuordnung der Kurvenscheibentabelle zu einer Master- und Slave-Achse werden die jeweiligen Einheiten der Achse übernommen. Die Einheit einer Achse wird indirekt mit den Konfigurationsdaten **MK_DISTANCE** und **MK_INCREMENTS** oder mit der Struktur **AXIS_CONFIG_CONST_TR** (**lrRoad** und **lrImpulse**) vorgegeben.
- ▶ Die Tabelle muss mit der Masterposition 0.0 beginnen.
- ▶ Masterpositionen müssen streng monoton wachsend oder fallend sein.
- ▶ Die Masterpositionen in der Tabelle können in nicht äquidistanten Schritten vorliegen.
- ▶ Die Tabelle kann absolute oder relative (im Bezug auf die erste Tabellenposition) Slave-Positionen beinhalten.
- ▶ Die Master- und Slave-Positionen können mit jeweils einem Master- oder Slaveoffset verschoben und mit einem Master- und Slave-Skalierungsfaktor (Steigung) multipliziert werden.
- ▶ Der Masteroffset wird vor allen anderen Operationen auf die Masterposition aufaddiert.
- ▶ Der Slaveoffset wird nach allen anderen Operationen auf die Slaveposition aufaddiert.
- ▶ Eine Tabelle **T** enthält **N** Zeilen und pro Zeile 3 Daten, die Masterposition **T[k][0]**, die zugehörige Slaveposition **T[k][1]** und die Interpolationsart **T[k][2]**, wobei: $0 < k < N$.
- ▶ Eine Tabelle kann linear oder zyklisch (periodisch) sein. Eine lineare Tabelle definiert einen begrenzten Verfahrensweg zwischen zwei Master-Positionen, der durch die Differenz zwischen dem letzten und ersten Tabelleneintrag festgelegt ist. Führt die Master-Achse aus dem definierten Bereich heraus, so fährt die Slave-Achse weiter im Zustand **Continuous Motion** mit aktueller Geschwindigkeit.
- ▶ Eine zyklische Tabelle definiert einen periodisch wiederkehrenden Streckenabschnitt. Führt die Master-Achse aus diesem Streckenabschnitt heraus, so wird die Tabelle wieder von Anfang an interpretiert. Dadurch kann sich ein periodischer Verlauf der Slave-Achse ergeben. Am Ende der Master-Periode wird auf die Position der Slave-Achse die Slave-Periode aufaddiert, so dass ein Übergang vom Tabellenende auf den Tabellenanfang ohne Positionssprung der Slave-Achse gewährleistet ist.

Zyklische Tabellen auswerten

Zyklische Tabellen sind durch die folgenden zusätzlichen Parameter charakterisiert:

► Master-Periode:

- Die Tabelle erstreckt sich von $T[0][0] = 0.0$ bis $T[N-1][0] =$ Master-Periode.

► Slave-Periode:

- $T[N-1][1] - T[0][1] =$ Slave-Periode (Slave-Differenz pro Master-Periode).

Die Tabelle verfügt über einen Zykluszähler, der bei Tabellenüberlauf inkrementiert und bei Tabellenunterlauf dekrementiert wird.

Bei Tabellenüberlauf/Tabellenunterlauf wird die Slavedifferenz pro Masterperiode auf die Slaveposition aufaddiert bzw. von der Slaveposition subtrahiert.

Slaveposition berechnen

Bezeichnung	Bedeutung
P_M	absolute Masterposition
P_{M0}	absolute Startposition der Master-Achse
P_{PH}	Phasenverschiebung der Masterposition
P_S	absolute Slaveposition
P_{S0}	absolute Startposition der Slave-Achse
P_{MC}	Master-Koppelposition
P_{SC}	Slave-Koppelposition
O_M	Master-Offset
O_S	Slave-Offset
S_M	Master-Skalierungsfaktor
S_S	Slave-Skalierungsfaktor
Z_{MP}	Master-Periode (Zyklus-Periode)
Z_{SP}	Slave-Periode

Der Anfang einer Tabelle wird auf das maximale Vielfache der Masterperiode, das kleiner als die aktuelle Position der Master-Achse ist, gesetzt.

1. Startmode: Slave absolut:

$$P_{MC} = P_{M0} - \text{MOD}(P_{M0}, Z_{MP})$$

a.) eine lineare Tabelle

$$P_S = O_S + S_S * T[O_M + S_M * [P_M + P_{PH} - P_{MC}]]$$

b.) eine zyklische Tabelle

$$P_S = O_S + S_S * T[\text{MOD}(O_M + S_M * [P_M + P_{PH} - P_{MC}], Z_{MP})]$$

2. Startmode: Slave relativ (die Tabelle fängt in der Slaveposition mit 0.0 an):

$$P_{SC} = P_{S0} - \text{MOD}(P_{S0}, Z_{SP})$$

a.) eine lineare Tabelle

$$P_S = P_{SC} + O_S + S_S * T[O_M + S_M * [P_M + P_{PH} - P_{MC}]]$$

b.) eine zyklische Tabelle

$$P_S = P_{SC} + O_S + S_S * T[\text{MOD}(O_M + S_M * [P_M + P_{PH} - P_{MC}], Z_{MP})]$$

Ablauf

- ▶ **Ankopplung im Standstill der Master-Achse**
 - Verfahren der Achsen auf Koppelpositionen
 - Starten der Masterachse

Wenn die Differenz zwischen der aktuellen Slaveposition und der Slaveposition aus der Tabelle größer als der maximale Ruck (**MK_MAX_JERK**) der Achse ist, wird entweder Fehler gemeldet oder die Differenz auf die Sollposition der überlagerten Bewegung aufaddiert (entsprechende Einstellung mit dem Eingangsparameter **CouplingMode** von MC_CamIn). Auf diese Weise wird ein zusätzlicher Offset für die Slave-Achse gebildet, der während der aktiven Kopplung über Funktionsbaustein **MC_MoveSuperimposed** dynamisch abgebaut werden kann. Beim Startmode **Ramp in** wird zunächst die Slave-Achse über einen vorgegebenen Master-Weg auf die entsprechende Slaveposition aus der Tabelle gefahren.
 - Evtl. Online Änderungen der Offsets, Steigung.
 - Evtl. Online Korrekturen der Tabelle oder Tabellenwechsel.
 - Abkoppeln (**MC_CamOut**). Die Slave-Achse fährt weiter mit der letzten Sollgeschwindigkeit.
- ▶ **Ankopplung bei fahrender Master-Achse**
 - Ankoppeln (**MC_CamIn**)

Wenn die Differenz zwischen der aktuellen Slaveposition und der Slavekoppelposition aus der Tabelle größer als der maximale Ruck (**MK_MAX_JERK**) ist, wird entweder Fehler gemeldet oder die Differenz auf die Sollposition der überlagerten Bewegung aufaddiert (entsprechende Einstellung mit dem Eingangsparameter **CouplingMode** bei MC_CamIn). Auf diese Weise wird ein zusätzliches Offset für die Slave-Achse gebildet, der während der aktiven Kopplung mit Funktionsbaustein MC_MoveSuperimposed dynamisch abgebaut werden kann. Beim Startmod **RampIn** wird zunächst die Slave-Achse auf die entsprechende Slaveposition aus der Tabelle gefahren. Eine Rotationsachse hat beim Startmod **RampIn** die Möglichkeit auf die Slaveposition aus der Tabelle entweder über den kürzesten Weg oder in die positive oder in die negative Richtung zu fahren.
 - Evtl. Online Änderungen der Offsets, Steigung
 - Evtl. Online Korrekturen der Tabelle oder Tabellenwechsel
 - Abkoppeln (**MC_CamOut**). Die Slave-Achse fährt weiter mit der letzten Sollgeschwindigkeit.

3.3.5 Schnelle Ein-/Ausgänge für Interpolator

Bis zu 32 digitale Eingänge sowie bis zu 32 digitale Ausgänge können dem Interpolator als schnelle Ein-/Ausgänge von der SPS aus freigegeben werden. Die SPS kann den Zustand der freigegebenen Ausgänge nicht mehr verändern, bis die Freigabe zurückgesetzt wird. Die freigegebenen Eingänge können direkt im Interpolator abgefragt werden.

- ▶ Digitale Eingänge werden mit dem Funktionsbaustein **MC_DefineQuickInputs** (📖 183) freigegeben.
- ▶ Digitale Ausgänge werden mit dem Funktionsbaustein **MC_DefineQuickOutputs** (📖 185) freigegeben.

Schnelle Ein- und Ausgänge werden von den Funktionsbausteinen **MC_TouchProbe** (📖 120) und **MC_DigitalCamSwitch** (📖 129) verwendet. Ein Zugriff erfolgt nur über folgende Funktionsbausteine, wobei das Setzen und Zurücksetzen eines schnellen Ausganges durch den Interpolator ausgeführt wird:

- ▶ **MC_ReadDigitalInput** (📖 187)
- ▶ **MC_ReadDigitalOutput** (📖 188)
- ▶ **MC_WriteDigitalOutput** (📖 189)

3.3.6 Software-Endschalter

Die Softwareendschalter können mit den Parametern **ENABLE_LIMIT_POS** und/oder **ENABLE_LIMIT_NEG** für jede Achse aktiviert und deaktiviert werden, indem diese Parameter mit dem Funktionsbaustein **MC_WriteBoolParameter** (📖 141) auf **TRUE** (aktivieren) bzw. auf **FALSE** (deaktivieren) gesetzt werden.

Die entsprechenden Bereichsgrenzen können in die Parameter **SW_LIMIT_POS** und **SW_LIMIT_NEG** mit dem Funktionsbaustein **MC_WriteParameter** (📖 140) geschrieben werden. Die beiden Parameter werden mit den konfigurierten Werten (**MC_DefineAxis** oder in der MK-Datei) vorinitialisiert.

Die durch den Interpolator erzeugten Sollpositionen werden mit aktiven Softwareendschaltern überwacht. Wenn die Softwareendschalter überschritten werden, wird die jeweilige Achse mit den konfigurierten maximalen Werten von Verzögerungsrampe und Ruck angehalten. Dabei wird davon ausgegangen, dass bei der Konfiguration der Softwareendschalter ein entsprechender Bremsweg vorgesehen wurde, der verhindert, dass die Achsen an einen mechanischen Anschlag fahren.

Die Funktionsbausteine **MC_MoveAbsolute**, **MC_MoveAdditive**, **MC_MoveRelative** und **MC_MoveSuperImposed**, bei denen der programmierte Weg zu einer Position hinter den Softwareendschalter führen könnte, werden nicht ausgeführt. Eine entsprechende Fehlermeldung erscheint.

3.3.7 Quickstop (Schnellhalt)

Die Funktion "Quickstop" ermöglicht, die Achsen mit einer größeren Bremsrampe und einem größeren Ruck als mit den über **MC_Stop** programmierten Werten anzuhalten. Die Werte für die Quickstop-Bremsrampe und für den Quickstop-Ruck müssen mit dem Funktionsbaustein **MC_WriteParameter** (📖 140) in die Parameter **QUICK_STOP_DECELERATION** und **QUICK_STOP_JERK** geschrieben werden.

Die beiden Parameter werden mit den konfigurierten Werten für die Bremsrampe und den Ruck vorinitialisiert. Um den Quickstop zu aktivieren, muss der Parameter **ENABLE_QUICK_STOP** mit dem Funktionsbaustein **MC_WriteBoolParameter** (📖 141) auf **TRUE** gesetzt werden. Beim Deaktivieren muss der Parameter **ENABLE_QUICK_STOP** auf **FALSE** gesetzt werden.

Mit dem Aktivieren der Quickstop-Funktion wird das Verhalten des Bremsens mit **MC_Stop** und das Bremsen bei Softwareendschaltern beeinflusst. Das heißt, in beiden Fällen wird das Bremsen mit den Werten aus den Parametern **QUICK_STOP_DECELERATION** und **QUICK_STOP_JERK** ausgeführt bzw. fortgesetzt.

3.3.8 Technische Einheiten

Die Einheit [**u**] für die Länge wird nicht spezifiziert. Die Einheit muss mit der ersten Ableitung (Velocity [**u/s**], mit der zweiten Ableitung (Acceleration/Deceleration [**u/s²**]) und mit der dritten Ableitung (Jerk [**u/s³**]) übereinstimmen.

Die physikalische Bedeutung der Einheit wird mit den Maschinenkonstanten **MK_DISTANCE** (Einheit [**u**]) und **MK_INCREMENTS** festgelegt (📖 49).

Beispiele für [**u**]: mm, m, Grad, Umdrehung.

3.3.9 Virtuelle Achse

Eine virtuelle Achse ist eine Achse, die nur in der Steuerung angelegt (definiert) ist, aber keinen realen Antrieb/Geber besitzt. Die virtuelle Achse kann in der Steuerung wie eine reale Achse über Befehle (MC-Funktionsbausteine) programmiert werden. Der Interpolator berechnet die Sollwerte, die z. B. für eine synchronisierte Bewegung der virtuellen Achse als Master-Achse mit einer realen oder auch virtuellen Slave-Achse genutzt werden können.

Eine virtuelle Achse wird angelegt, indem ihre CAN-Adresse (**MK_CAN_DRIVE_NODE_ID**) und Kanalnummer der analogen Schnittstelle (**MK_ANALOG_DRIVE_CHANNEL**) gleich -1 ist und dem Interpolator mit **MC_DefineAxis** bekannt gegeben wird.

3.3.10 Rotationsachse

Positionen von Achsen, die als Rotationsachsen definiert sind, werden modulo **MK_DISTANCE** (oder **IrRoad** (📖 85)) berechnet.

3.4 Fehlerhandling

3.4.1 Fehlergruppen

Es gibt 3 Fehlergruppen:

Gruppe 1: Fehler, die von Funktionsbausteinen gemeldet werden.

Über die Ausgänge **Error** und **ErrorID** von Funktionsbausteinen werden Fehler gemeldet, die bei der Initialisierung einer neuen Aufgabe festgestellt werden. Diese Fehler bedeuten, dass die neue Aufgabe nicht ausführbar ist, weil die programmierten Parameter nicht plausibel sind oder die Steuerung sich in einem für diese Aufgabe unerlaubten Zustand befindet.

Die Aufgabe des SPS-Programms ist eine entsprechende Reaktion auf diese Fehlermeldungen. Durch das Betriebssystem der Steuerung wird in diesem Fall keine Aktion unternommen.

Gruppe 2: Fehlermeldungen von Achsen (Achsfehler)

Diese Fehlermeldungen werden durch den Achshandler (Bus-Treiber) sowie die Antriebsrechner gemeldet.

Alle Fehler aus dieser Gruppe werden durch die aktiven Funktionsbausteine mit der ErrorID=100 gemeldet. Die genaue Fehlermeldung kann danach mit dem Funktionsbaustein MC_ReadAxisError ausgelesen werden.

Gruppe 3: Fehlermeldungen vom Betriebssystem der Steuerung (Systemfehler)

Diese Fehlermeldungen werden vom Betriebssystem (Firmware) der Steuerung gemeldet.

Alle Fehler aus dieser Gruppe werden durch die aktiven Funktionsbausteine mit der ErrorID=101 gemeldet. Die genaue Fehlermeldung kann danach mit dem Funktionsbaustein MC_ReadAxisError ausgelesen werden.

3.4.2 Achsfehlermeldungen

Diese Fehlermeldungen werden durch den Achshandler (Bus-Treiber) sowie die Antriebsrechner gemeldet. Mit dem Funktionsbaustein **MC_ReadAxisError** kann der gemeldete Fehler ausgelesen werden.

Man unterscheidet zwischen 4 Fehlerklassen (Kategorien):

ER_CATEGORY_LOKAL_LEICHT_CONST = 1;

ER_CATEGORY_LOKAL_SCHWER_CONST = 2;

ER_CATEGORY_GLOBAL_LEICHT_CONST = 3;

ER_CATEGORY_GLOBAL_SCHWER_CONST = 4;

Bei Fehlerklasse 3 und 4 wird die betroffene Achse sofort (ohne Rampen) angehalten und in den Zustand **ErrorStop** übergehen. In diesem Zustand wird dauernd die Sollposition der Achse auf die aktuelle, durch den Antrieb gemeldete Istposition gesetzt.

Die Reaktion auf die Fehlermeldung von der Klasse 1 und 2 wird der SPS-Applikation überlassen.

Wenn eine Master-Achse im Fehlerfall durch den Antriebsrechner angehalten wird, werden auch die gekoppelten Slave-Achsen zwangsweise angehalten.

Wenn eine Slave-Achse im Fehlerfall durch den Antriebsrechner angehalten wird, ist die Aufgabe der SPS die aktive Kopplung, an der diese Slave-Achse beteiligt ist, aufzuheben und über den Funktionsbaustein **MC_Stop** (📖 105) die restlichen gekoppelten Achsen anzuhalten.



Hinweis!

Der durch die Achse gemeldete Fehler wird sofort mit **MC_ReadAxisError** (📖 135) über den Ausgang "Error" ausgegeben, die Fehlerklasse über den Ausgang "ErrorCategory". Unter Umständen kann es einige SPS-Zyklen dauern, bis die durch den Funktionsbaustein gelieferte Fehlernummer (ErrorID) ungleich 0 ist (antriebsabhängig).

Jede Fehlermeldung muss mit dem Funktionsbaustein **MC_Reset** (📖 137) quittiert werden. Wenn sich die Achse im Zustand **ErrorStop** befindet, dann geht sie nach dem Quittieren in den Zustand **Standstill**.

Beim Auftreten eines Achsfehlers wird an die in der SPS-Applikation definierte Task, die mit dem Ereignis **PLC_PRG4** gesteuert wird, ein Event gesendet. Die Task wird außerhalb der konfigurierten Zykluszeit sofort aufgerufen, so dass die SPS schneller auf den Fehler reagieren kann. Die Art des Events kann mit dem Funktionsbaustein **MC_ReadEvent** (📖 205) ausgelesen werden.

3.4.3 Systemfehlermeldungen

Diese Fehlmeldungen werden vom Betriebssystem (Firmware) der Steuerung gemeldet. Mit dem Funktionsbaustein **MC_ReadSysError** kann der gemeldete Fehler ausgelesen werden.

Man unterscheidet zwischen 4 Fehlerklassen (Kategorien):

```
ER_CATEGORY_LOKAL_LEICHT_CONST = 1;  
ER_CATEGORY_LOKAL_SCHWER_CONST = 2;  
ER_CATEGORY_GLOBAL_LEICHT_CONST = 3;  
ER_CATEGORY_GLOBAL_SCHWER_CONST = 4;
```

Bei Fehlerklasse 4 werden sofort alle Achsen in den Zustand **SysErrorStop** übergehen. Entsprechende zusätzliche Reaktion auf diese Fehlmeldung wird der SPS-Applikation überlassen (z. B. alle Freigaben für die Achsen wegnehmen).

Bei Fehlerklasse 3 werden keine Aktionen seitens des Betriebssystems unternommen, obwohl auch in dem Fall alle Aktivitäten angehalten werden sollten. Die entsprechende Reaktion wird der SPS-Applikation überlassen.

Die Reaktion auf die Fehlmeldung der Klasse 1 und 2 wird der SPS-Applikation überlassen, wobei die Fehlmeldungen der Klasse 1 und 2 als Informationen und Hinweise zu betrachten sind.

Nur die Fehlmeldungen der **ErrorCategory = 4** müssen mit dem Funktionsbaustein **MC_ResetSysError** (📖 204) quittiert werden. Nach dem Quittieren werden alle Achsen in den Zustand **Standstill** versetzt, wenn keine weiteren Fehler von der Klasse 4 anstehen.

Fehler **ErrorCategory = 1** bis **3** führen nicht zum Abbruch der laufenden Bewegung und zum Übergang der Achse in den Zustand **SysErrorStop**. Sie müssen also nicht mit **MC_ResetSysError** (📖 204) quittiert werden. Somit werden sie beim Auslesen mit **MC_ReadSysError** (📖 202) nur über einen SPS-Zyklus angezeigt.

Beim Auftreten eines Systemfehlers wird an die in der SPS-Applikation definierte Task, die mit dem Ereignis **PLC_PRG4** gesteuert wird, ein Event gesendet. Die Task wird außerhalb der konfigurierten Zykluszeit sofort aufgerufen, so dass die SPS schneller auf den Fehler reagieren kann. Die Art des Events kann mit **MC_ReadEvent** (📖 205) ausgelesen werden.

**Stop!**

Systemfehler haben größere Priorität als Achsfehler. D. h. wenn sich eine Achse im Zustand **Errorstop** nach einem Achsfehler befindet und ein Systemfehler von der Klasse 4 gemeldet wird, geht die betroffene Achse in den Zustand **SysErrorStop**. Erst nach dem Quittieren des Systemsfehlers mit **MC_ResetSysError** (📖 204) geht die Achse wieder in den Zustand **ErrorStop** über.

Wenn sich eine Achse im Zustand **SysErrorStop** befindet und ein Achsfehler gemeldet wird, bleibt die betroffene Achse weiterhin im Zustand **SysErrorStop**. Erst nach dem Quittieren des Systemfehlers mit **MC_ResetSysError** (📖 204) geht die Achse wieder in den Zustand **ErrorStop** über.

Gleichzeitig anstehende System- und Achsfehler können in beliebiger Reihenfolge quitiert werden.

3.4.4 Liste aller von Funktionsbausteinen gemeldeten Fehler

ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
1	Achse befindet sich in einem für diesen Befehl unzulässigen Zustand	
2	Index der Achse zu groß	Erlaubter Wert 1. ETC-Gerätegeneration: 0 ... 11 2. ETC-Gerätegeneration: 0 ... 63
3	Achse nicht definiert	Achse muss mit MC_DefineAxis definiert werden.
4	Slave-Achse nicht definiert	Achse muss mit MC_DefineAxis definiert werden.
5	Master-Achse nicht definiert	Achse muss mit MC_DefineAxis definiert werden.
6	Reglerfreigabe fehlt	Mit MC_Power muss zunächst Reglerfreigabe gesetzt werden.
7	Velocity zu groß oder < 0	Velocity > MAX_VELOCITY_APPL (206)
8	Acceleration zu groß oder <= 0	Acceleration > MAX_ACCELERATION_APPL (206)
9	Deceleration zu groß oder <= 0	Deceleration > MAX_DECELERATION_APPL (206)
10	Jerk zu groß oder <= 0	Jerk > MAX_JERK_APPL (206)
11	Unzulässiger Wert von Direction	
12	RatioDenominator gleich 0	MC_GerIn
13	Zeiger CamPosPtr gleich 0	MC_CamTableSelect
14	Speicher für CAM-Tabellen voll	MC_CamTableSelect
15	Unzulässiger Wert für CamTabelID	MC_CamIn
16	Unzulässiger Wert für StartMode	MC_CamIn
17	Unzulässiger Wert für CouplingMode	MC_CamIn
18	Differenz zwischen aktueller Position und Startposition aus der CAM-Tabelle der Slave-Achse zu groß	MC_CamIn Größer als maximaler Ruck
19	Diese Instanz von Funktionsbaustein ist für eine andere Achse aktiv.	
20	Unzulässiger Wert für BufferMode	
22	Unzulässiger Wert von ParameterNumber	MC_ReadParameter ; MC_ReadBoolParameter , MC_WriteParameter , MC_WriteBoolParameter
23	Für eine Achse von dieser Achsenart ist die Freigabe nicht notwendig.	MC_PowerOn . Keine geregelte Achse
24	Für eine Achse von dieser Achsenart ist diese Bewegung nicht erlaubt.	Kontrolle MK_AXIS_TYPE
25	Bei der Ausführung dieser Bewegung würde der negative Softwareendschalter überschritten werden.	MC_MoveAbsolute , MC_MoveRelative , MC_MoveAdditive , MC_MoveSuperimposed
26	Bei der Ausführung dieser Bewegung würde der positive Softwareendschalter überschritten werden.	
27	OverrSpecification falsch	MC_SetOverride
28	VelFactor kleiner 0 oder größer 1.2	MC_SetOverride
29	AccFactor < 0 oder >1.2	MC_SetOverride
30	JerkFactor < 0 oder >1.2	MC_SetOverride

ErrorID	Bedeutung	Hinweis
31	Nicht genügend Speicher, um den Filter zu aktivieren	MC_DefineAxis ; der notwendige Speicher für den Filter kann nicht reserviert werden. Die Zeitkonstante MK_FILTER_TIME_CONST oder rT2 muss verringert werden.
32	NoOfRows in CamTable-Tabelle < 2	MC_CamTableSelect ; die Tabelle muss mindestens zwei Stützstellen beinhalten.
33	NoOfRows in TimePosition-Tabelle < 1	MC_PositionProfile ; die Tabelle muss mindestens eine Stützstelle beinhalten.
34	NoOfRows in TimeVelocity-Tabelle < 1	MC_VelocityProfile ; die Tabelle muss mindestens eine Stützstelle beinhalten.
35	NoOfRows in TimeAcceleration-Tabelle < 1	MC_AccelerationProfile ; die Tabelle muss mindestens eine Stützstelle beinhalten.
36	CouplingDistance = 0	MC_CamIn
40	Unzulässiger Wert von SubTableType	MC_CamLargeTableSelect
41	Speicher voll	MC_CamLargeTableSelect ; zu wenig Speicher zum Anlegen einer neuen Tabelle
42	Tabelle kann zur Zeit nicht gelöscht werden.	MC_CamTableDelete ; eine nach dieser Tabelle gekoppelte Bewegung ist gerade aktiv.
50	Eingang TimePosition = 0	MC_PositionProfile
51	Eingang TimeVelocity = 0	MC_VelocityProfile
52	Eingang TimeAcceleration = 0	MC_AccelerationProfile
53	Eingang PositionScale = 0	MC_PositionProfile
54	Eingang VelocityScale = 0	MC_VelocityProfile
55	Eingang AccelerationScale = 0	MC_AccelerationProfile
56	Eingang TimeScale <= 0	MC_PositionProfile/MC_VelocityProfile/MC_AccelerationProfile
100	Achsfehler ist während der Ausführung des Funktionsbausteins aufgetreten.	Achsfehler ist mit MC_ReadAxisError einzulesen.
101	Systemfehler ist während der Ausführung des Funktionsbausteins aufgetreten.	Systemfehler ist mit MC_ReadSysError einzulesen.
1000	MC_DefineAxis : Impulse = 0	
1001	MC_DefineAxis : Road = 0	
1002	MC_DefineAxis : Velocity <= 0	
1003	MC_DefineAxis : Acceleration <= 0	
1004	MC_DefineAxis : Deceleration <= 0	
1005	MC_DefineAxis : Jerk <= 0	
1006	MC_DefineAxis : CAN-Knotennummer außerhalb des zulässigen Bereichs	Erlaubte Knotennummer 1. ETC-Gerätegeneration: 1 ... 12 2. ETC-Gerätegeneration: 1 ... 64
1008	MC_DefineAxis : ReglerMode falsch	
1010	MC_DefineAxis : zu viele Achsen konfiguriert	Maximale Anzahl von Achsen 1. ETC-Gerätegeneration: 16 2. ETC-Gerätegeneration: 64
1011	MC_TorqueControl : Torque <=0	
1012	MC_TorqueControl : TorqueRamp <=0	
1013	MC_DefineAxis : TorqueTechUnitsDenom=0	

ErrorID	Bedeutung	Hinweis
1014	MC_DefineAxis: TorqueTechUnits-Num=0	
1015	MC_DefineAxis: VelocityTechUnitsDenom = 0	
1016	MC_DefineAxis: VelocityTechUnitsNum = 0	
1020	MC_DefineAxis: Achse bereits definiert	
1021	MC_WriteMachineConstant: Const-Name ist falsch	Maschinenkonstante nicht vorhanden
1022	MC_WriteMachineConstant: Typ der Maschinenkonstante ist falsch	Interner Fehler
1023	MC_WriteMachineConstant: zu viele Parameter für diese Maschinenkonstante	Diese Knotennummer wurde schon an eine Achse vergeben
1024	MC_DefineAxis: Doppelte Knotennummer	
1025	MC_DefineAxis: Aktuelle Steuerung ist keine EMC66-Steuerung	Firmware austauschen
1026	MC_DefineAxis: Ungültiger Wert von dwAxis-DigitalInput oder MK_AXIS_DIGITAL_INPUTS	Nur eine Kombination aus Ziffern 1,2,3,4 z.B 2314 erlaubt.
1027	MC_DefineAxis: Zu viele Achsen mit analoger Schnittstelle definiert	Maximale Anzahl von Achsen mit analoger Schnittstelle ist 12.
1028	MC_DefineAxis: Zwei Achsen mit analoger Schnittstelle in demselben Kanal definiert	
1040	MC_TouchProbe: TriggerPattern falsch	> 2
1052	MC_TouchProbe: Input falsch	> 127
1060	MC_ReadDigitalInput/MCReadDigitalOutput/MC_WriteDigitalOutput: IO-Modul falsch	
1061	MC_DefineQuickInputs / MC_DefineQuickOutputs: Unzulässiger Wert für Offset	> 2047
1062	MC_ReadDigitalInput/MCReadDigitalOutput/MC_WriteDigitalOutput: Eingangs-/Ausgangsnummer falsch	>31
1063	MCReadDigitalOutput/MC_WriteDigitalOutput: Ausgangsnummer nicht freigegeben	Mit MC_DefineAxis wurde der Ausgang über CyclicDataOutputsMask nicht freigegeben.
1064	MC_TouchProbe/MC_AbortTrigger: TriggerNumber falsch	> 31
1065	MC_TouchProbe: InputNumber falsch	> 31
1066	MC_TouchProbe: TriggerPlace falsch	>1
1067	MC_TouchProbe: TriggerPattern falsch	>1
1068	Datentransfer bei SDO abgebrochen	MC_TouchProbe/MC_TouchProbe_CAN_LH/MC_AbortTrigger/MC_AbortTrigger_CAN_LH
1069	Timeout bei SDO-Senden	MC_TouchProbe/MC_TouchProbe_CAN_LH/MC_AbortTrigger/MC_AbortTrigger_CAN_LH
1070	MC_DigitalCamSwitch: Tracknummer falsch	> 31
1071	MC_DigitalCamSwitch: Direction in Switches falsch	> 2
1072	MC_DigitalCamSwitch: CamSwitch-Mode in Switches falsch	> 2

ErrorID	Bedeutung	Hinweis
1073	MC_DigitalCamSwitch: Ein Nocken ist für diese Achse und für diese Spur schon aktiv	Für eine Achse können gleichzeitig mehrere Nocken aber für verschiedene Spuren aktiv sein.
1074	MC_DigitalCamSwitch: Unzulässiger Wert für OnCompensationMode	> 1
1080	MC_SetPosition: Mode falsch	> 1
1081	MC_SetPosition: SetPosSpecification falsch	> 2
1090	MC_TorqueControl: Torque zu groß	Torque > MAX_TORQUE_APPL
1091	MC_TorqueControl: TorqueRamp zu groß oder <= 0	TorqueRamp > MAX_TORQUE_RAMP_APPL
1100	StopSpecification falsch	MC_Stop
1110	MC_GearInPos: Geschwindigkeit während der Synchronisationsphase zu groß	
1070	MC_AttachTaskToInt: Eine andere PLC-Task ist schon am Interpolator angekoppelt.	
1071	MC_AttachTaskToInt: Nach der Ankopplung der Task an den Interpolator ist es 5x nacheinander zum Overrun der Task gekommen.	Mit dieser Fehlermeldung wird die Task vom Interpolator entkoppelt.
1111	MC_GearInPos: Beschleunigung während der Synchronisationsphase zu groß	
1112	MC_GearInPos: Verzögerung während der Synchronisationsphase zu groß	
1113	MC_GearInPos: Ruck während der Synchronisationsphase zu groß	
1114	MC_GearInPos: Ausführung der Synchronisationsphase, ohne Geschwindigkeit oder Beschleunigung oder Verzögerung oder Ruck zu verletzen, nicht möglich trotz der fünfmaligen Verlängerung MasterStartDistance um Masterperiode.	
1115	MasterStartDistance = 0	
1150	MC_CalcTimeToMove/MC_CalcVelocityToMove: Berechnung nicht möglich	Die Bewegung ist bei den vorgegebenen Parametern nicht möglich.
1151	MC_CalcTimeToMove/MC_CalcVelocityToMove: Eingangsparameter Distance = 0	
1160	MC_GearInPos für vorgegebene Slave-Achse befindet sich nicht beim Warten auf die Startposition der Master-Achse.	MC_GearInPos wurde entweder nicht aktiviert oder befindet sich schon mindestens in der Synchronisationsphase.
1200	MC_CamExtTableSelect: Master-Periode = 0	Die erste und die letzte Masterposition in der Tabelle sind gleich
1201	MC_CamExtTableSelect: Unzulässige Interpolationsart in Interpol	> 2
1202	MC_CamExtTableSelect: Unzulässige Anzahl von relevanten Spalten in Interpol	> 3
1204	MC_CamExtTableSelect: Masterpositionen nicht aufsteigend	
1205	MC_CamExtTableSelect: Vorgabe der 1. Ableitung bei den gleichen Masterpositionen fehlt	
1206	MC_CamExtTableSelect: Vorgabe der 2. Ableitung bei den gleichen Masterpositionen fehlt	

ErrorID	Bedeutung	Hinweis
1207	MC_CamExtTableSelect: Vorgabe der 1. und der 2. Ableitung fehlt	Bei $m=2$, d.h. Spline 5.ter Ordnung muss bei der 1ten Stütz-stelle die 1.te und die 2.te Ableitung vorgegeben werden
1208	MC_CamExtTableSelect: Ungleiche Slavepositionen bei den zwei gleichen Masterpositionen	
1209	MC_CamExtTableSelect: Zwei gleiche Masterpositionen bei Linearinterpolation	

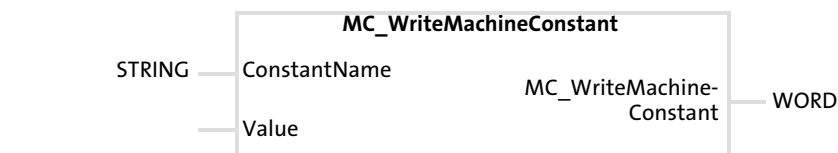
4 Funktionsbibliothek MC.lib

4.1 Funktionsbausteine für die Konfiguration und Initialisierung

4.1.1 Funktion MC_WriteMachineConstant

Aufgabe Mit dieser Funktion können die nicht achsbezogenen Konfigurationsparameter geschrieben werden.

Schematische KOP/FUP-Darstellung



Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_INPUT			
ConstantName	STRING	P	Name der Maschinenkonstante wie in der Maschinenkonstantendatei. Erlaubte Namen: MK_TEST_WITHOUT_MECHANICS MK_CAN_DRIVE_BAUDRATE MK_LANGUAGE MK_INTERPOL_CYCLE MK_FINE_INTERPOL_CYCLE_NUMBER MK_NETDISK_SRVADDR
Value	STRING	P	Der neue Wert der Maschinenkonstanten
VAR_OUTPUT			
MC_WriteMachineConstant	WORD	-	0: MK wurde erfolgreich geschrieben <>0: Fehlercode

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Funktionsbeschreibung

Beispiele und Zeitpunkt der Wirksamkeit einzelner Maschinenkonstanten:

MK_TEST_WITHOUT_MECHANICS	
Beispiel	Testbetrieb einschalten: wOut:=MC_WriteConstantParameter("MK_TEST_WITHOUT_MECHANICS", '1'); Testbetrieb ausschalten: wOut:=MC_WriteConstantParameter("MK_TEST_WITHOUT_MECHANICS", '0');
Wirksamkeit	Die Maschinenkonstante ist erst nach der erneuten Definition von Achsen wirksam (MC_DefineAxisReset->MC_DefineAxis-> MC_InitAxes).
MK_CAN_DRIVE_BAUDRATE	
Beispiel	wOut:=MC_WriteConstantParameter("MK_CAN_DRIVE_BAUDRATE", '1000');
Wirksamkeit	Die Maschinenkonstante ist sofort wirksam.

MK_LANGUAGE

Beispiel	wOut:=MC_WriteConstantParameter("MK_LANGUAGE", '1');
Wirksamkeit	Die Maschinenkonstante ist sofort wirksam.



Hinweis!

Die Sprache kann auch mit der Funktion SETLANGUAGE gesetzt werden.

MK_INTERPOL_CYCLE

Beispiel	wOut:=MC_WriteConstantParameter("MK_INTERPOL_CYCLE", '2');
Wirksamkeit	Die Maschinenkonstante ist erst nach der erneuten Definition von Achsen wirksam (MC_DefineAxisReset → MC_DefineAxis → MC_InitAxes).

MK_FINE_INTERPOL_CYCLE_NUMBER

Beispiel	wOut:=MC_WriteConstantParameter("MK_FINE_INTERPOL_CYCLE_NUMBER", '2');
Wirksamkeit	Die Maschinenkonstante ist erst nach der erneuten Definition von Achsen wirksam (MC_DefineAxisReset → MC_DefineAxis → MC_InitAxes).

MK_NETDISK_SRVADDR

Beispiel	wOut:=MC_WriteConstantParameter("MK_NETDISK_SRVADDR", '172.16.5.114');
Wirksamkeit	Die Maschinenkonstante ist sofort wirksam.



Hinweis!

Die IP-Adresse des Netdisk-Fileservers (ND:) kann auch mit der Funktion "SetCurrentPath" gesetzt werden (SetCurrentPath(ND:172.16.5.114)).

Fehlerbeschreibung

Fehler-code	Bedeutung	Hinweis
0	Kein Fehler	-
1021	ConstName ist falsch	Maschinenkonstante nicht vorhanden
1022	Typ der Maschinenkonstante ist falsch	Interner Fehler
1023	Zu viele Parameter für diese Maschinenkonstante	

4.1.2 Funktion MC_DefineAxisReset

Aufgabe

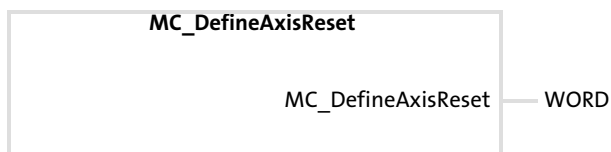
Mit dieser Funktion wird der Interpolator der Steuerung initialisiert. Damit können die Achsen erneut definiert werden. Nach der Ausführung dieser Funktion befindet sich Interpolator im Zustand **Init_Drive_NOT**. Nun kann die Definition von Achsen mit der Funktion **MC_DefineAxis** erfolgen.



Stop!

Die Funktion sollte nur dann aufgerufen werden, wenn sich alle Achsen im Zustand Standstill oder in keiner der Bewegungszustände befinden. Der Interpolator bricht sofort alle laufenden Aufgaben ab, was zu Schleppfehlern bei den Achsen führen kann.

Schematische KOP/FUP-Darstellung



4.1.3 Funktion MC_DefineAxis

Aufgabe

Mit dieser Funktion wird eine Achse definiert und mit entsprechenden Konfigurationsdaten versorgt.



Stop!

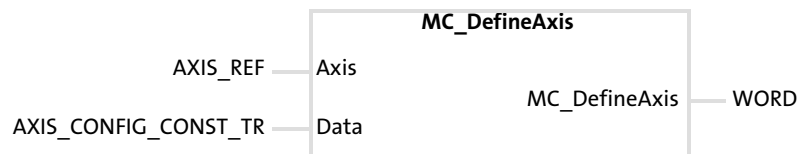
Achsen werden durch den Interpolator in der Reihenfolge bearbeitet, in der sie definiert sind.

Diese Bearbeitungsreihenfolge ist bei gekoppelten Bewegungen von Bedeutung. Definieren Sie immer zunächst eine Master-Achse und erst dann die Slave-Achsen.

Maximale definierbare Achsenanzahl bei den ETCHM-Gerätegenerationen:

1. **Gerätegeneration:** 16 Achsen, davon max. 12 CAN-Achsen
2. **Gerätegeneration:** 60 Achsen, davon max. 48 CAN-Achsen

Schematische KOP/FUP-Darstellung



Parameter

Name	Datentyp	Bedeutung
VAR_INPUT		
Axis	AXIS_REF	Index einer Achse. Wert 0 ... MaxAxis
Data	AXIS_CONFIG_CONST_TR	Zeiger auf die Struktur mit Konfigurationsdaten.
OUTPUT		
MC_DefineAxis	WORD	0: Definition einer Achse erfolgreich abgeschlossen <>0: Fehlercode. Fehler in den Konfigurationsdaten

Funktionsbeschreibung

Mit der Funktion **MC_DefineAxis** werden die Achsen definiert, d. h. die Konfigurationsdaten an den Interpolator und den Drivehandler übergeben.

Über den Eingang **Data** kann ein Zeiger auf eine Datenstruktur übergeben werden, die achsbezogene Maschinenkonstanten beinhaltet. Wenn der Zeiger gleich Null ist, werden die achsbezogenen Maschinenkonstanten aus der Maschinenkonstantendatei gelesen.

Die allgemeinen Maschinenkonstanten werden immer aus der Maschinenkonstantendatei gelesen.

Wenn der Eingang **Data** gleich Null ist und die Maschinenkonstantendatei nicht vorhanden ist, werden die Steuerung und die Achsen mit den Initialisierungswerten der Maschinenkonstanten initialisiert.

Definierte Datenstrukturen

```

TYPE AXIS_CONFIG_CONST_TR :      (* relevant für: equivalent zu: *)
STRUCT                            (* Achse- MK: Initialisierungswert *)
                                (* ANA - analog *)
                                (* VIRT - virtuell *)

    sDriveAddress:                SINT;  (* CAN/VIRT: MK_CAN_DRIVE_NODE_ID: -1 *)
    bArt:                        BYTE;   (* CAN/ANA/VIRT: MK_AXIS_TYPE: 0 *)
    lrImpulse:                   LREAL;  (* CAN/ANA: MK_INCREMENTS: 65536 *)
    lrRoad:                      LREAL;  (* CAN/ANA: MK_DISTANCE: 10 *)
    lrHomeoffset:                LREAL;  (* CAN/ANA: MK_HOME_OFFSET: 0 *)
    lrSW_LimitNeg:               LREAL;  (* CAN/ANA/VIRT: MK_SW_LIMIT_NEGATIVE:
-100000 *)
    lrSW_LimitPos:               LREAL;  (* CAN/ANA/VIRT: MK_SW_LIMIT_POSITIVE: 100000
*)
    lrMaxVelocity:               LREAL;  (* CAN/ANA/VIRT: MK_MAX_VELOCITY: 100 *)
    lrMaxAcceleration:           LREAL;  (* CAN/ANA/VIRT: MK_MAX_ACCELERATION: 1000 *)
    lrMaxDeceleration:           LREAL;  (* CAN/ANA/VIRT: MK_MAX_DECELERATION: 1000 *)
    lrMaxJerk:                   LREAL;  (* CAN/ANA/VIRT: MK_MAX_JERK: 10000 *)
    rT2:                         REAL;   (* CAN/ANALOG: MK_FILTER_TIME_CONST: 0 *)
    lrMaxTorque:                 LREAL;  (* CAN: MK_MAX_TORQUE: 1 *)
    lrMaxTorqueRamp:             LREAL;  (* CAN: MK_MAX_TORQUE_RAMP: 10 *)
    lrTorqueTechUnitsDenom:      LREAL;  (* CAN: MK_TORQUE_TECH_UNITS_DENOM: 1 *)
    lrTorqueTechUnitsNum:        LREAL;  (* CAN: MK_TORQUE_TECH_UNITS_NUM: 1 *)
    lrVelocityTechUnitsDenom:    LREAL;  (* CAN: MK_VELOCITY_TECH_UNITS_DENOM: 1 *)
    lrTorqueTechUnitsNum:        LREAL;  (* CAN: MK_VELOCITY_TECH_UNITS_NUM: 1 *)

    sAnalogDriveChannel:         SINT;  (* ANALOG/VIRT: MK_ANALOG_DRIVE_CHANNEL: -1
*)
    bReglerMode:                 BYTE;   (* ANALOG: MK_REGLER_MODE: 2 *)
    bSSI_Encoder:                BYTE;   (* ANALOG: MK_SSI_ENCODER: 0 *)
    bUA0:                        BYTE;   (* ANALOG: MK_UA0: 0 *)
    bUA12:                       BYTE;   (* ANALOG: MK_UA12: 0 *)
    bUS:                         BYTE;   (* ANALOG: MK_US: 0 *)
    dwAxisDigitalInput:          WORD;   (* ANALOG: MK_AXIS_DIGITAL_INPUTS: 1234 *)
    rTV:                         REAL;   (* ANALOG: MK_TV: 0 *)
    rTN:                         REAL;   (* ANALOG: MK_TN: 0 *)
    rKP:                         REAL;   (* ANALOG: MK_KP: 1 *)
    rKF:                         REAL;   (* ANALOG: MK_KF: 0 *)
    rKB:                         REAL;   (* ANALOG: MK_KB: 0 *)
    bHomingType:                 BYTE;   (* ANALOG: MK_HOMING_TYPE: 0 *)
    rHomingMaxVelToCam:           REAL;  (* ANALOG: MK_HOMING_MAX_VEL_TO_CAM: 200
*)
    rHomingMaxAccToCam:           REAL;  (* ANALOG: MK_HOMING_MAX_ACC_TO_CAM:
500*)
    rHomingMaxVelToZero:          REAL;  (* ANALOG: MK_HOMING_MAX_VEL_TO_ZERO: 100
*)
    rHomingMaxAccToZero:          REAL;  (* ANALOG: MK_REF_MAX_ACC_TO_ZERO: 500 *)
    rDriftCompensation:           REAL;  (* ANALOG: MK_DRIFT_COMPENSATION: 0 *)
    rQuickStop:                  REAL;   (* ANALOG: MK_QUICKSTOP: 0 *)
    wMaxPositionLag:              WORD;  (* ANALOG: MK_MAX_POSITION_LAG: 0x7fff *)
    wMaxPositionLagCounter:       BYTE;  (* ANALOG: MK_MAX_POSITION_LAG_COUNTER: 1 *)
END_STRUCT
END_TYPE

```

Fehlerbeschreibung

Beim Auftreten eines Fehlers in der Funktion wird die Fehlernummer als Ausgabewert geliefert.

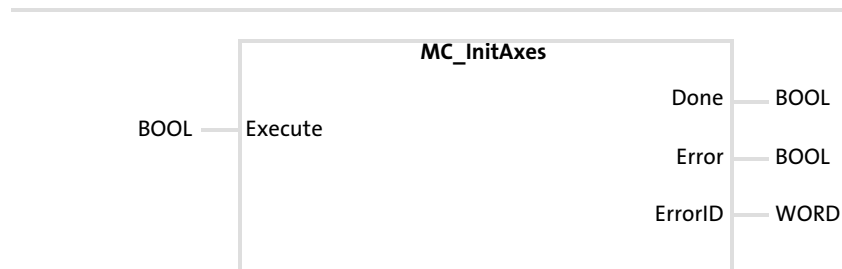
Fehler-code	Bedeutung	Hinweis
0	Kein Fehler	-
1	Unerlaubter Zustand der Achse	Der FB kann nur im Zustand des Interpolators INIT_DRIVE_NOT_E ausgeführt werden. Der Zustand kann mit dem Aufruf von MC_DefineAxisReset erzwungen werden.
2	Eingang Axis außerhalb des zulässigen Bereichs	Erlaubter Wert 1. ETC-Gerätegeneration: 0 ... 15 2. ETC-Gerätegeneration: 0 ... 59
31	Nicht genügend Speicher, um Filter zu aktivieren	Der notwendige Speicher für Filter kann nicht reserviert werden. Die Zeitkonstante MK_Filter_Time_Const oder rT2 muss verringert werden.
1000	MK_INCREMENTS oder IrImpulse = 0.	
1001	MK_DISTANCE oder IrRoad = 0.	
1002	MK_MAX_VELOCITY oder IrMaxVelocity <= 0	
1003	MK_MAX_ACCELERATION oder IrMaxAcceleration <= 0	
1004	MK_MAX_DECELERATION oder IrMaxDeceleration <= 0	
1005	MK_MAX_JERK oder IrMaxJerk <= 0	
1006	CAN-Knotennummer außerhalb des zulässigen Bereichs (MK_CAN_DRIVE_NODE_ID/bDriveAddress)	Erlaubte Knotennummer 1. ETC-Gerätegeneration: 1 ... 12 2. ETC-Gerätegeneration: 1 ... 48
1010	Zu viele Achsen konfiguriert	Maximale Anzahl von Achsen 1. ETC-Gerätegeneration: 16 2. ETC-Gerätegeneration: 60
1011	MK_MAX_TORQUE oder IrMaxTorque <= 0	
1012	MK_MAX_TORQUE_RAMP oder IrMaxTorqueRamp <= 0	
1013	MK_TORQUE_TECH_UNITS_DENOM oder IrTorqueTechUnitsDenom = 0	
1014	MK_TORQUE_TECH_UNITS_NUM oder IrTorqueTechUnitsNum = 0	
1015	MK_VELOCITY_TECH_UNITS_DENOM oder IrVelocityTechUnitsDenom = 0	
1016	MK_VELOCITY_TECH_UNITS_NUM oder IrVelocityTechUnitsNum = 0	
1020	Achse bereits definiert	
1024	Doppelte Knotennummer	Einer Achse wurde diese Knotennummer schon vergeben.
1025	Aktuelle Steuerung ist keine ETC-Steuerung Generation 2	Firmware austauschen.
1026	Ungültiger Wert von dwAxisDigitalInput oder MK_AXIS_DIGITAL_INPUTS	Nur eine Kombination aus Ziffern 1,2,3,4 z.B 2314 erlaubt.
1027	Zu viele Achsen mit analoger Schnittstelle definiert	Maximale Anzahl von Achsen mit analoger Schnittstelle ist 12.
1028	Zwei Achsen mit analoger Schnittstelle in demselben Kanal definiert	

4.1.4 Funktionsbaustein MC_InitAxes

Aufgabe

Mit diesem Funktionsbaustein werden alle mit **MC_DefineAxis** definierten Achsen beim Interpolator und Drivehandler initialisiert.

Schematische KOP/FUP-Darstellung



Parameter

Name	Datentyp	Bedeutung
VAR_INPUT		
Execute	BOOL	Mit steigender Flanke wird die Initialisierung gestartet.
VAR_OUTPUT		
Done	BOOL	TRUE = Initialisierung beendet
Error	BOOL	Dieser Ausgang wird auf TRUE gesetzt, wenn ein Fehler bei der Ausführung dieses Befehls aufgetreten ist.
ErrorID	WORD	Fehlercode.

Fehlerbeschreibung

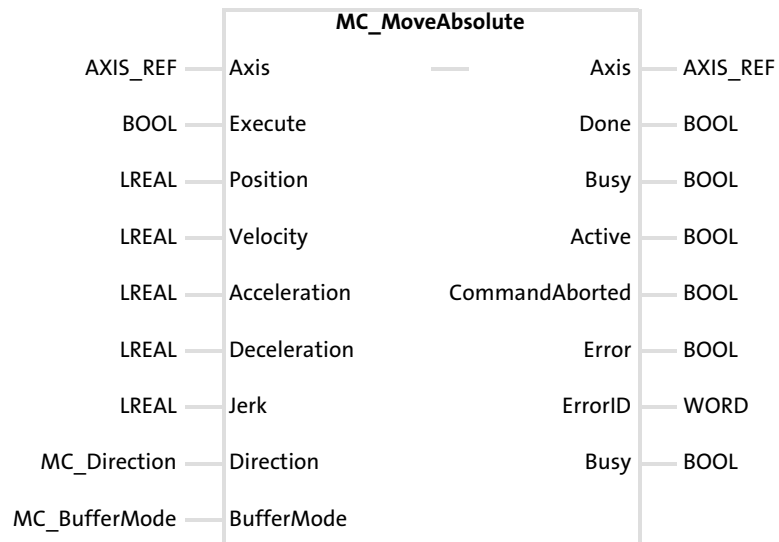
ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
1	Zustand nicht erlaubt	Interpolator befindet sich nicht im Zustand Init_Drive_NOT . Interpolator muss mit der Funktion MC_DefineAxisReset in diesen Zustand versetzt werden.
100	Achsfehler ist während der Ausführung des Funktionsbausteins aufgetreten.	Achsfehler ist mit MC_ReadAxisError einzulesen.
101	Systemfehler ist während der Ausführung des Funktionsbausteins aufgetreten.	Systemfehler ist mit MC_ReadSysError einzulesen.

4.2 Motion-Funktionsbausteine nach PLCopen-Standard

4.2.1 Funktionsbaustein MC_MoveAbsolute

Aufgabe Mit diesem Funktionsbaustein kann man eine Achse auf die programmierte Position absolut positionieren.

Schematische KOP/FUP-Darstellung



Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Axis	AXIS_REF	P	Index einer Achse Wert 0 ... MaxAxis
VAR_INPUT			
Execute	BOOL	P	Mit steigender Flanke wird die Bewegung gestartet.
Position	LREAL	O	Abs. Zielposition in [u] (negativ oder positiv)
Velocity	LREAL	O	Sollgeschwindigkeit in [u/s] (nur positiv)
Acceleration	LREAL	O	Sollbeschleunigung in [u/s ²] (nur positiv)
Deceleration	LREAL	O	Sollverzögerung in [u/s ²] (nur positiv)
Jerk	LREAL	O	Sollbeschleunigungs-/Verzögerungsanstieg (Ruck) in [u/s ³] (nur positiv)

Name	Datentyp	P/O ¹⁾	Bedeutung
Direction	MC_Direction	O	Enum-Typ MC_Direction. Nur relevant für Rotationsachsen. <pre> TYPE MC_Direction : (MD_NEGATIV_E := -1, MD_POSITIV_E := 1, MD_SHORTEST_WAY_E := 2); END_TYPE </pre> Bei MD_NEGATIV_E wird bezogen auf das Achskoordinatensystem in negative Richtung auf die Zielposition gefahren. Bei MD_POSITIV_E wird bezogen auf das Achskoordinatensystem in positive Richtung auf die Zielposition gefahren. Bei MD_SHORTEST_WAY_E wird auf dem kürzesten Weg auf die Zielposition gefahren.
BufferMode	MC_BufferMode	O	Enum-Typ MC_BufferMode. Gilt nur für die Übergänge zwischen MC_MoveAbsolute, MC_MoveRelative, MC_MoveAdditive. <pre> TYPE MC_BufferMode : (BM_ABORTING_E := 0, BM_BUFFERED_E := 1, BM_BLENDING_LOW_E := 2, BM_BLENDING_PREVIOUS_E := 3, BM_BLENDING_NEXT_E := 4, BM_BLENDING_HIGH_E := 5, BM_MASTER_POSITION_E := 6); END_TYPE </pre> Bei BM_ABORTING_E wird die aktuelle Bewegung sofort abgelöst. Bei BM_BUFFERED_E wird diese Aufgabe erst gestartet, wenn die aktuelle beendet wird. Die Achse wird zwischen beiden Bewegungen angehalten. Bei BM_BLENDING_LOW_E wird diese Aufgabe erst gestartet, wenn die aktuelle beendet wird. Die Achse wird zwischen beiden Bewegungen nicht angehalten. Der Übergang wird mit der kleineren Geschwindigkeit von beiden Bewegungen ausgeführt. Bei BM_BLENDING_PREVIOUS_E wird diese Aufgabe erst gestartet, wenn die aktuelle beendet wird. Die Achse wird zwischen beiden Bewegungen nicht angehalten. Der Übergang wird mit der Geschwindigkeit der ersten Bewegung ausgeführt. Bei BM_BLENDING_NEXT_E wird diese Aufgabe erst gestartet, wenn die aktuelle beendet wird. Die Achse wird zwischen beiden Bewegungen nicht angehalten. Der Übergang wird mit der Geschwindigkeit der nächsten Bewegung ausgeführt. Bei BM_BLENDING_HIGH_E wird diese Aufgabe erst gestartet, wenn die aktuelle beendet wird. Die Achse wird zwischen beiden Bewegungen nicht angehalten. Der Übergang wird mit der höheren Geschwindigkeit von beiden Bewegungen ausgeführt.
VAR_OutPUT			
Done	BOOL	-	TRUE zeigt, dass die Zielposition erreicht wurde.
Busy	BOOL	-	Wird mit steigender Flanke von Execute gesetzt.
Active	BOOL	-	Bewegung in Ausführung

Name	Datentyp	P/O ¹⁾	Bedeutung
CommandAborted	BOOL	-	TRUE zeigt, dass die gestartete Bewegung durch eine andere Bewegung für diese Achse abgebrochen wurde. Es handelt sich dabei um eine andere Instanz von demselben Funktionsbaustein oder eine andere Bewegungsart, die zum Zustandswechsel der Achse führt.
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn ein Fehler bei der Ausführung dieses Befehls aufgetreten ist.
ErrorID	WORD	-	Fehlercode

1) P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Initialisierungswerte

Position	0.0
Velocity	50
Acceleration	1000
Deceleration	1000
Jerk	10000
Direction	MD_POSITIV_E
BufferMode	BM_ABORTING_E

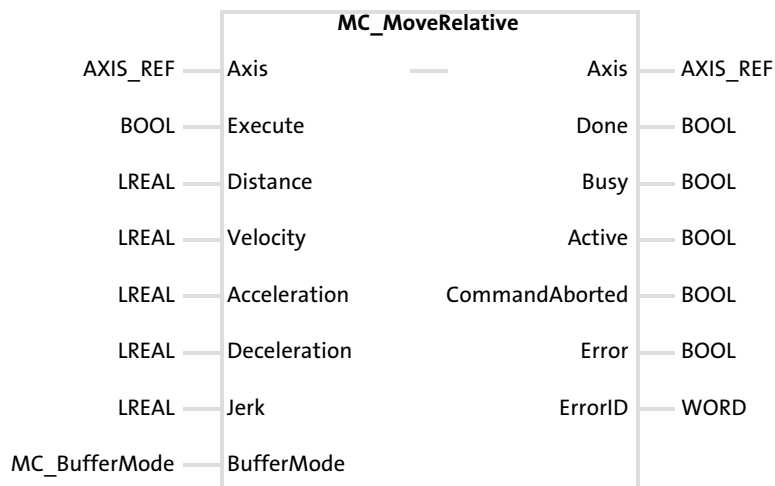
Fehlerbeschreibung

ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
1	Achse befindet sich in einem Zustand, aus dem diese Bewegung nicht gestartet werden kann.	Erlaubte Zustände: Standstill, Discrete Motion, Continuous Motion, Synchronized Motion
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
6	Reglerfreigabe fehlt	Mit MC_Power muss zunächst Reglerfreigabe gesetzt werden.
7	Velocity zu groß oder < 0	Velocity > MAX_VELOCITY_APPL (206)
8	Acceleration zu groß oder <= 0	Acceleration > MAX_ACCELERATION_APPL (206)
9	Deceleration zu groß oder <= 0	Deceleration > MAX_DECELERATION_APPL (206)
10	Jerk zu groß oder <= 0	Jerk > Parameter MAX_JERK_APPL (206)
11	Unzulässiger Wert von Direction	
19	Instanz ist für eine andere Achse aktiv	
20	Unzulässiger Wert für BufferMode	
24	Für eine Achse von dieser Achsenart ist diese Bewegung nicht erlaubt.	Kontrolle MK_AXIS_TYPE
25	Bei der Ausführung dieser Bewegung würde negativer Softwareendschalter überschritten werden.	
26	Bei der Ausführung dieser Bewegung würde positiver Softwareendschalter überschritten werden.	
100	Achsfehler ist während der Ausführung des Befehls aufgetreten.	Achsfehler ist mit MC_ReadAxisError einzulesen.
101	Systemfehler ist während der Ausführung des Befehls aufgetreten.	Systemfehler ist mit MC_ReadSysError einzulesen.

4.2.2 Funktionsbaustein MC_MoveRelative

Aufgabe

Mit diesem Funktionsbaustein kann man eine Achse um die programmierte Distanz ab der aktuellen Position fahren.

Schematische
KOP/FUP-Darstellung

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Axis	AXIS_REF	P	Index einer Achse Wert 0 ... MaxAxis
VAR_INPUT			
Execute	BOOL	P	Mit steigender Flanke wird Bewegung gestartet
Distance	LREAL	O	Distanz/Strecke in [u] (negativ oder positiv)
Velocity	LREAL	O	Sollgeschwindigkeit in [u/s] (nur positiv)
Acceleration	LREAL	O	Sollbeschleunigung in [u/s ²] (nur positiv)
Deceleration	LREAL	O	Sollverzögerung in [u/s ²] (nur positiv)
Jerk	LREAL	O	Sollbeschleunigungs-/Verzögerungsanstieg (Ruck) in [u/s ³] (nur positiv)

Name	Datentyp	P/O ¹⁾	Bedeutung
BufferMode	MC_BufferMode	O	Enum-Typ MC_BufferMode. Gilt nur für die Übergänge zwischen MC_MoveAbsolute, MC_MoveRelative, MC_MoveAdditive. <pre> TYPE MC_BufferMode : (BM_ABORTING_E := 0, BM_BUFFERED_E := 1, BM_BLENDING_LOW_E := 2, BM_BLENDING_PREVIOUS_E := 3, BM_BLENDING_NEXT_E := 4, BM_BLENDING_HIGH_E := 5, BM_MASTER_POSITION_E := 6); END_TYPE </pre> Bei BM_ABORTING_E wird die aktuelle Bewegung sofort abgelöst. Bei BM_BUFFERED_E wird diese Aufgabe erst gestartet, wenn die aktuelle beendet wird. Die Achse wird zwischen beiden Bewegungen angehalten. Bei BM_BLENDING_LOW_E wird diese Aufgabe erst gestartet, wenn die aktuelle beendet wird. Die Achse wird zwischen beiden Bewegungen nicht angehalten. Der Übergang wird mit der kleineren Geschwindigkeit von beiden Bewegungen ausgeführt. Bei BM_BLENDING_PREVIOUS_E wird diese Aufgabe erst gestartet, wenn die aktuelle beendet wird. Die Achse wird zwischen beiden Bewegungen nicht angehalten. Der Übergang wird mit der Geschwindigkeit der ersten Bewegung ausgeführt. Bei BM_BLENDING_NEXT_E wird diese Aufgabe erst gestartet, wenn die aktuelle beendet wird. Die Achse wird zwischen beiden Bewegungen nicht angehalten. Der Übergang wird mit der Geschwindigkeit der nächsten Bewegung ausgeführt. Bei BM_BLENDING_HIGH_E wird diese Aufgabe erst gestartet, wenn die aktuelle beendet wird. Die Achse wird zwischen beiden Bewegungen nicht angehalten. Der Übergang wird mit der höheren Geschwindigkeit von beiden Bewegungen ausgeführt.
VAR_OUTPUT			
Done	BOOL	-	TRUE zeigt, dass die Zielposition erreicht wurde.
Busy	BOOL	-	Wird mit steigender Flanke von Execute gesetzt.
Active	BOOL	-	Bewegung in Ausführung
CommandAborted	BOOL	-	TRUE zeigt, dass die gestartete Bewegung durch eine andere Bewegung für diese Achse abgebrochen wurde. Es handelt sich dabei um eine andere Instanz von demselben Funktionsbaustein oder eine andere Bewegungsart, die zum Zustandswechsel der Achse führt.
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn ein Fehler bei der Ausführung dieses Befehls aufgetreten ist.
ErrorID	WORD	-	Fehlercode

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Initialisierungswerte

Distance	0.0
Velocity	50
Acceleration	1000
Deceleration	1000
Jerk-	10000
BufferMode	BM_ABORTING_E

Fehlerbeschreibung

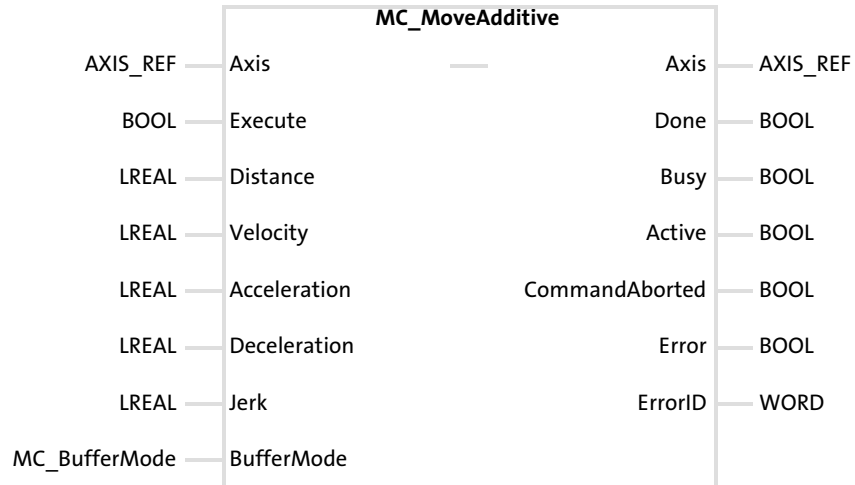
ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
1	Achse befindet sich in einem Zustand, aus dem diese Bewegung nicht gestartet werden kann.	Erlaubte Zustände: Standstill, Discrete Motion, Continuous Motion, Synchronized Motion
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert..
6	Reglerfreigabe fehlt	Mit MC_Power muss zunächst Reglerfreigabe gesetzt werden.
7	Velocity zu groß oder < 0	Velocity > MAX_VELOCITY_APPL (206)
8	Acceleration zu groß oder <= 0	Acceleration > MAX_ACCELERATION_APPL (206)
9	Deceleration zu groß oder <= 0	Deceleration > MAX_DECELERATION_APPL (206)
10	Jerk zu groß oder <= 0	Jerk > MAX_JERK_APPL (206)
19	Diese Instanz ist für eine andere Achse aktiv	
20	Unzulässiger Wert für BufferMode	
24	Für eine Achse von dieser Achsenart ist diese Bewegung nicht erlaubt.	Kontrolle MK_AXIS_TYPE
25	Bei der Ausführung dieser Bewegung würde negativer Softwareendschalter überschritten werden.	
26	Bei der Ausführung dieser Bewegung würde positiver Softwareendschalter überschritten werden.	
100	Achsfehler ist während der Ausführung des Befehls aufgetreten.	Achsfehler ist mit MC_ReadAxisError einzulesen.
101	Systemfehler ist während der Ausführung des Befehls aufgetreten.	Systemfehler ist mit MC_ReadSysError einzulesen.

4.2.3 Funktionsbaustein MC_MoveAdditive

Aufgabe

Mit diesem Funktionsbaustein kann man eine Achse relativ zu dem noch verbleibenden Weg um die programmierte Distanz additiv verfahren.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Wenn die Funktion während der **MC_MoveVelocity**-, **MC_CamIn**- oder **MC_GearIn**-Bewegung gestartet wird, wird diese Basisbewegung abgebrochen und ab der aktuellen Achsposition (zum Zeitpunkt des Starts von MC_MoveAdditive) eine relative Bewegung über die Strecke (Distance) gestartet.

Wenn die Funktion während der **MC_MoveAbsolute**-, **MC_MoveRelative**- oder **MC_MoveAdditive**-Bewegung gestartet wird, wird folgende Position angefahren: Zielposition/Distanz der Basisbewegung (z. B. MC_MoveAbsolute, MC_MoveRelative oder MC_MoveAdditive) plus Distanz des **MC_MoveAdditive**.

Wenn sich die Achse im Zustand Standstill befindet, wird ab der aktuellen Position der Achse die programmierte Strecke zurückgelegt.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Axis	AXIS_REF	P	Index einer Achse Wert 0 ... MaxAxis
VAR_INPUT			
Execute	BOOL	P	Mit steigender Flanke wird die Bewegung gestartet.
Distance	LREAL	O	Additive/relative Distanz/Strecke in [u] (negativ oder positiv)
Velocity	LREAL	O	Sollgeschwindigkeit in [u/s] (nur positiv)
Acceleration	LREAL	O	Sollbeschleunigung in [u/s ²] (nur positiv)
Deceleration	LREAL	O	Sollverzögerung in [u/s ²] (nur positiv)
Jerk	LREAL	O	Sollbeschleunigungs-/Verzögerungsanstieg (Ruck) in [u/s ³] (nur positiv)

Name	Datentyp	P/O ¹⁾	Bedeutung
BufferMode	MC_BufferMode	O	Enum-Typ MC_BufferMode. Gilt nur für die Übergänge zwischen MC_MoveAbsolute, MC_MoveRelative, MC_MoveAdditive. <pre> TYPE MC_BufferMode : (BM_ABORTING_E :=0, BM_BUFFERED_E :=1, BM_BLENDING_LOW_E :=2, BM_BLENDING_PREVIOUS_E :=3, BM_BLENDING_NEXT_E :=4, BM_BLENDING_HIGH_E :=5, BM_MASTER_POSITION_E :=6); END_TYPE </pre> Bei BM_ABORTING_E wird die aktuelle Bewegung sofort abgelöst. Bei BM_BUFFERED_E wird diese Aufgabe erst gestartet, wenn die aktuelle beendet wird. Die Achse wird zwischen beiden Bewegungen angehalten. Bei BM_BLENDING_LOW_E wird diese Aufgabe erst gestartet, wenn die aktuelle beendet wird. Die Achse wird zwischen beiden Bewegungen nicht angehalten. Der Übergang wird mit der kleineren Geschwindigkeit von beiden Bewegungen ausgeführt. Bei BM_BLENDING_PREVIOUS_E wird diese Aufgabe erst gestartet, wenn die aktuelle beendet wird. Die Achse wird zwischen beiden Bewegungen nicht angehalten. Der Übergang wird mit der Geschwindigkeit der ersten Bewegung ausgeführt. Bei BM_BLENDING_NEXT_E wird diese Aufgabe erst gestartet, wenn die aktuelle beendet wird. Die Achse wird zwischen beiden Bewegungen nicht angehalten. Der Übergang wird mit der Geschwindigkeit der nächsten Bewegung ausgeführt. Bei BM_BLENDING_HIGH_E wird diese Aufgabe erst gestartet, wenn die aktuelle beendet wird. Die Achse wird zwischen beiden Bewegungen nicht angehalten. Der Übergang wird mit der höheren Geschwindigkeit von beiden Bewegungen ausgeführt.
VAR_OUTPUT			
Done	BOOL	-	TRUE zeigt, dass die Zielposition erreicht wurde.
Busy	BOOL	-	Wird mit steigender Flanke von Execute gesetzt.
Active	BOOL	-	Bewegung in Ausführung
CommandAborted	BOOL	-	TRUE zeigt, dass die gestartete Bewegung durch eine andere Bewegung für diese Achse abgebrochen wurde. Es handelt sich dabei um eine andere Instanz von demselben Funktionsbaustein oder eine andere Bewegungsart, die zum Zustandswechsel der Achse führt.
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn ein Fehler bei der Ausführung dieses Befehls aufgetreten ist.
ErrorID	WORD	-	Fehlercode

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Initialisierungswerte

Distance	0.0
Velocity	50
Acceleration	1000
Deceleration	1000
Jerk	10000
BufferMode	BM_ABORTING_E

Fehlerbeschreibung

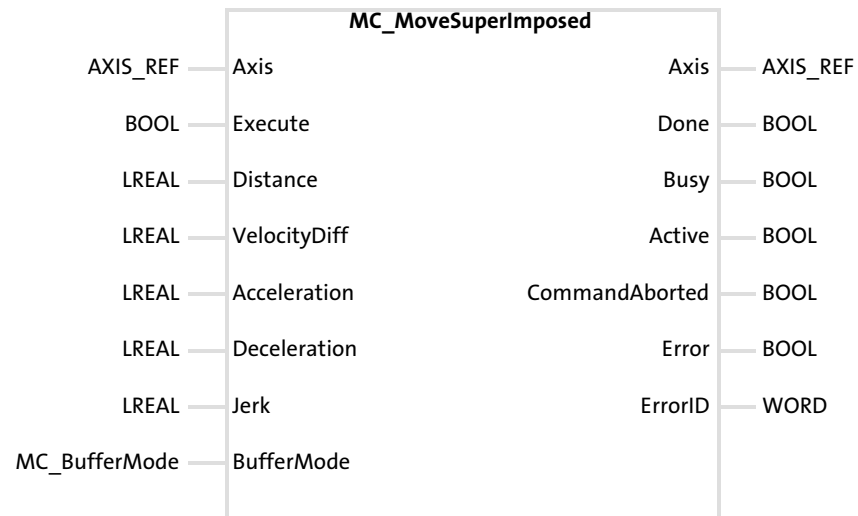
ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
1	Achse befindet sich in einem Zustand, aus dem diese Bewegung nicht gestartet werden kann.	Erlaubte Zustände: Standstill, Discrete Motion, Continuous Motion, Synchronized Motion
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
6	Reglerfreigabe fehlt	Mit MC_Power muss zunächst Reglerfreigabe gesetzt werden
7	Velocity zu groß oder < 0	Velocity > MAX_VELOCITY_APPL (206)
8	Acceleration zu groß oder <= 0	Acceleration > MAX_ACCELERATION_APPL (206)
9	Deceleration zu groß oder <= 0	Deceleration > MAX_DECELERATION_APPL (206)
10	Jerk zu groß oder <= 0	Jerk > MAX_JERK_APPL (206)
19	Diese Instanz ist für eine andere Achse aktiv	
20	Unzulässiger Wert für BufferMode	
24	Für eine Achse von dieser Achsenart ist diese Bewegung nicht erlaubt	Kontrolle MK_AXIS_TYPE
25	Bei der Ausführung dieser Bewegung würde negativer Softwareendschalter überschritten werden	
26	Bei der Ausführung dieser Bewegung würde positiver Softwareendschalter überschritten werden	
100	Achsfehler ist während der Ausführung des Befehls aufgetreten.	Achsfehler ist mit MC_ReadAxisError einzulesen.
101	Systemfehler ist während der Ausführung des Befehls aufgetreten.	Systemfehler ist mit MC_ReadSysError einzulesen.

4.2.4 Funktionsbaustein MC_MoveSuperImposed

Aufgabe

Mit diesem Funktionsbaustein kann man zu einer Achse mit bereits vorhandener Bewegung eine neue Bewegung relativ überlagern.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Im Zustand **Standstill** wirkt **MC_MoveSuperImposed** wie **MC_MoveAbsolute**.

Bei Überlagerung der Basisbewegung mit **MC_MoveSuperImposed**, wird die Achse mit der Geschwindigkeit fahren, die sich aus beiden Bewegungen ergibt. Das gilt auch für Beschleunigung, Verzögerung und Ruck. Es können größere Werte für diese Parameter entstehen als zulässig sind.

Es findet keine automatische Begrenzung der resultierenden Geschwindigkeit, Beschleunigung, Verzögerung und des resultierenden Rucks im Interpolator statt.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Axis	AXIS_REF	P	Index einer Achse 0 ... MaxAxis
VAR_INPUT			
Execute	BOOL	P	Mit steigender Flanke wird die Bewegung gestartet.
Distance	LREAL	O	relative Überlagerungstrecke in [u] (negativ oder positiv)
VelocityDiff	LREAL	O	Geschwindigkeitsdifferenz in [u/s] (nur positiv)
Acceleration	LREAL	O	Sollbeschleunigung in [u/s ²] (nur positiv)
Deceleration	LREAL	O	Sollverzögerung in [u/s ²] (nur positiv)
Jerk	LREAL	O	Sollbeschleunigungs-/Verzögerungsanstieg (Ruck) in [u/s ³] (nur positiv)

Name	Datentyp	P/O ¹⁾	Bedeutung
BufferMode	MC_BufferMode	O	Enum-Typ MC_BufferMode. Gilt nur für die Übergänge zwischen zwei MC_MoveSuperImposed. <pre> TYPE MC_BufferMode : (BM_ABORTING_E :=0, BM_BUFFERED_E :=1, BM_BLENDING_LOW_E :=2, BM_BLENDING_PREVIOUS_E :=3, BM_BLENDING_NEXT_E :=4, BM_BLENDING_HIGH_E :=5, BM_MASTER_POSITION_E :=6); END_TYPE </pre> Bei BM_ABORTING_E wird die aktuelle SuperImposed-Bewegung sofort abgelöst. Bei BM_BUFFERED_E wird diese Aufgabe erst gestartet, wenn die aktuelle SuperImposed-Bewegung beendet wird. Die Achse wird zwischen beiden Bewegungen angehalten. Bei BM_BLENDING_LOW_E wird diese Aufgabe erst gestartet, wenn die aktuelle SuperImposed-Bewegung beendet wird. Die Achse wird zwischen beiden Bewegungen nicht angehalten. Der Übergang wird mit der kleineren Geschwindigkeit von beiden Bewegungen ausgeführt. Bei BM_BLENDING_PREVIOUS_E wird diese Aufgabe erst gestartet, wenn die aktuelle SuperImposed-Bewegung beendet wird. Die Achse wird zwischen beiden Bewegungen nicht angehalten. Der Übergang wird mit der Geschwindigkeit der ersten Bewegung ausgeführt. Bei BM_BLENDING_NEXT_E wird diese Aufgabe erst gestartet, wenn die aktuelle SuperImposed-Bewegung beendet wird. Die Achse wird zwischen beiden Bewegungen nicht angehalten. Der Übergang wird mit der Geschwindigkeit der nächsten Bewegung ausgeführt. Bei BM_BLENDING_HIGH_E wird diese Aufgabe erst gestartet, wenn die aktuelle SuperImposed-Bewegung beendet wird. Die Achse wird zwischen beiden Bewegungen nicht angehalten. Der Übergang wird mit der höheren Geschwindigkeit von beiden Bewegungen ausgeführt.
VAR_OUTPUT			
Done	BOOL	-	TRUE zeigt, dass die Zielposition erreicht wurde.
Busy	BOOL	-	Wird mit steigender Flanke von Execute gesetzt.
Active	BOOL	-	Bewegung in Ausführung
CommandAborted	BOOL	-	TRUE zeigt, dass die gestartete Bewegung durch eine andere Bewegung mit MC_MoveSuperImposed für diese Achse abgebrochen wurde. Es handelt sich dabei um eine andere Instanz von demselben Funktionsbaustein. Wenn diese Bewegung mit MC_Stop angehalten wird, wird auch der Ausgang auf TRUE gesetzt.
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn ein Fehler bei der Ausführung dieses Befehls aufgetreten ist.
ErrorID	WORD	-	Fehlercode

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Initialisierungswerte

Distance	0.0
VelocityDiff	50
Acceleration	1000
Deceleration	1000
Jerk	10000
BufferMode	BM_ABORTING_E

Fehlerbeschreibung

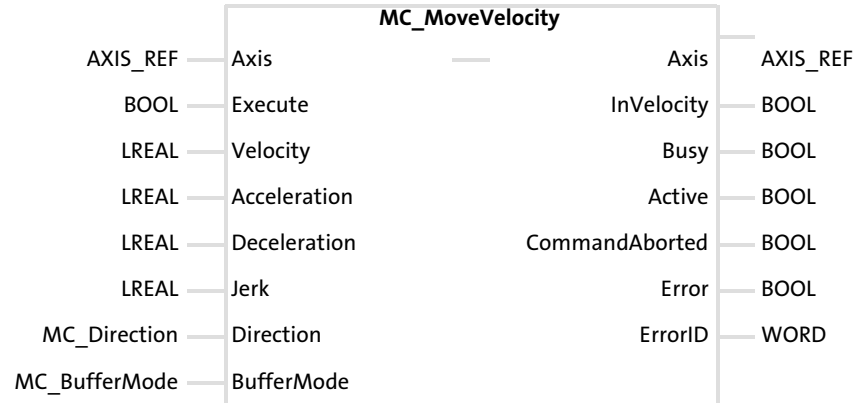
ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
1	Achse befindet sich in einem Zustand, aus dem diese Bewegung nicht gestartet werden kann	Erlaubte Zustände: Standstill, Discrete Motion, Continuous Motion, Synchronized Motion
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
6	Reglerfreigabe fehlt	Mit MC_Power muss zunächst Reglerfreigabe gesetzt werden
7	VelocityDiff zu groß oder < 0	Velocity > MAX_VELOCITY_APPL (206)
8	Acceleration zu groß oder <= 0	Acceleration > MAX_ACCELERATION_APPL (206)
9	Deceleration zu groß oder <= 0	Deceleration > MAX_DECELERATION_APPL (206)
10	Jerk zu groß oder <= 0	Jerk > MAX_JERK_APPL (206)
19	Diese Instanz ist für eine andere Achse aktiv	
20	Unzulässiger Wert für BufferMode	
24	Für eine Achse von dieser Achsenart ist diese Bewegung nicht erlaubt	Kontrolle MK_AXIS_TYPE
25	Bei der Ausführung dieser Bewegung würde negativer Softwareendschalter überschritten werden	
26	Bei der Ausführung dieser Bewegung würde positiver Softwareendschalter überschritten werden	
100	Achsfehler ist während der Ausführung des Befehls aufgetreten.	Achsfehler ist mit MC_ReadAxisError einzulesen.
101	Systemfehler ist während der Ausführung des Befehls aufgetreten.	Systemfehler ist mit MC_ReadSysError einzulesen.

4.2.5 Funktionsbaustein MC_MoveVelocity

Aufgabe

Mit diesem Funktionsbaustein wird eine Endlosfahrt gestartet.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Die **MC_MoveVelocity** wird mit jedem neuen Bewegungsbefehl abgebrochen. Der Ausgang **InVelocity** wird auf TRUE gesetzt, wenn die programmierte Sollgeschwindigkeit erreicht wird und wieder auf FALSE gesetzt wird, wenn diese Bewegung abgebrochen wird oder bei der fallender Flanke des Eingangs **Execute**.

Bei einer mit **MC_SuperImposed** überlagerten Bewegung bleibt der Ausgang **InVelocity** weiterhin gesetzt, wenn die programmierte Geschwindigkeit einmal erreicht wurde.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Axis	AXIS_REF	P	Index einer Achse 0 ... MaxAxis
VAR_INPUT			
Execute	BOOL	P	Mit steigender Flanke wird die Bewegung gestartet.
Velocity	LREAL	O	Sollgeschwindigkeit in [u/s] (nur positiv)
Acceleration	LREAL	O	Sollbeschleunigung in [u/s ²] (nur positiv)
Deceleration	LREAL	O	Sollverzögerung in [u/s ²] (nur positiv)
Jerk	LREAL	O	Sollbeschleunigungs-/Verzögerungsanstieg (Ruck) in [u/s ³] (nur positiv)
Direction	MC_Direction	O	Enum-Typ MC_Direction . TYPE MC_Direction : (MD_NEGATIV_E := -1, MD_POSITIV_E := 1, MD_SHORTEST_WAY_E := 2); END_TYPE Bei MD_NEGATIV_E wird bezogen auf das Achskoordinatensystem in neg. Richtung verfahren. Bei MD_POSITIV_E wird bezogen auf das Achskoordinatensystem in pos. Richtung verfahren. Richtung MD_SHORTEST_WAY_E ist für diese Bewegung nicht relevant.

Name	Datentyp	P/O ¹⁾	Bedeutung
BufferMode	MC_BufferMode		Enum-Typ MC_BufferMode. <pre> TYPE MC_BufferMode : (BM_ABORTING_E :=0, BM_BUFFERED_E :=1, BM_BLENDING_LOW_E :=2, BM_BLENDING_PREVIOUS_E :=3, BM_BLENDING_NEXT_E :=4, BM_BLENDING_HIGH_E :=5, BM_MASTER_POSITION_E :=6); END_TYPE </pre> Bei BM_ABORTING_E wird die aktuelle Bewegung sofort abgelöst. Übergang MC_MoveVelocity -> MC_MoveAbsolute/MC_MoveRelative/MC_MoveAdditive: Bei BM_BUFFERED_E wird die nächste Aufgabe erst gestartet, wenn die aktuelle sich im Zustand inVelocity befindet. Die Achse wird zwischen beiden Bewegungen nicht angehalten. Der Übergang wird mit der Geschwindigkeit der ersten Bewegung ausgeführt. Übergang MC_MoveAbsolute/MC_MoveRelative/MC_MoveAdditive->MC_MoveVelocity Bei BM_BUFFERED_E wird die nächste Aufgabe erst gestartet, wenn die aktuelle beendet wird. Die Achse wird zwischen beiden Bewegungen angehalten. Übergang MC_MoveVelocity <-> MC_MoveAbsolute/MC_MoveRelative/MC_MoveAdditive: Bei BM_BLENDING_LOW_E wird die nächste Aufgabe erst gestartet, wenn die aktuelle beendet wird. Die Achse wird zwischen beiden Bewegungen nicht angehalten. Der Übergang wird mit der kleineren Geschwindigkeit von beiden Bewegungen ausgeführt. Bei BM_BLENDING_PREVIOUS_E wird die nächste Aufgabe erst gestartet, wenn die aktuelle beendet wird. Die Achse wird zwischen beiden Bewegungen nicht angehalten. Der Übergang wird mit der Geschwindigkeit der ersten Bewegung ausgeführt. Bei BM_BLENDING_NEXT_E wird die nächste Aufgabe erst gestartet, wenn die aktuelle beendet wird. Die Achse wird zwischen beiden Bewegungen nicht angehalten. Der Übergang wird mit der Geschwindigkeit der nächsten Bewegung ausgeführt. Bei BM_BLENDING_HIGH_E wird die nächste Aufgabe erst gestartet, wenn die aktuelle beendet wird. Die Achse wird zwischen beiden Bewegungen nicht angehalten. Der Übergang wird mit der höheren Geschwindigkeit von beiden Bewegungen ausgeführt.
VAR_OUTPUT			
InVelocity	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn die programmierte Sollgeschwindigkeit erreicht wurde.
Busy	BOOL	-	Wird mit steigender Flanke von Execute gesetzt.
Active	BOOL	-	Bewegung in Ausführung

Name	Datentyp	P/O ¹⁾	Bedeutung
CommandAborted	BOOL	-	TRUE zeigt, dass die gestartete Bewegung durch eine andere Bewegung für diese Achse abgebrochen wurde. Es handelt sich dabei um eine andere Instanz von demselben Funktionsbaustein oder eine andere Bewegungsart, die zum Zustandswechsel der Achse führt.
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn ein Fehler bei der Ausführung dieses Befehls aufgetreten ist.
ErrorID	WORD	-	Fehlercode

1) P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Initialisierungswerte

Velocity	50
Acceleration	1000
Deceleration	1000
Jerk	10000
Direction	POSITIV
BufferMode	BM_ABORTING_E

Fehlerbeschreibung

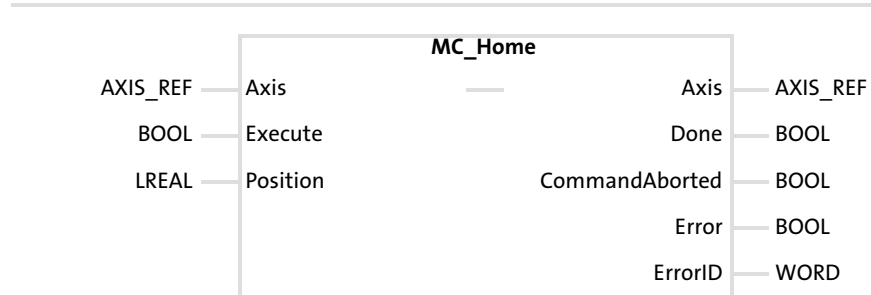
ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
1	Achse befindet sich in einem Zustand, aus dem diese Bewegung nicht gestartet werden kann.	Erlaubte Zustände: Standstill, Discrete Motion, Continuous Motion, Synchronized Motion
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
6	Reglerfreigabe fehlt	Mit MC_Power muss zunächst Reglerfreigabe gesetzt werden.
7	Velocity zu groß oder < 0	Velocity > MAX_VELOCITY_APPL (206)
8	Acceleration zu groß oder <= 0	Acceleration > MAX_ACCELERATION_APPL (206)
9	Deceleration zu groß oder <= 0	Deceleration > MAX_DECELERATION_APPL (206)
10	Jerk zu groß oder <= 0	Jerk > MAX_JERK_APPL (206)
11	Unzulässiger Wert von Direction	
19	Diese Instanz ist für eine andere Achse aktiv.	
20	Unzulässiger Wert für BufferMode	
24	Für eine Achse von dieser Achsenart ist diese Bewegung nicht erlaubt.	Kontrolle MK_AXIS_TYPE
25	Negativer Softwareendschalter	
26	Positiver Softwareendschalter	
100	Achsfehler ist während der Ausführung des Befehls aufgetreten.	Achsfehler ist mit MC_ReadAxisError einzulesen.
101	Systemfehler ist während der Ausführung des Befehls aufgetreten.	Systemfehler ist mit MC_ReadSysError einzulesen.

4.2.6 Funktionsbaustein MC_Home

Aufgabe

Mit diesem Funktionsbaustein kann man eine Achse referenzieren. (Voraussetzung: Referenzpunktfahrt beim Antriebsregler konfiguriert.)

Schematische KOP/FUP-Darstellung



Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Axis	AXIS_REF	P	Index einer Achse. 0 ... MaxAxis
VAR_INPUT			
Execute	BOOL	P	Mit steigender Flanke wird die Referenzpunktfahrt gestartet.
Position	LREAL	O	Achsposition im Referenzpunkt in [u] (positiv oder negativ). Dafür kann die Maschinenkonstante MK_HOME_OFFSET eingesetzt werden: Position = - MK_HOME_OFFSET . Der Wert von MK_HOME_OFFSET kann mit MC_ReadParameter ausgelesen werden.
VAR_OUTPUT			
Done	BOOL	-	Referenziervorgang abgeschlossen
CommandAborted	BOOL	-	Befehl abgebrochen
Error	BOOL	-	TRUE, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Initialisierungswerte

Position	0.0
----------	-----

Fehlerbeschreibung

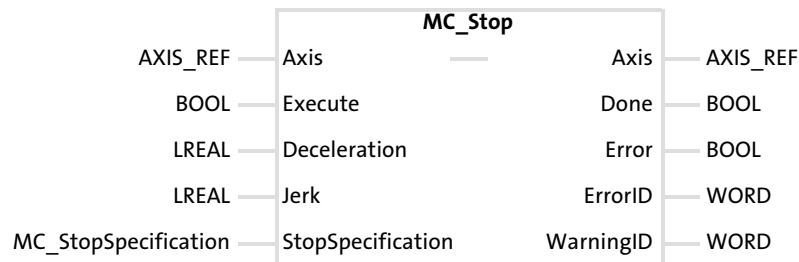
ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
1	Achse nicht im Zustand Standstill	
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
6	Reglerfreigabe fehlt	Mit MC_Power muss zunächst Reglerfreigabe gesetzt werden.
24	Für eine Achse von dieser Achsenart ist diese Bewegung nicht erlaubt	Kontrolle MK_AXIS_TYPE
100	Achsfehler ist während der Ausführung des Befehls aufgetreten.	Achsfehler ist mit MC_ReadAxisError einzulesen.
101	Systemfehler ist während der Ausführung des Befehls aufgetreten.	Systemfehler ist mit MC_ReadSysError einzulesen.

4.2.7 Funktionsbaustein MC_Stop

Aufgabe

Mit diesem Funktionsbaustein kann eine Achse gestoppt werden.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

- Bei **StopSpecification = STOP_SPEC_ALL_AXIS_MOTION_E**:
 - Mit steigender Flanke von **Execute** geht die Achse in den Zustand **Stopping**.
 - Im Zustand **Stopping** wird die Achse über die programmierten Dynamikparameter (Deceleration und Jerk) angehalten und dann der Ausgang **Done** auf TRUE gesetzt.
 - Die Achse bleibt im Zustand **Stopping** und keine weiteren Bewegungsbefehle werden für diese Achse ausgeführt.
 - Wenn der Ausgang **Done** = TRUE ist, geht die Achse erst mit fallender Flanke von **Execute** in den Zustand **Standstill**.
- Bei **StopSpecification = STOP_SPEC_SUPERIMPOSED_MOTION_E**:
 - Mit steigender Flanke von **Execute** wird die überlagerte Bewegung, die mit **MC_MoveSuperImposed** gestartet wurde, gestoppt, wobei die Grundbewegung weiter ausgeführt wird und die Achse im aktuellen Bewegungszustand bleibt.
- Bei **StopSpecification = STOP_SPEC_PHASING_E**:
 - Mit steigender Flanke von **Execute** wird **Phasing** gestoppt, wobei die Grundbewegung weiter ausgeführt wird und die Achse im aktuellen Bewegungszustand bleibt.

Beachten Sie weiterhin:

- Wenn der programmierte Wert für den Eingangsparameter **Deceleration** den maximalen Wert im Parameter **MAX_DECELERATION_APPL** übersteigt, wird das Anhalten der Achse mit dem Wert des Parameters **MAX_DECELERATION_APPL** ausgeführt. Über den Ausgangsparameter **WarningID** wird eine Warnung angezeigt.
- Wenn der programmierte Wert für den Eingangsparameter **Jerk** den maximalen Wert im Parameter **MAX_JERK_APPL** übersteigt, wird das Anhalten der Achse mit dem Wert des Parameters **MAX_JERK_APPL** ausgeführt. Über den Ausgangsparameter **WarningID** wird eine Warnung angezeigt.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Axis	AXIS_REF	P	Index einer Achse. 0 ... MaxAxis
VAR_INPUT			
Execute	BOOL	P	Mit steigender Flanke wird die Ausführung der Funktion gestartet..
Deceleration	LREAL	O	Sollverzögerung in [u/s ²] (nur positiv)
Jerk	LREAL	O	Sollverzögerungsanstieg (Ruck) in [u/s ³] (nur positiv)
StopSpecification	MC_StopSpecification	O	Enum-Typ MC_StopSpecification: TYPE MC_StopSpecification : (STOP_SPEC_ALL_AXIS_MOTION_E :=0, STOP_SPEC_SUPERIMPOSED_E :=1, STOP_SPEC_PHASING_E :=2); END_TYPE Bei STOP_SPEC_ALL_AXIS_MOTION_E – alle Bewegungen der Achse werden gestoppt. Bei STOP_SPEC_SUPERIMPOSED_E – nur die überlagerte Bewegung, die mit MC_MoveSuperimposed gestartet wurde, wird gestoppt. Bei STOP_SPEC_PHASING_E – nur Phasing wird gestoppt.
VAR_OUTPUT			
Done	BOOL	-	TRUE, wenn die Geschwindigkeit den Wert 0 erreicht hat.
Error	BOOL	-	TRUE, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode
WarningID	WORD	-	Warnungscode

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Initialisierungswerte

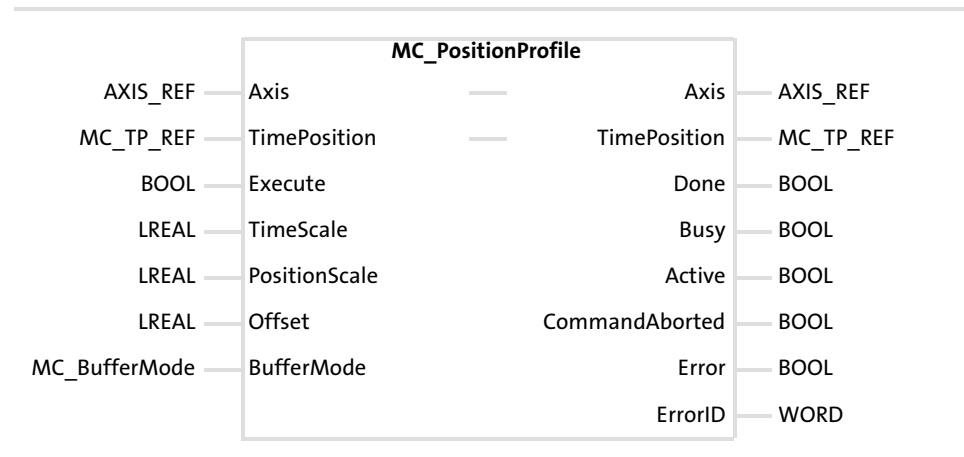
Deceleration	1000
Jerk	10000
StopSpecification	ALL_AXIS_MOTION

Fehlerbeschreibung

ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
1	Achse befindet sich in einem für diese Bewegung unerlaubten Zustand	Achse befindet sich im Zustand Error-stop oder SysErrorStop .
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
6	Reglerfreigabe fehlt	Mit MC_Power muss zunächst Reglerfreigabe gesetzt werden.
9	Deceleration zu groß oder ≤ 0	Deceleration > MAX_DECELERATION_APPL (206)
10	Jerk zu groß oder ≤ 0	Jerk > MAX_JERK_APPL (206)
19	Diese Instanz ist für eine andere Achse aktiv	
24	Für eine Achse von dieser Achsenart ist diese Bewegung nicht erlaubt	Kontrolle MK_AXIS_TYPE
100	Achsfehler ist während der Ausführung des Befehls aufgetreten.	Achsfehler ist mit MC_ReadAxisError einzulesen.
101	Systemfehler ist während der Ausführung des Befehls aufgetreten.	Systemfehler ist mit MC_ReadSysError einzulesen.
1100	Unzulässiger Wert von StopSpecification	

4.2.8 Funktionsbaustein MC_PositionProfile**Aufgabe**

Mit diesem Funktionsbaustein wird eine Positionieraufgabe in Form einer Tabelle, die die Positionen einer Achse abhängig von der Zeit beschreibt, aktiviert.

Schematische KOP/FUP-Darstellung

Funktionsbeschreibung

Vordefinierte Datentypen in mc.lib:

```

TYPE MC_TP_REF :
    STRUCT
        NoOfRows:    DWORD;
        TP_TablePtr:  DWORD;
    END_STRUCT
END_TYPE
TYPE MC_TP_TR :
    STRUCT
        delta_time:   LREAL;
        position:      LREAL;
        interpolation:  MC_PROFILE_INT_ART;
    END_STRUCT
END_TYPE
    
```

NoOfRows: Anzahl von Stützstellen

TP_TablePtr: Adresse der Tabelle mit den Stützstellen. Die Tabelle muss in der Applikation als **ARRAY[0...NoOfRows-1] OF MC_TP_TR** deklariert sein.

delta_time: Die zeitliche Differenz zwischen zwei Positionen in [s].

Bei aktiver Bearbeitung können die Tabellenwerte geändert werden. Man kann auch zwischen verschiedenen Geschwindigkeitsprofil-Tabellen umschalten. Für jedes Segment der Profil-Tabelle kann eine andere Interpolationsart eingegeben werden:

0 - linear

1 - kubischer Spline (stetige 1. Ableitung: die Geschwindigkeit)

2 - Spline 5. Grades (stetige 1. und 2. Ableitung: Geschwindigkeit und Beschleunigung)

Die Tabelle wird immer auf die aktuelle Position und die aktuelle Geschwindigkeit der Achse aufgesetzt. Die aktuelle Beschleunigung der Achse wird vernachlässigt. Bei der Erstellung der Tabelle muss darauf geachtet werden, dass kein Geschwindigkeits- und Beschleunigungssprung beim Start der Bewegung aus der Tabelle entsteht.

Die in der Tabelle vorgegebenen **positionen** können mit den Eingangsparameter **PositionScale** und **Offset** folgendermaßen beeinflusst werden:

position_{korr} = position * PositionScale + Offset

Die in der Tabelle für jede Position vorgegebene Zeit kann mit dem Eingangsparameter **TimeScale** folgendermaßen beeinflusst werden:

delta_time_{korr} = delta_time * TimeScale

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Axis	AXIS_REF	P	Index einer Achse. 0 ... MaxAxis
TimePosition	MC_TP_REF	P	Referenz auf die Zeit/Position-Tabelle
VAR_INPUT			
Execute	BOOL	P	Start der Bewegung mit steigender Flanke
TimeScale	LREAL	O	Skalierungsfaktor für die Zeit des Profils
PositionScale	LREAL	O	Skalierungsfaktor für die Positionen des Profils
Offset	LREAL	O	Verschiebung des Profils [u]
BufferMode	MC_BufferMode	O	Enum-Typ MC_BufferMode. TYPE MC_BufferMode : (BM_ABORTING_E := 0, BM_BUFFERED_E := 1, BM_BLENDING_LOW_E := 2, BM_BLENDING_PREVIOUS_E := 3, BM_BLENDING_NEXT_E := 4, BM_BLENDING_HIGH_E := 5, BM_MASTER_POSITION_E := 6); END_TYPE Bei BM_ABORTING_E wird die aktuelle Bewegung sofort abgelöst. Bei BM_BUFFERED_E wird diese Aufgabe erst gestartet, wenn die aktuelle beendet wird. Bei BM_BLENDING_LOW_E , BM_BLENDING_PREVIOUS_E , BM_BLENDING_NEXT_E , BM_BLENDING_HIGH_E wird der Übergang wie bei BM_BUFFERED_E ausgeführt.
VAR_OUTPUT			
Done	BOOL	-	Ausführung des Profils erfolgreich abgeschlossen.
Busy	BOOL	-	Wird mit steigender Flanke von Execute gesetzt.
Active	BOOL	-	Bewegung in Ausführung.
CommandAborted	BOOL	-	Diese Bewegung wurde durch einen anderen Bewegungsbefehl abgebrochen.
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Initialisierungswerte

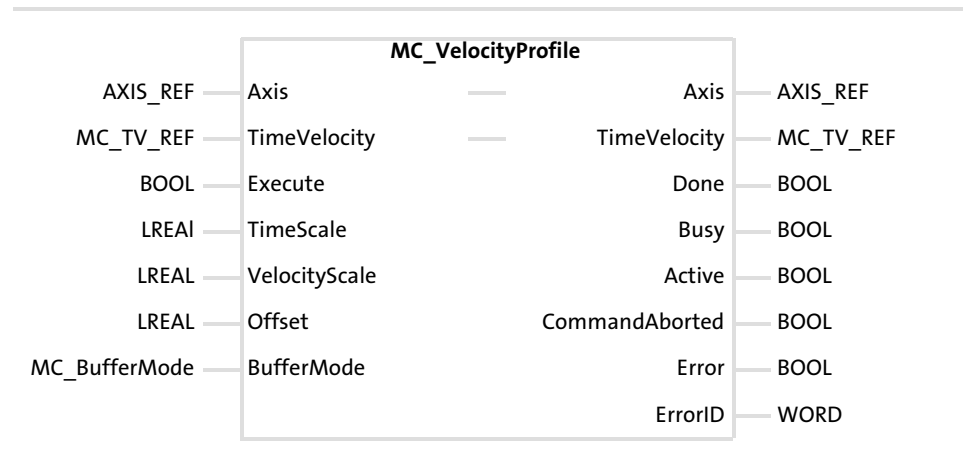
TimeScale	1.0
PositionScale	1.0
Offset	0.0
BufferMode	BM_ABORTING_E

Fehlerbeschreibung

ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
1	Achse befindet sich in einem für diese Bewegung unzulässigen Zustand.	Erlaubte Zustände: Standstill, Discrete Motion, Continuous Motion, Synchronized Motion
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
6	Reglerfreigabe fehlt	Mit MC_Power muss zunächst Reglerfreigabe gesetzt werden.
19	Diese Instanz führt gerade diese Bewegung für eine andere Achse aus.	
20	Unzulässiger Wert für BufferMode	
24	Für eine Achse von dieser Achsenart ist diese Bewegung nicht erlaubt.	Kontrolle MK_AXIS_TYPE
25	Negativer Softwareendschalter	
26	Positiver Softwareendschalter	
33	NoOfRows in TimePosition-Tabelle < 1	Die Tabelle muss mindestens eine Stütze beinhalten.
50	Eingang TimePosition gleich NULL	
53	PositionScale = 0	
54	TimeScale <= 0	
100	Achsfehler ist während der Ausführung des Befehls aufgetreten.	Achsfehler ist mit MC_ReadAxisError einzulesen.
101	Systemfehler ist während der Ausführung des Befehls aufgetreten.	Systemfehler ist mit MC_ReadSysError einzulesen.

4.2.9 Funktionsbaustein MC_VelocityProfile**Aufgabe**

Mit diesem Funktionsbaustein wird eine Positionieraufgabe in Form einer Tabelle, die die Geschwindigkeit einer Achse abhängig von der Zeit beschreibt, aktiviert.

Schematische KOP/FUP-Darstellung

Funktionsbeschreibung

Vordefinierte Datentypen in mc.lib:

```

TYPE MC_TV_REF :
    STRUCT
        NoOfRows:    DWORD;
        TV_TablePtr:  DWORD;
    END_STRUCT
END_TYPE
TYPE MC_TV_TR :
    STRUCT
        delta_time:   LREAL;
        velocity:     LREAL;
        interpolation:  MC_PROFILE_INT_ART;
    END_STRUCT
END_TYPE
    
```

NoOfRows: Anzahl von Stützstellen

TP_TablePtr: Adresse der Tabelle mit den Stützstellen. Die Tabelle muss in der Applikation als **ARRAY[0...NoOfRows-1] OF MC_TV_TR** deklariert sein.

delta_time: Die zeitliche Differenz zwischen zwei Positionen in [s].

Bei aktiver Bearbeitung können die Tabellenwerte geändert werden. Man kann auch zwischen verschiedenen Geschwindigkeitsprofil-Tabellen umschalten.

Für jedes Segment der Profil-Tabelle kann eine andere Interpolationsart eingegeben werden:

- 0 - linear
- 1 - kubischer Spline (stetige 1. Ableitung)
- 2 - Spline 5. Grades (Stetige 1. und 2. Ableitung)

Die Tabelle wird immer auf die aktuelle Position und die aktuelle Geschwindigkeit der Achse aufgesetzt. Die aktuelle Beschleunigung der Achse wird vernachlässigt. Bei der Erstellung der Tabelle muss darauf geachtet werden, dass kein Beschleunigungssprung beim Start der Bewegung aus der Tabelle entsteht.

Die in der Tabelle vorgegebenen Werte von **velocity** können mit den Eingangsparameter **VelocityScale** und **Offset** folgendermaßen beeinflusst werden:

$$\text{velocity}_{\text{korr}} = \text{velocity} * \text{VelocityScale} + \text{Offset}$$

Die in der Tabelle für jede Geschwindigkeit (**velocity**) vorgegebene Zeit (**delta_time**) kann mit dem Eingangsparameter **TimeScale** folgendermaßen beeinflusst werden:

$$\text{delta_time}_{\text{korr}} = \text{delta_time} * \text{TimeScale}$$

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Axis	AXIS_REF	P	Index einer Achse. 0 ... MaxAxis
TimeVelocity	MC_TV_REF	P	Referenz auf die Zeit/Geschwindigkeit-Tabelle
VAR_INPUT			
Execute	BOOL	P	Start der Bewegung mit steigender Flanke
TimeScale	LREAL	O	Skalierungsfaktor für die Zeit des Profils
VelocityScale	LREAL	O	Skalierungsfaktor für die Geschwindigkeit des Profils
Offset	LREAL	O	Verschiebung der Geschwindigkeit [u/s]
BufferMode	MC_BufferMode	O, B	Enum-Typ MC_BufferMode. TYPE MC_BufferMode : (BM_ABORTING_E :=0, BM_BUFFERED_E :=1, BM_BLENDING_LOW_E :=2, BM_BLENDING_PREVIOUS_E:=3, BM_BLENDING_NEXT_E :=4, BM_BLENDING_HIGH_E :=5, BM_MASTER_POSITION_E:=6); END_TYPE Bei BM_ABORTING_E wird die aktuelle Bewegung sofort abgelöst. Bei BM_BUFFERED_E wird diese Aufgabe erst gestartet, wenn die aktuelle beendet wird. Bei BM_BLENDING_LOW_E , BM_BLENDING_PREVIOUS_E , BM_BLENDING_NEXT_E , BM_BLENDING_HIGH_E wird der Übergang wie bei BM_BUFFERED_E ausgeführt.
VAR_OUTPUT			
Done	BOOL	-	Ausführung des Profils erfolgreich abgeschlossen.
Busy	BOOL	-	Wird mit steigender Flanke von Execute gesetzt.
Active	BOOL	-	Bewegung in Ausführung.
CommandAborted	BOOL	-	Diese Bewegung wurde durch einen anderen Bewegungsbefehl abgebrochen.
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Initialisierungswerte

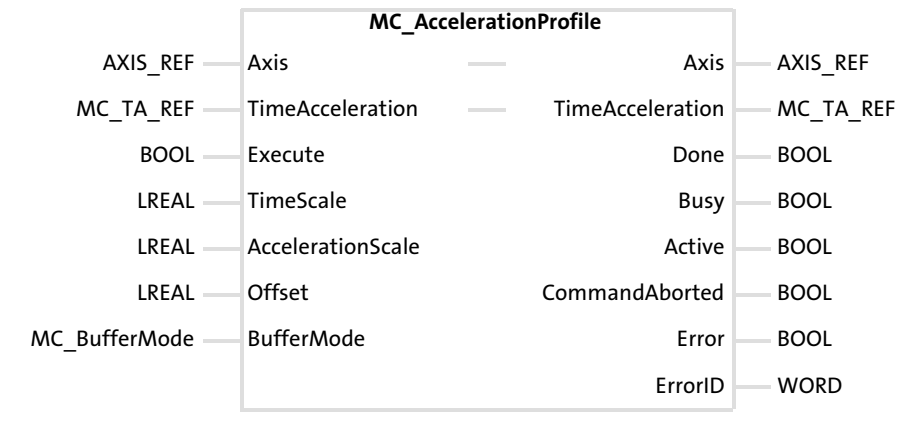
TimeScale	1.0
VelocityScale	1.0
Offset	0.0
BufferMode	BM_ABORTING_E

Fehlerbeschreibung

ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
1	Achse befindet sich in einem für diese Bewegung unzulässigen Zustand.	Erlaubte Zustände: Standstill, Discrete Motion, Continuous Motion, Synchronized Motion
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
6	Reglerfreigabe fehlt	Mit MC_Power muss zunächst Reglerfreigabe gesetzt werden.
19	Diese Instanz führt gerade diese Bewegung für eine andere Achse aus.	
20	Unzulässiger Wert für BufferMode	
24	Für eine Achse von dieser Achsenart ist diese Bewegung nicht erlaubt.	Das ist keine lagegeregelte Achse. (Vielleicht Geber?)
25	Negativer Softwareendschalter	
26	Positiver Softwareendschalter	
33	NoOfRows in TimePosition-Tabelle < 1	Die Tabelle muss mindestens eine Stütze beinhalten.
50	Eingang TimePosition gleich NULL	
54	VelocityScale = 0	
56	TimeScale <= 0	
100	Achsfehler ist während der Ausführung des Befehls aufgetreten.	Achsfehler ist mit MC_ReadAxisError einzulesen.
101	Systemfehler ist während der Ausführung des Befehls aufgetreten.	Systemfehler ist mit MC_ReadSysError einzulesen.

4.2.10 Funktionsbaustein MC_AccelerationProfile**Aufgabe**

Mit diesem Funktionsbaustein wird eine Positionieraufgabe in Form einer Tabelle, die die Beschleunigung einer Achse abhängig von der Zeit beschreibt, aktiviert.

**Schematische
KOP/FUP-Darstellung**

Funktionsbeschreibung

Vordefinierte Datentypen in mc.lib:

```

TYPE MC_TA_REF :
    STRUCT
        NoOfRows:    DWORD;
        TA_TablePtr:  DWORD;
    END_STRUCT
END_TYPE
TYPE MC_TA_TR :
    STRUCT
        delta_time:   LREAL;
        acceleration:  LREAL;
        interpolation:  MC_PROFILE_INT_ART;
    END_STRUCT
END_TYPE
    
```

NoOfRows: Anzahl von Stützstellen

TP_TablePtr: Adresse der Tabelle mit den Stützstellen. Die Tabelle muss in der Applikation als **ARRAY[0...NoOfRows-1] OF MC_TA_TR** deklariert sein.

delta_time: Die zeitliche Differenz zwischen zwei Positionen in [s].

Bei aktiver Bearbeitung können die Tabellenwerte geändert werden. Man kann auch zwischen verschiedenen Geschwindigkeitsprofil-Tabellen umschalten.

Für jedes Segment der Profil-Tabelle kann eine andere Interpolationsart eingegeben werden:

- 0 - linear
- 1 - kubischer Spline (stetige 1. Ableitung)
- 2 - Spline 5.Grades (stetige 1. und 2. Ableitung)

Die Tabelle wird immer auf die aktuelle Position und die aktuelle Geschwindigkeit der Achse aufgesetzt. Die aktuelle Beschleunigung der Achse wird vernachlässigt. Bei der Erstellung der Tabelle muss darauf geachtet werden, dass kein Beschleunigungssprung beim Start der Bewegung aus der Tabelle entsteht.

Die in der Tabelle vorgegebenen Werte von **acceleration** können mit den Eingangsparameter **Accelerati-onScale** und **Offset** folgendermaßen beeinflusst werden:

$$\text{accelerationkorr} = \text{acceleration} * \text{AccelerationScale} + \text{Offset}$$

Die in der Tabelle für jede Beschleunigung (**acceleration**) vorgegebene Zeit (**delta_time**) kann mit dem Eingangsparameter **TimeScale** folgendermaßen beeinflusst werden:

$$\text{delta_timekorr} = \text{delta_time} * \text{TimeScale}$$

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Axis	AXIS_REF	P	Index einer Achse. 0 ... MaxAxis
TimeAcceleration	MC_TA_REF	P	Referenz auf die Zeit/Beschleunigungs-Tabelle
VAR_INPUT			
Execute	BOOL	P	Start der Bewegung mit steigender Flanke
TimeScale	LREAL	O	Skalierungsfaktor für die Beschleunigung des Profils
AccelerationScale	LREAL	O	Skalierungsfaktor für die Beschleunigung des Profils
Offset	LREAL	O	Verschiebung der Beschleunigung [u/s]
BufferMode	MC_BufferMode	O	Enum-Typ MC_BufferMode. TYPE MC_BufferMode : (BM_ABORTING_E :=0, BM_BUFFERED_E :=1, BM_BLENDING_LOW_E :=2, BM_BLENDING_PREVIOUS_E :=3, BM_BLENDING_NEXT_E :=4, BM_BLENDING_HIGH_E :=5, BM_MASTER_POSITION_E :=6); END_TYPE Bei BM_ABORTING_E wird die aktuelle Bewegung sofort abgelöst. Bei BM_BUFFERED_E wird diese Aufgabe erst gestartet, wenn die aktuelle beendet wird. Bei BM_BLENDING_LOW_E , BM_BLENDING_PREVIOUS_E , BM_BLENDING_NEXT_E , BM_BLENDING_HIGH_E wird der Übergang wie bei BM_BUFFERED_E ausgeführt.
VAR_OUTPUT			
Done	BOOL	-	Ausführung des Profils erfolgreich abgeschlossen.
Busy	BOOL	-	Wird mit steigender Flanke von Execute gesetzt.
Active	BOOL	-	Bewegung in Ausführung.
CommandAborted	BOOL	-	Diese Bewegung wurde durch einen anderen Bewegungsbefehl abgebrochen.
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode.

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Initialisierungswerte

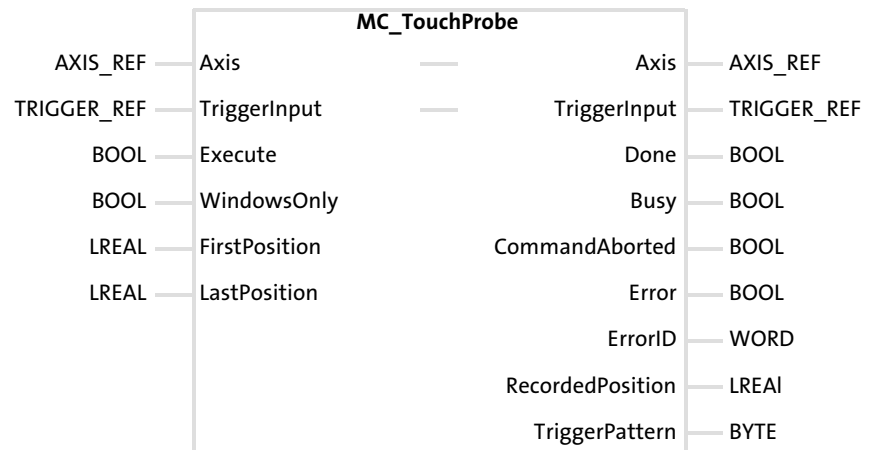
TimeScale	1.0
AccelerationScale	1.0
Offset	0.0
BufferMode	BM_ABORTING_E

Fehlerbeschreibung

ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
1	Achse befindet sich in einem für diese Bewegung unzulässigen Zustand.	Erlaubte Zustände: Standstill, Discrete Motion, Continuous Motion, Synchronized Motion
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
6	Reglerfreigabe fehlt	Mit MC_Power muss zunächst Reglerfreigabe gesetzt werden.
19	Diese Instanz führt gerade diese Bewegung für eine andere Achse aus.	
20	Unzulässiger Wert für BufferMode	
24	Für eine Achse von dieser Achsenart ist diese Bewegung nicht erlaubt.	Das ist keine lagegeregelte Achse. (Vielleicht Geber?)
25	Negativer Softwareendschalter	
26	Positiver Softwareendschalter	
33	NoOfRows in TimePosition-Tabelle < 1	Die Tabelle muss mindestens eine Stütze beinhalten.
50	Eingang TimePosition gleich NULL	
55	AccelerationScale = 0	
56	TimeScale <= 0	
100	Achsfehler ist während der Ausführung des Befehls aufgetreten.	Achsfehler ist mit MC_ReadAxisError einzulesen.
101	Systemfehler ist während der Ausführung des Befehls aufgetreten.	Systemfehler ist mit MC_ReadSysError einzulesen.

4.2.11 Funktionsbaustein MC_TouchProbe**Aufgabe**

Mit diesem Funktionsbaustein kann die Position einer Achse im Interpolator bei einem Trigger-Ereignis aufgenommen werden. Diese Funktion kann nur für schnelle Eingänge des Interpolators benutzt werden.

**Schematische
KOP/FUP-Darstellung**

Funktionsbeschreibung

Diese Funktion kann nur für schnelle Eingänge des Interpolators benutzt werden. Ein Teil von den digitalen Eingängen aus den Prozessdaten kann dem Interpolator als **schnelle Eingänge** dem Interpolator zur Verfügung gestellt werden. Die Lage von schnellen Eingängen sowie auch die Freigabe über entsprechende Maske erfolgt mit dem Funktionsbaustein **MC_DefineQuickInputs**. Vordefinierte Datentypen in mc.lib:

```

TYPE TRIGGER_REF :
STRUCT
    Inputs                : INPUT_REF; // Referenz auf Eingänge:
                                // 0: schnelle Eingänge des Interpolators:
                                // Modul 0
                                // 1: schnelle Eingänge des Interpolators:
                                // Modul 1
                                // 2: Antriebsmodul mit CAN-Bus
                                // 3 Antriebsmodul mit analoger Schnittstelle
                                // (LBM-ARI)

    InputNumber           : BYTE;      // Nummer von digitalen Triggereingang (0-15)
    TriggerNumber         : BYTE;      // Triggernummer (0-2)
    TriggerPattern        : BYTE;      // 0-steigende Flanke; 1-fallende Flanke;
                                // 2-beide

    TriggerObjectDataIdx  : WORD;      // Codestelle 6001
    TriggerDoneObjectDataIdx : WORD;    // Codestelle 6002
    RecPosObjectDataIdx   : WORD;      // Codestelle 6000
    AxisPeriodeObjectDataIdx : WORD;    // noch nicht implementiert
    FirstPosObjectDataIdx : WORD;      // noch nicht implementiert
    LastPosObjectDataIdx  : WORD;      // noch nicht implementiert
END_STRUCT
END_TYPE
    
```

Bei der Anwendung von LBM-ARI (Inputs=3) ist der Triggereingang fest (digitaler Eingang I4 vom Modul). Der Wert am Eingangsparameter **TriggerNumber** hat keine Bedeutung. Die Strukturdaten **TriggerObjectDataIdx**, **TriggerDoneObjectDataIdx**, **RecPosObjectDataIdx** sind für **MC_TouchProbe** nicht relevant.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Axis	AXIS_REF	P	Index einer Achse. 0 ... MaxAxis
TriggerInput	TRIGGER_REF	P	Referenz auf Triggerbeschreibung
VAR_INPUT			
Execute	BOOL	P	Mit steigender Flanke wird der Trigger aktiviert.
WindowsOnly	BOOL	O	Triggerereignis nur innerhalb vom vorgegebenen Fenster akzeptieren.
FirstPosition	LREAL	O	Startposition des Fensters, ab welcher (inclusive) das Triggerereignis akzeptiert werden soll in [u].
LastPosition	LREAL	P	Endposition des Fensters, bis welche (inclusive) das Triggerereignis akzeptiert werden soll in [u].
VAR_OUTPUT			
Done	BOOL	-	TRUE zeigt, dass die Position aufgenommen ist. Der gültige Wert ist über den Ausgangsparameter RecordedPosition geliefert.
Busy	BOOL	-	Funktion in Ausführung.
CommandAborted	BOOL	-	TRUE zeigt, dass die Ausführung der Funktion durch MC_AbortTrigger abgebrochen wurde.
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn ein Fehler bei der Ausführung dieses Befehls aufgetreten ist.
ErrorID	WORD	-	Fehlercode
RecordedPosition	LREAL	-	Aufgenommene Position
TriggerPattern	Byte	-	Triggerpattern bei dem die Position aufgenommen ist. 0 – steigende Flanke, 1 – fallende Flanke

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Initialisierungswerte

WindowsOnly	FALSE
FirstPosition	0

Fehlerbeschreibung

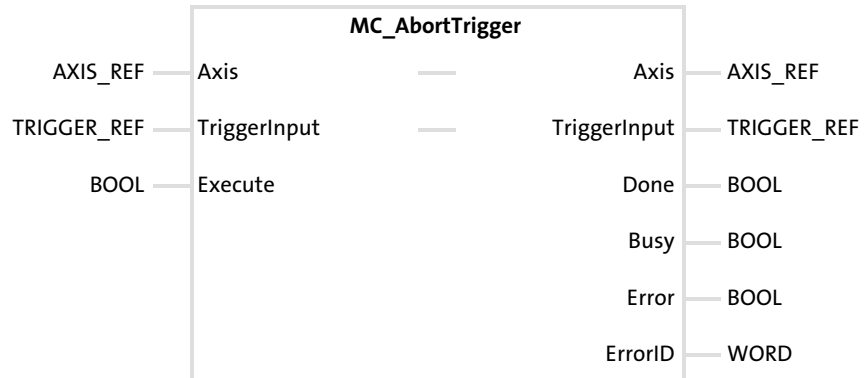
ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
1	Unzulässiger Zustand der Achse	Achse noch nicht initialisiert oder im Fehlerzustand.
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
19	Diese Instanz ist für eine andere Achse aktiv	
1064	Unzulässiger Wert für TriggerNumber	TriggerNumber > 2
1066	Unzulässiger Wert für TriggerInput	Inputs > 3
1067	Unzulässiger Wert für TriggerPatter	TriggerPattern > 2

4.2.12 Funktionsbaustein MC_Abort_Trigger

Aufgabe

Mit diesem Funktionsbaustein kann der mit dem Funktionsbaustein MC_TouchProbe aktivierte Trigger deaktiviert werden. Diese Funktion kann nur für Servoantriebe der Firma Lenze benutzt werden, die über CAN-Drive-bus an die Steuerung angeschlossen sind.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Vordefinierte Datentypen in mc.lib:

```

TYPE TRIGGER_REF :
STRUCT
    Input                : INPUT_REF; // Referenz auf Eingänge:
                                // 0: schnelle Eingänge des Interpolators:
                                // Modul 0
                                // 1: schnelle Eingänge des Interpolators:
                                // Modul 1
                                // 2: Antriebsmodul mit CAN-Bus
                                // 3 Antriebsmodul mit analoger Schnittstelle
                                // (LBM-ARI)

    InputNumber          : BYTE;      // Nummer von digitalen Triggereingang
    TriggerNumber        : BYTE;      // Triggernummer (0-2)
    TriggerPattern        : BYTE;      // 0-steigende Flanke; 1-fallende Flanke;
                                        2-beide

    TriggerObjectDataIdx : WORD;       // Codestelle 6001
    TriggerDoneObjectDataIdx : WORD;    // Codestelle 6002
    RecPosObjectDataIdx   : WORD;       // Codestelle 6000
    AxisPeriodeObjectDataIdx : WORD;    // noch nicht implementiert
    FirstPosObjectDataIdx : WORD;       // noch nicht implementiert
    LastPosObjectDataIdx  : WORD;       // noch nicht implementiert
END_STRUCT
END_TYPE
    
```

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Axis	AXIS_REF	P	Index einer Achse. 0 ... MaxAxis
TriggerInput	TRIGGER_REF	P	Referenz auf Triggerbeschreibung
VAR_INPUT			
Execute	BOOL	P	Mit steigender Flanke wird die aktive Touch Probe Funktion deaktiviert.
VAR_OUTPUT			
Done	BOOL	-	TRUE zeigt, dass die Touch Probe deaktiviert wurde.
Busy	BOOL	-	Funktion in Ausführung
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn ein Fehler bei der Ausführung dieses Befehls aufgetreten ist.
ErrorID	WORD	-	Fehlercode

1) P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Fehlerbeschreibung

ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
1	Unzulässiger Zustand der Achse	Achse noch nicht initialisiert oder im Fehlerzustand.
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
19	Diese Instanz ist für eine andere Achse aktiv	
1050	Unzulässiger Wert für TriggerInput	Inputs > 3
1064	Unzulässiger Wert für TriggerNumber	TriggerNumber > 2

4.2.13 Funktionsbaustein MC_TorqueControl



Stop!

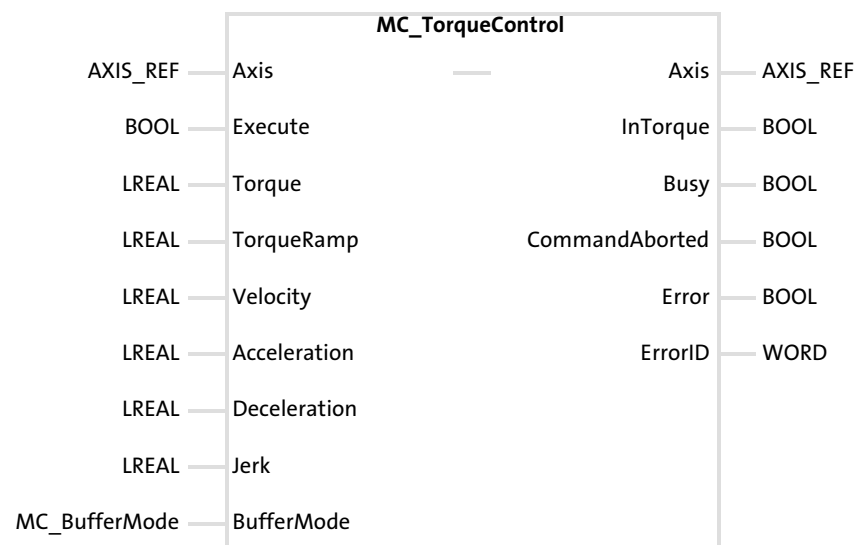
Dieser Funktionsbaustein setzt voraus, dass die CANopen-Antriebe die Regelung des Drehmoments über CANopen unterstützen.

Bei Achsen mit analoger Schnittstelle ist der FB nicht verwendbar.

Aufgabe

Mit diesem Funktionsbaustein kann das vorgegebene Drehmoment mit der vorgegebenen Rampe aufgebaut werden. Für die Begrenzung der Drehzahl wird vorgegebene Grenzgeschwindigkeit mit vorgegebenen Parametern wie Beschleunigung, Verzögerung und Ruck aufgebaut.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung



Hinweis!

Der Funktionsbaustein ist auch bei linearen Motoren anwendbar, wobei es sich in dem Fall nicht um das Drehmoment in [Nm] sondern um die Kraft in [N] handelt.

Die Drehmomentregelung ist als eine Bewegungsart aus der Gruppe **Continuous Motion** zu sehen. Beim Aktivieren der Drehmomentregelung (steigende Flanke von **Execute**) wird jede laufende Bewegung abgebrochen und die Achse in den Zustand **ContinuousMotion** transferiert.

Das Zieldrehmoment wird mit vorgegebener Rampe vom aktuellen Ist-Drehmoment aus aufgebaut. Das Antriebsmodul soll das in jedem Interpolationszyklus vorgegebene Drehmoment als Soll-Drehmoment betrachten. Die tatsächliche Drehmomentregelung muss im Antriebsmodul stattfinden.

Die mit dem Funktionsbaustein vorgegebene Geschwindigkeit (Eingangsparameter **Velocity**) wird von aktueller Geschwindigkeit der Achse aus mit vorgegebenen Parameter: **Acceleration**, **Deceleration** und **Jerk** aufgebaut. Das Antriebsmodul soll die in jedem Interpolationszyklus vorgegebene Geschwindigkeit als Drehzahlbegrenzung bei aktiver Drehmomentregelung betrachten. Die Vorgehensweise in diesem Fall ist antriebsmodulspezifisch, es kann z. B. bei Überschreiten des Solldrehmoments sofort auf Drehzahlregelung umgeschaltet werden usw.

Diese Bewegung wird, wie jede andere Bewegung im Zustand **Continuous-Motion**, mit **MC_Stop** angehalten oder durch eine andere Bewegungsart abgebrochen.

CAN-OPEN nach DSP 402: Hier muss noch eine Vereinbarung mit Antriebshersteller getroffen werden.

Wahrscheinlich muss im Antrieb **Profile Torque Mode** implementiert werden, Konfiguration mit vorgegebenen Daten über SDO's und Ein-/Ausschalten über Steuerword erfolgen.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Axis	AXIS_REF	P	Index einer Achse. 0 ... MaxAxis
VAR_INPUT			
Execute	BOOL	P	Mit steigender Flanke wird die Funktion gestartet.
Torque	LREAL	O	Das Zieldrehmoment in [Nm] oder Kraft in [N]
TorqueRamp	LREAL	O	Die Rampe (1. Ableitung vom Drehmoment), mit der das Zieldrehmoment aufgebaut wird in [Nm/s] oder in [N/s].
Velocity	LREAL	O	Maximale Geschwindigkeit für Drehzahlbegrenzung in [u/s]
Acceleration	LREAL	O	Maximale Beschleunigung beim Aufbauen der maximalen Geschwindigkeit in [u/s ²]
Deceleration	LREAL	O	Maximale Verzögerung beim Aufbauen der maximalen Geschwindigkeit [u/s ²]
Jerk	LREAL	O	Maximaler Ruck beim Aufbauen der maximalen Geschwindigkeit in [u/s ³]
BufferMode	MC_BufferMode	O	Enum-Typ MC_BufferMode. TYPE MC_BufferMode : (BM_ABORTING_E :=0, BM_BUFFERED_E :=1, BM_BLENDING_LOW_E :=2, BM_BLENDING_PREVIOUS_E :=3, BM_BLENDING_NEXT_E :=4, BM_BLENDING_HIGH_E :=5, BM_MASTER_POSITION_E :=6); END_TYPE Bei BM_ABORTING_E wird die aktuelle Bewegung sofort abgelöst. Bei BM_BUFFERED_E wird diese Aufgabe erst gestartet, wenn die aktuelle beendet wird. Bei BM_BLENDING_LOW_E , BM_BLENDING_PREVIOUS_E , BM_BLENDING_NEXT_E , BM_BLENDING_HIGH_E wird der Übergang wie bei BM_BUFFERED_E ausgeführt.
VAR_OUTPUT			
InTorque	BOOL	-	TRUE, wenn das vorgegebene Drehmoment erreicht wird.
Busy	BOOL	-	Funktionsbaustein ist aktiv.
CommandAborted	BOOL	-	Die Aufgabe wurde durch eine andere Bewegungsaufgabe für diese Achse abgebrochen.
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Initialisierungswerte

Torque	1
TorqueRamp	10
Velocity	50
Acceleration	1000
Deceleration	1000
Jerk	10000
BufferMode	BM_ABORTING_E

Fehlerbeschreibung

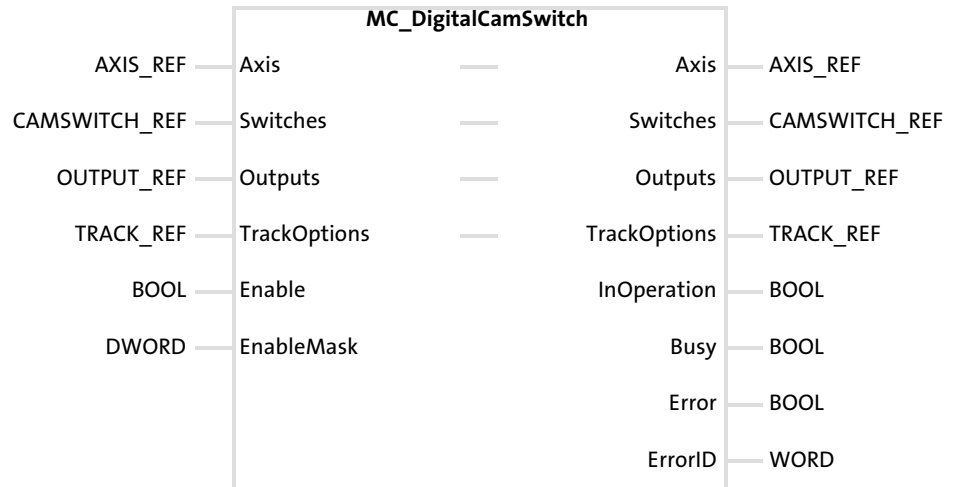
ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
6	Reglerfreigabe fehlt	Mit MC_Power muss zunächst Reglerfreigabe gesetzt werden.
7	Velocity zu groß oder < 0	Velocity > MAX_VELOCITY_APPL  206
8	Acceleration zu groß oder <= 0	Acceleration > MAX_ACCELERATION_APPL  206
9	Deceleration zu groß oder <= 0	Deceleration > MAX_DECELERATION_APPL  206
10	Jerk zu groß oder <= 0	Jerk > MAX_JERK_APPL  206
19	Diese Instanz ist für eine andere Achse aktiv.	
20	Unzulässiger Wert für BufferMode	
24	Für eine Achse von dieser Achsenart ist diese Bewegung nicht erlaubt.	Das ist keine lagegeregelte Achse. (Vielleicht Geber?)
100	Achsfehler ist während der Ausführung des Befehls aufgetreten.	Achsfehler ist mit MC_ReadAxisError einzulesen.
101	Systemfehler ist während der Ausführung des Befehls aufgetreten.	Systemfehler ist mit MC_ReadSysError einzulesen.

4.2.14 Funktionsbaustein MC_DigitalCamSwitch

Aufgabe

Mit diesem Funktionsbaustein können digitale Ausgänge, abhängig von den vorgegebenen Bedingungen in Form von Achspositionen, geschaltet werden. Es können mehrere Schaltbedingungen für einen digitalen Ausgang vorgegeben werden. Ein Satz von Schaltbedingungen für einen digitalen Ausgang wird weiter als eine Spur genannt.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Nur eine Instanz von **MC_DigitalCamSwitch** kann pro Achse aktiv sein. Wenn die schnellen Ausgänge als Outputs benutzt werden, müssen diese vorher mit **MC_DefineQuickOutputs** freigegeben werden.

Für eine Achse können bis 32 Nocken (Switches) aktiviert werden. Jeder Nocken bezieht sich auf einen schnellen Ausgang (eine Spur/Track). Das heißt, verschiedene Switches können auf einen Ausgang wirken.

Mit **EnableMask** wird eine Spur (Ausgang) aktiviert/deaktiviert, dass alle Switches, die auf diesen Ausgang wirken, damit betroffen werden. Alle programmierten Daten werden nur bei positiver Flanke von **Enable** übernommen, mit Ausnahme von **EnableMask**, deren Änderung sofort wirksam ist.

Vordefinierte Datentypen in mc.lib:

```
TYPE
    OUTPUT_REF : WORD;
END_TYPE
```

OUTPUT_REF: Ort der Ausgänge:
 0 - schnelle Ausgänge des Moduls 0
 1 - schnelle Ausgänge des Moduls 1

```
TYPE CAMSWITCH_REF :
    STRUCT
        NoOfSwitches:    BYTE;
        CamSwitchPtr:    DWORD;
    END_STRUCT
END_TYPE
```

NoOfSwitches: Anzahl der Schaltbedingungen

CamSwitchPtr: Adresse der Tabelle mit Schaltbedingungen; die Tabelle muss in der Applikation als ARRAY[0...NoOfSwitches-1] OF CAMSWITCH_TR deklariert werden.

```

TYPE CAMSWITCH_TR :
  STRUCT
    TrackNumber:      INT;
    FirstOnPosition:  LREAL;
    LastOnPosition:   LREAL;
    AxisDirection:    INT;
    CamSwitchMode:    INT;
    Duration:         TIME;
  END_STRUCT
END_TYPE

```

TrackNumber: Index einer Spur (Ausgang). Verschiedene Schaltbedingungen können den gleichen Index der Spur haben, d. h. sie können sich auf den gleichen Ausgang beziehen (0 ... 15 = schnelle Ausgänge des Moduls 0; 16 ... 31 = schnelle Ausgänge des Moduls 1)

FirstOnPosition: Der kleinste Wert des Bereiches, in dem der Ausgang auf TRUE gesetzt wird

LastOnPosition: Der größte Wert des Bereiches, in dem der Ausgang auf TRUE gesetzt wird

AxisDirection: Erforderliche Bewegungsrichtung der Achse für die Aktivierung der Schaltbedingungen (0 = beide Richtungen, 1 = positive Richtung, 2 = negative Richtung)

CamSwitchMode: Einschaltdauer (0 = auf Positionsbasis, 1 = auf Zeitbasis)

Duration: Einschaltdauer bei CamSwitchMode = 1

```

TYPE Track_REF :
  ARRAY[0..31] OF TRACK_TR;
END_TYPE

```

```

TYPE Track_TR :
  OnCompensation:    TIME;
  OnCompensation-    INT;
  Mode:
  OffCompensation:   TIME;
  Hysteresis:        LREAL;
END_TYPE

```

OnCompensation: Die Zeit des vorzeitigen (bei negativem Wert) oder verzögerten (bei positivem Wert) Einschalten des Ausgangs vor oder nach Erreichen der Umschaltposition.

OnCompensationMode:

Wert 0 - Einschalten vorzeitig

Wert 1 - Einschalten verzögert

OffCompensation: Die Zeit des vorzeitigen (bei negativem Wert) oder verzögerten (bei positivem Wert) Ausschalten des Ausgangs vor oder nach Erreichen der Umschaltposition (0 = Ausschalten vorzeitig; 1 = Ausschalten verzögert)

Hysteresis: Ein Abstand in [u] zu der Umschaltposition auf dem keine Umschaltung stattfindet, obwohl die Schaltbedingungen erfüllt sind, bis der Bereich verlassen wird. Auf diese Weise wird vermieden, dass die Umschaltung mehrmals ausgeführt wird, wenn die Achse sich in der Nähe von der Umschaltposition befindet.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Axis	AXIS_REF	P	Index einer Achse. 0 ... MaxAxis
Switches	CAMSWITCH_REF	P	Schaltbedingungen
Outputs	OUTPUT_REF	P	Ort des digitalen Ausgangs: 0 - Achsmodul, 1 - schnelle Ausgänge des Moduls 0 2 - schnelle Ausgänge des Moduls 1 Zur Zeit nur schnelle Ausgänge des Interpolators möglich, d. h. Wert 0 und 1.
TrackOptions	TRACK_REF	P	Zusatzeigenschaften einer Spur
VAR_INPUT			
Enable	BOOL	P	Gibt das Umschalten von Ausgängen frei.
EnableMask	DWORD	O	Gibt eine Spur frei. Bitweise kodiert. Bit=1 – entsprechende Spur ist freigegeben.
VAR_OUTPUT			
InOperation	BOOL	-	Vorgegebene Spuren sind frei gegeben.
Busy	BOOL	-	Funktionsbaustein ist aktiv.
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Initialisierungswerte

EnableMask	16#FFFFFFFF
------------	-------------

Fehlerbeschreibung

ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
19	Diese Instanz ist für eine andere Achse aktiv.	
1070	Unzulässiger Wert in Switches für Track-Number	> 31
1071	Unzulässiger Wert in Switches für Axis-Direction	> 2
1072	Unzulässiger Wert in Switches für CamSwitchMode	> 1
1073	CamSwitch für diese Achse bereits aktiv	Nur eine Instanz von MC_DigitalCamSwitch kann pro Achse aktiv sein.
1074	Unzulässiger Wert für OnCompensation-Mode	> 1

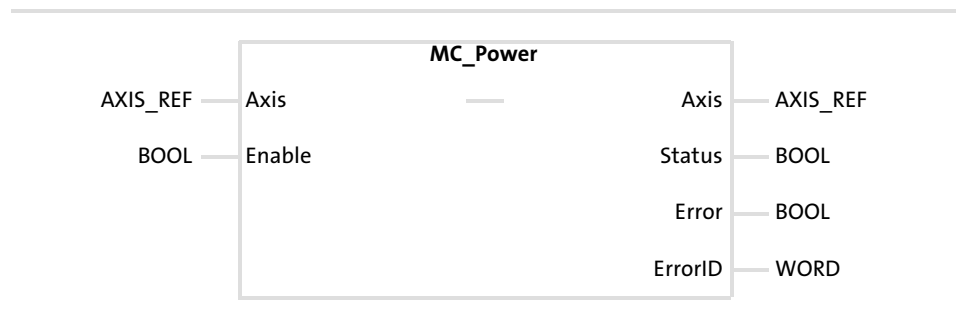
4.3 Administrative Funktionsbausteine nach PLCopen-Standard

4.3.1 Funktionsbaustein MC_Power

Aufgabe

Mit diesem Funktionsbaustein werden alle Freigaben für eine Achse gesetzt oder zurückgesetzt.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Ist der Eingangsparameter **Enable** = **TRUE**, werden alle Freigaben der Achse gesetzt. Ist der Eingangsparameter **Enable** = **FALSE**, werden alle Freigaben der Achse zurück gesetzt.

Bei Positionierachsen wird der Laderegler freigegeben.

Bei Felddbusantrieben ändert sich der Ausgangsparameter **Status** erst, wenn am Eingangsparameter **Enable** der Zustand für mehrere Bausteinzyklen anliegt (z. B. Busverzögerung).

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Axis	AXIS_REF	P	Index einer Achse. 0 ... MaxAxis
VAR_INPUT			
Enable	BOOL	P	TRUE = Achsfreigaben setzen, FALSE = Achsfreigaben zurück setzen
VAR_OUTPUT			
Status	BOOL	-	TRUE = Achse ist freigegeben FALSE = Achse ist nicht freigegeben
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Fehlerbeschreibung

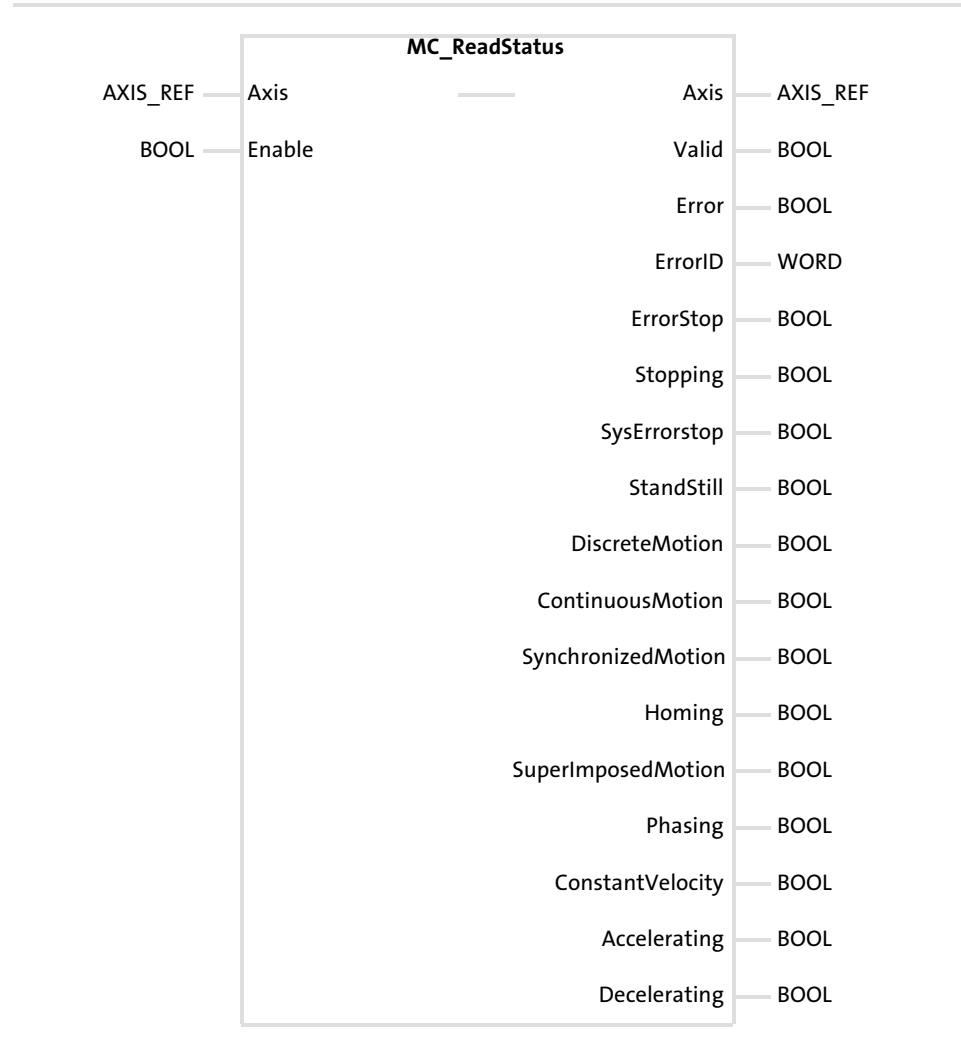
ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
1	Achse befindet sich nicht im Zustand Standstill .	
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
23	Für eine Achse von dieser Achsart ist die Freigabe nicht notwendig.	Keine geregelte Achse

4.3.2 Funktionsbaustein MC_ReadStatus

Aufgabe

Dieser Funktionsbaustein liefert den aktuellen Bewegungszustand einer Achse.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Es können gleichzeitig mehrere Ausgänge gesetzt werden, z. B.: DiscretMotion, SuperImposedMotion und Accelerating.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Axis	AXIS_REF	P	Index einer Achse. 0 ... MaxAxis
VAR_INPUT			
Enable	BOOL	P	Solange TRUE ist, wird der aktuelle Bewegungszustand kontinuierlich ausgegeben.
VAR_OUTPUT			
Valid	BOOL	-	TRUE = ein gültiger Ausgang steht an
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode
Errorstop	BOOL	-	TRUE = Achse im Zustand Errorstop
Stopping	BOOL	-	TRUE = Achse im Zustand Stopping
Disable	BOOL	-	TRUE = Freigabe für die Achse zurückgesetzt
SysErrorStop	BOOL	-	TRUE = Achse im Zustand SysErrorStop
StandStill	BOOL	-	TRUE = Achse im Zustand StandStill
DiscretMotion	BOOL	-	TRUE = Achse im Zustand DiscretMotion
ContinuousMotion	BOOL	-	TRUE = Achse im Zustand ContinuousMotion
SynchronizedMotion	BOOL	-	TRUE = Achse im Zustand SynchronizedMotion
Homing	BOOL	-	TRUE = Achse im Zustand Homing
SuperImposedMotion	BOOL	-	TRUE = Überlagerte Bewegung der Achse aktiv
Phasing	BOOL	-	TRUE = MC_Phasing aktiv
ConstantVelocity	BOOL	-	TRUE = Achse fährt mit konstanter Geschwindigkeit
Accelerating	BOOL	-	TRUE = Achse beschleunigt
Decelerating	BOOL	-	TRUE = Achse bremst

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Fehlerbeschreibung

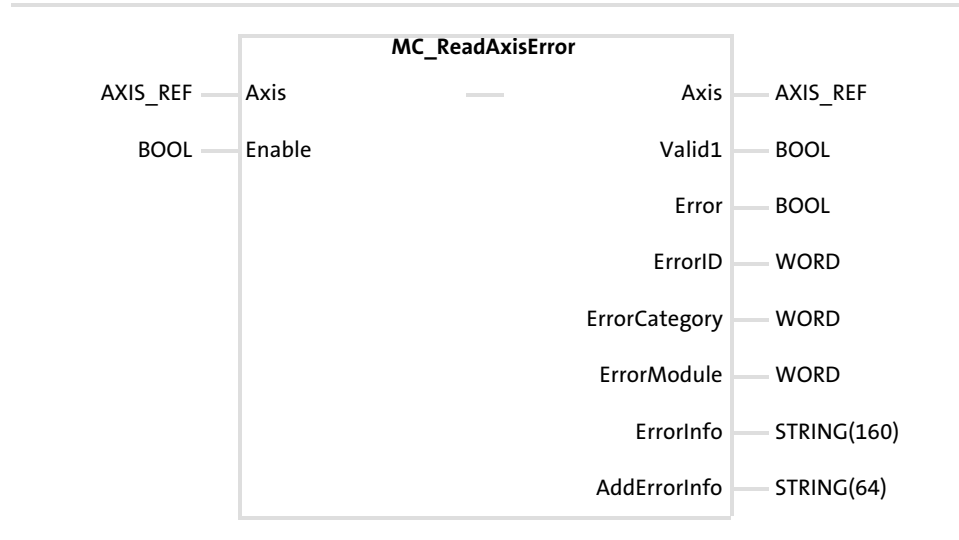
ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.

4.3.3 Funktionsbaustein MC_ReadAxisError

Aufgabe

Mit diesem Funktionsbaustein wird ein Achsfehler, der im Antrieb aufgetreten ist, angezeigt.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Der Baustein zeigt über die Ausgangsparameter **Error** und **ErrorId** sowohl die Fehler von Antrieben als auch die Fehler vom Baustein selbst.

Fehler Klasse 1 und 2 sind als Hinweise oder Warnungen zu betrachten. Sie führen nicht zum Abbruch der laufenden Bewegung und Übergang der Achse in den Zustand **Errorstop**. Sie müssen also nicht quittiert werden und werden nur über einen SPS-Zyklus angezeigt.

Bei Fehlern Klasse 3 und 4 wird die betroffene Achse sofort angehalten und geht in den Zustand **Errorstop**. Diese Fehler werden solange angezeigt bis sie mit **MC_Reset** quittiert werden.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Axis	AXIS_REF	P	Index einer Achse. 0 ... MaxAxis
VAR_INPUT			
Enable	BOOL	P	TRUE = Achsfehler anzeigen
VAR_OUTPUT			
Valid	BOOL	-	TRUE = Funktion ausgeführt
Error	BOOL	-	TRUE = über den Ausgang ErroID wird ein Fehlercode angezeigt. FALSE = kein Fehler wird angezeigt
ErrorID	WORD	-	Fehlercode
ErrorCategory	WORD	-	Fehlerklasse: 1: Lokal leicht 2: Lokal schwer 3: Global leicht 4: Global schwer
ErrorModule	WORD	-	Herstellerspezifisches Antriebsmodul: 19 – Lenze
ErrorInfo	STRING(160)	-	Fehlertext aus der Datei ncrspch0.txt (oder aus einer anderen sprachabhängigen Datei, die sich in Flash-Disk befindet).
AddErrorInfo	STRING(64)	-	Fehlerabhängige Zusatzinformation

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Fehlerbeschreibung

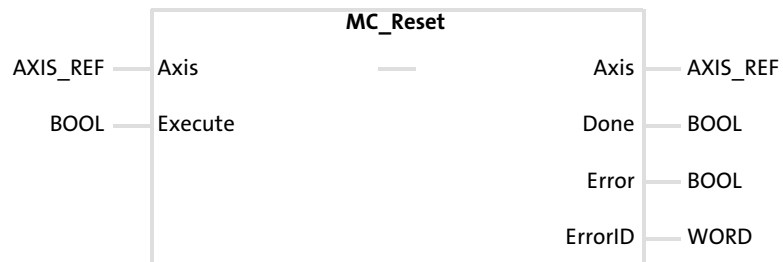
ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.

4.3.4 Funktionsbaustein MC_Reset

Aufgabe

Mit diesem Funktionsbaustein wird die anstehende Fehlermeldung quittiert und der Übergang der Achse aus dem Zustand **ErrorStop** in den Zustand **Standstill** durchgeführt, wenn keine weiteren Fehlermeldungen anstehen.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Alle Achsfehler der Klasse 3 oder 4, die zum Abbruch einer laufenden Bewegung und dem Übergang einer Achse in den Zustand **ErrorStop** geführt haben, müssen mit dem **MC_Reset** Funktionsbaustein quittiert werden. Mit der Quittung des letzten Fehlers wird die Achse in den Zustand **Standstill** übergehen.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Axis	AXIS_REF	P	Index einer Achse. 0 ... MaxAxis
VAR_INPUT			
Execute	BOOL	P	Mit positiver Flanke wird der Fehler quittiert.
VAR_OUTPUT			
Done	BOOL	-	TRUE = Zustand StandStill wurde erreicht FALSE = Achse bleibt im Zustand ErrorStop , weitere Fehler stehen an.
Error	BOOL	-	TRUE, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Fehlerbeschreibung

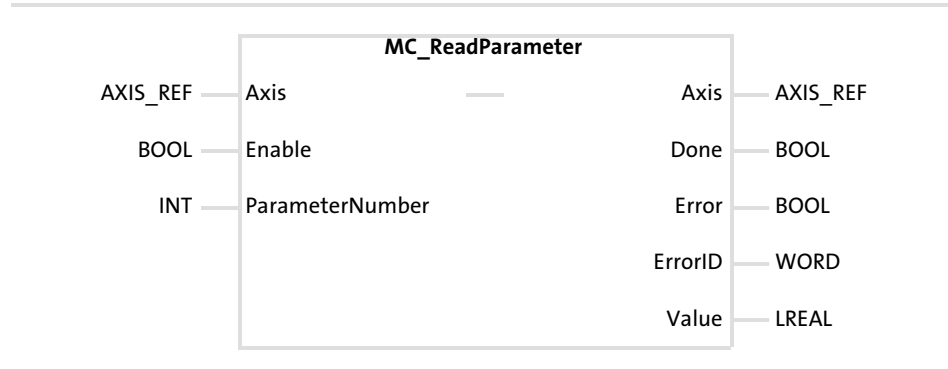
ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
1	Achse befindet sich nicht im Zustand Errorstop .	
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
100	Achsfehler ist während der Ausführung des Befehls aufgetreten.	Achsfehler ist mit MC_ReadAxisError einzulesen.
101	Systemfehler ist während der Ausführung des Befehls aufgetreten.	Systemfehler ist mit MC_ReadSysError einzulesen.

4.3.5 Funktionsbaustein MC_ReadParameter

Aufgabe

Mit diesem Funktionsbaustein können die definierten Achsparameter (Datentyp: LREAL) gelesen werden.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Solange der Eingangsparameter **Enable** = TRUE, wird der Ausgangsparameter **Value** mit dem aktuellen Parameterwert zyklisch versorgt. Bei **Enable** = FALSE wird der Ausgangsparameter **Value** auf Null gesetzt.

Die Werte aller nicht LREAL-Parameter werden für die Ausgabe in das LREAL-Format konvertiert.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Axis	AXIS_REF	P	Index einer Achse. 0 ... MaxAxis
VAR_INPUT			
Enable	BOOL	P	TRUE = Parameterwert wird kontinuierlich ausgegeben.
ParameterNumber	INT	O	Parameternummer (EN 206)
VAR_OUTPUT			
Done	BOOL	-	TRUE = Parameterwert wurde über den Ausgang „Value“ ausgegeben.
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode
Value	LREAL	-	Wert des spezifizierten Parameters

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Initialisierungswerte

ParameterNumber - 1 (EN 206)

Fehlerbeschreibung

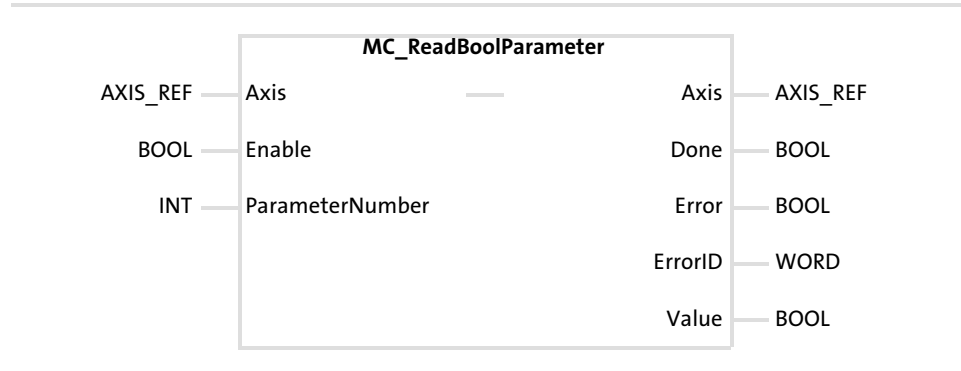
ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
22	Unzulässiger Wert von ParameterNumber	

4.3.6 Funktionsbaustein MC_ReadBoolParameter

Aufgabe

Mit diesem Funktionsbaustein können die definierten Achsparameter von Datentyp BOOL gelesen werden.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Solange der Eingangsparameter **Enable** = TRUE, wird der Ausgangsparameter **Value** mit dem aktuellen Parameterwert zyklisch versorgt.

Bei **Enable** = FALSE wird der Ausgangsparameter **Value** auf FALSE gesetzt.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Axis	AXIS_REF	P	Index einer Achse. 0 ... MaxAxis
VAR_INPUT			
Enable	BOOL	P	TRUE = Parameterwert wird kontinuierlich ausgegeben
ParameterNumber	INT	O	Parameter Nummer (☞ 206)
VAR_OUTPUT			
Done	BOOL	-	TRUE = Parameterwert wurde über den Ausgang "Value" ausgegeben
Error	BOOL	-	TRUE, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode
Value	BOOL	-	Wert des spezifizierten Parameters

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Initialisierungswerte

ParameterNumber - 1200 (☞ 206)

Fehlerbeschreibung

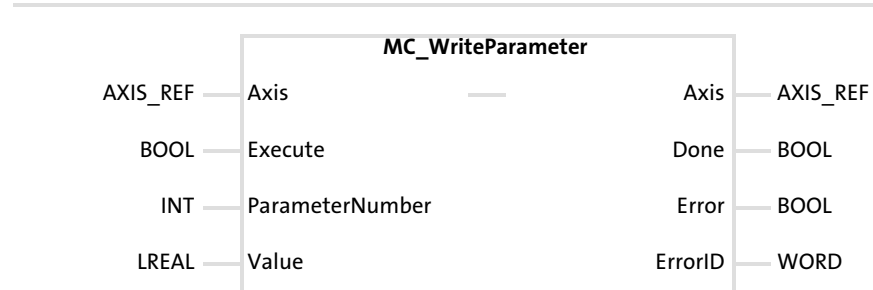
ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
22	Unzulässiger Wert von ParameterNumber .	

4.3.7 Funktionsbaustein MC_WriteParameter

Aufgabe

Mit diesem Funktionsbaustein können die definierten Achsparameter (Datentyp: LREAL) geschrieben werden.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Mit steigender Flanke am Eingang **Execute** wird der programmierte Parameter auf den Wert vom Eingang **Value** gesetzt. Danach wird der Ausgang **Done** auf TRUE gesetzt.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Axis	AXIS_REF	P	Index einer Achse. 0 ... MaxAxis
VAR_INPUT			
Execute	BOOL	P	Schreiben des Parameters mit steigender Flanke
ParameterNumber	INT	P	Parameternummer (EN 206)
Value	LREAL	P	Parameterwert
VAR_OUTPUT			
Done	BOOL	-	TRUE = Parameter ParameterNumber wurde geschrieben.
Error	BOOL	-	TRUE, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Fehlerbeschreibung

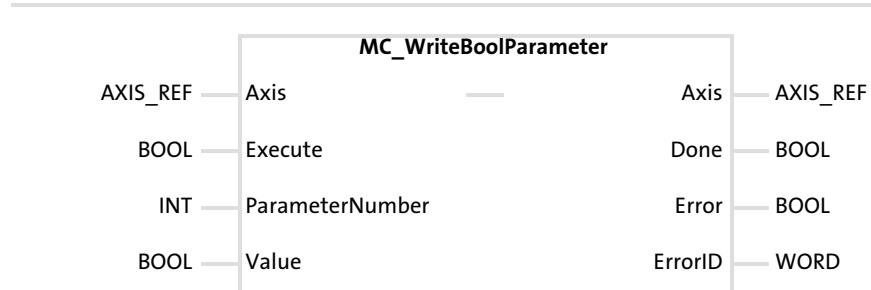
ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
22	Unzulässiger Wert von ParameterNumber .	

4.3.8 Funktionsbaustein MC_WriteBoolParameter

Aufgabe

Mit diesem Funktionsbaustein können die definierten Achsparameter vom Typ BOOL geschrieben werden.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Mit steigender Flanke am Eingang **Execute** wird das programmierte Parameter auf den Wert vom Eingang **Value** gesetzt. Danach wird der Ausgang **Done** auf TRUE gesetzt.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Axis	AXIS_REF	P	Index einer Achse. 0 ... MaxAxis
VAR_INPUT			
Execute	BOOL	P	Schreiben des Parameters mit steigender Flanke
ParameterNumber	INT	P	Parameternummer (EN 206)
Value	BOOL	P	Parameterwert
VAR_OUTPUT			
Done	BOOL	-	TRUE = Parameterwert wurde geschrieben
Error	BOOL	-	TRUE, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Fehlerbeschreibung

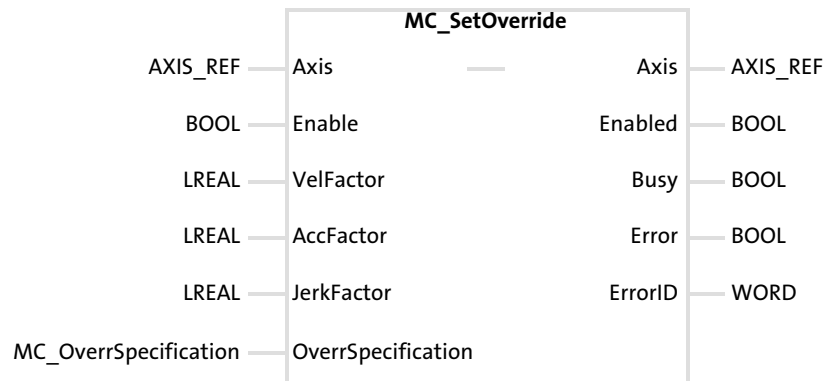
ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
22	Unzulässiger Wert von ParameterNumber	

4.3.9 Funktionsbaustein MC_SetOverride

Aufgabe

Dieser Funktionsbaustein setzt einen Multiplikationsfaktor für die Geschwindigkeit, die Beschleunigung/Verzögerung und den Ruck.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Der **VelFactor**-Parameter ist ein Faktor mit dem die mit anderen Funktionsbausteinen programmierte Geschwindigkeit multipliziert wird.

Der **AccFactor**-Parameter ist ein Faktor mit dem die mit anderen Funktionsbausteinen programmierte Beschleunigung oder Verzögerung multipliziert wird.

Der **JerkFactor**-Parameter ist ein Faktor mit dem der mit anderen Funktionsbausteinen programmierte Ruck multipliziert wird.

Als **VelFactor** kann ein Wert zwischen 0 und 1,2 programmiert werden.

Der Wert 0 für **VelFactor** hält eine Achse an, ohne sie in den Zustand **Standstill** zu transferieren. (Die Achse fährt weiter mit der Geschwindigkeit 0).

Die Werte für **AccFactor** und **JerkFactor** müssen größer als 0 und kleiner oder gleich 1,2 sein.

Der programmierte Overridefaktor bleibt solange erhalten, bis er mit einem neuen programmierten Wert überschrieben wird. Er wirkt nicht auf die Slave-Achsen (Achsen, die sich im Zustand **Synchronized Motion** befinden).

Der Funktionsbaustein hat keinen Einfluss auf das Zustandsdiagramm der Achse.

Overridefaktoren können jederzeit geändert werden und sie wirken sofort auf die laufende Bewegung, außer den Bremsphasen (Phasen, in denen eine Achse auf die programmierte Zielposition bremst).

Wenn einer der Eingangsparameter **VelFactor**, **AccFactor** oder **JerkFactor** geändert wird, wird sofort der Ausgangsparameter **Enabled** auf FALSE gesetzt und erst mit der Übernahme des Wertes durch den Interpolator wieder auf TRUE.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Axis	AXIS_REF	P	Index einer Achse 0 ... MaxAxis
VAR_INPUT			
Enable	BOOL	P	Bei TRUE wird der Overridefaktor ständig übernommen.
VelFactor	LREAL	O	Overridefaktor für die Geschwindigkeit
AccFactor	LREAL	O	Overridefaktor für die Beschleunigung und Verzögerung
JerkFactor	LREAL	O	Overridefaktor für den Ruck
OverrSpecification	MC_OverrSpecification	O	Enum-Typ MC_OverrSpecification: TYPE MC_OverrSpecification : (OVERR_SPEC_FOR_ALL := 0, OVERR_SPEC_FOR_POS := 1, OVERR_SPEC_FOR_SIMP := 2, OVERR_SPEC_FOR_PHAS := 3); END_TYPE OVERR_SPEC_FOR_ALL Override gilt für alle Bewegungen einschließlich für MC_MoveSuperImposed und MC_Phasing. OVERR_SPEC_FOR_POS Override gilt für alle Bewegungen außer Superimposed und Phasing. OVERR_SPEC_FOR_SIMP Override gilt nur für Superimposed-Bewegung. OVERR_SPEC_FOR_PHAS Override gilt nur für Phasing.
VAR_OUTPUT			
Enabled	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn der Override übernommen wurde.
Busy	BOOL	-	Funktion in Ausführung
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Initialisierungswerte

VelFactor	1.0
AccFactor	1.0
JerkFactor	1.0
OverrSpecification	OVERR_SPEC_FOR_ALL

Fehlerbeschreibung

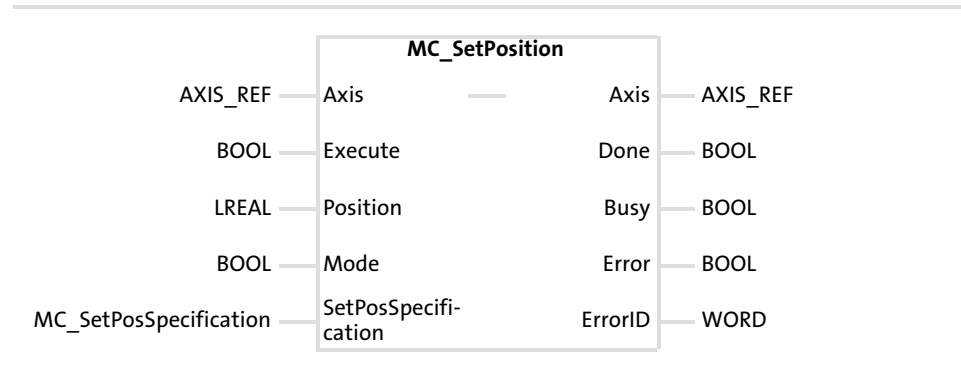
ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
27	Unerlaubter Wert für OverrSpecification .	> 3
28	Der Wert für VelFactor außerhalb des zulässigen Bereiches.	0 <= Zulässiger Bereich <= 1.2
29	Der Wert für AccFactor außerhalb des zulässigen Bereiches.	0 < Zulässiger Bereich <= 1.2
30	Der Wert für JerFactor außerhalb des zulässigen Bereiches.	0 < Zulässiger Bereich <= 1.2
100	Achsfehler ist während der Ausführung des Befehls aufgetreten.	Achsfehler ist mit MC_ReadAxisError einzulesen.
101	Systemfehler ist während der Ausführung des Befehls aufgetreten.	Systemfehler ist mit MC_ReadSysError einzulesen.

4.3.10 Funktionsbaustein MC_SetPosition

Aufgabe

Mit diesem Funktionsbaustein kann die aktuelle Position einer Achse auf neue Position gesetzt werden, was effektiv zu einer Koordinatensystemverschiebung führt.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Diese Funktion verschiebt das Koordinatensystem einer Achse so, dass die aktuelle Position der Achse der vorgegebenen Position entspricht. Die Funktion kann auch während der Bewegung einer Achse ausgeführt werden, wobei die aktuelle Zielkoordinate unverändert bleibt und in neuem Koordinatensystem angefahren wird. Die laufende Bewegung wird von den neuen Startbedingungen aus erneut gestartet.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Axis	AXIS_REF	P	Index einer Achse. 0 ... MaxAxis
VAR_INPUT			
Execute	BOOL	P	Mit steigender Flanke wird die Aufgabe ausgeführt.
Position	LREAL	O	Position, auf welche die aktuelle Position gesetzt werden soll.
Mode	BOOL	O	= FALSE - die aktuelle Position soll auf den über den Eingangsparameter Position vorgegebenen Wert gesetzt werden. = TRUE – die aktuelle Position soll um den über den Eingangsparameter Position vorgegebenen Wert verschoben werden.
SetPosSpecification	MC_SetPosSpecification	O	Enum-Typ MC_SetPosSpecification: TYPE MC_SetPosSpecification : (SET_POS_ALL_AXIS_MOTION_E:=0 SET_POS_E :=1, SET_POS_SUPERIMPOSED_E :=2); END_TYPE Bei SET_POS_ALL_AXIS_MOTION_E wird die aktuelle Position der Achse auf vorgegebene Position gesetzt, wobei die Superimposedposition ²⁾ genullt wird. Bei SET_POS_E wird die aktuelle Position der Achse auf vorgegebene Position gesetzt, wobei die Superimposedposition ²⁾ erhalten bleibt. Bei SET_POS_SUPERIMPOSED_E wird die aktuelle Superimposedposition ²⁾ der Achse auf die vorgegebene Position gesetzt.
VAR_OUTPUT			
Done	BOOL	-	TRUE zeigt, dass die Aufgabe ausgeführt wurde.
Busy	BOOL	-	Funktion in Ausführung
Error	BOOL	-	TRUE, wenn ein Fehler bei der Ausführung dieses Befehls aufgetreten ist.
ErrorID	WORD	-	Fehlercode

1) P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

2) Superimposedposition – Position, die mit MC_MoveSuperImposed erzeugt wird

Initialisierungswerte

Position	0
SetPosSpecification	SET_POS_ALL_AXIS_MOTION

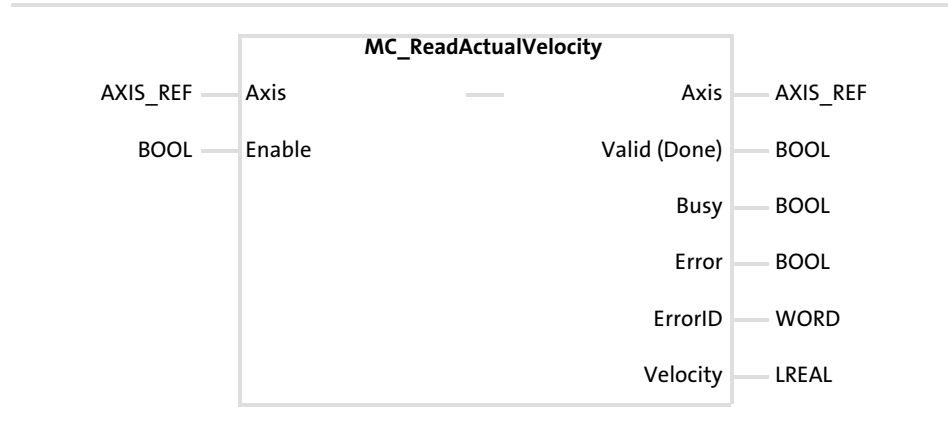
Fehlerbeschreibung

ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
1	Unzulässiger Zustand der Achse	Achse noch nicht initialisiert oder im Fehlerzustand.
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
19	Diese Instanz ist für eine andere Achse aktiv	
100	Achsfehler ist während der Ausführung des Befehls aufgetreten.	Achsfehler ist mit MC_ReadAxisError einzulesen.
101	Systemfehler ist während der Ausführung des Befehls aufgetreten.	Systemfehler ist mit MC_ReadSysError einzulesen.
1080	Unzulässiger Wert für Mode	> 1
1081	Unzulässiger Wert für SetPosSpecification	> 2

4.3.11 Funktionsbaustein MC_ReadActualVelocity

Aufgabe

Mit diesem Funktionsbaustein kann die aktuelle Geschwindigkeit der Achse gelesen werden.

Schematische
KOP/FUP-Darstellung

Funktionsbeschreibung

Solange der Eingangsparameter **Enable** = TRUE ist, wird der Ausgangsparameter **Velocity** auf die aktuelle Istgeschwindigkeit der programmierten Achse gesetzt.

Bei **Enable** = FALSE wird der Ausgangsparameter **Velocity** auf Null gesetzt. Bei Antriebsachsen wird die Istgeschwindigkeit aus den letzten zwei durch den Antrieb gemeldeten Istpositionen berechnet. Bei virtuellen Achsen wird die Istgeschwindigkeit auf die vom Interpolator erzeugte Sollgeschwindigkeit gesetzt.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Axis	AXIS_REF	P	Index einer Achse. 0 ... MaxAxis
VAR_INPUT			
Enable	BOOL	P	TRUE = aktuelle Istgeschwindigkeit wird kontinuierlich ausgegeben.
VAR_OUTPUT			
Valid (Done)	BOOL	-	TRUE = Istgeschwindigkeit wurde über den Ausgang Velocity ausgegeben.
Busy	BOOL	-	Funktion in Ausführung
Error	BOOL	-	TRUE, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode
Velocity	LREAL	-	Aktuelle Sollgeschwindigkeit der Achse

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Fehlerbeschreibung

ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.

4.3.12 Funktionsbaustein MC_ReadActualTorque



Stop!

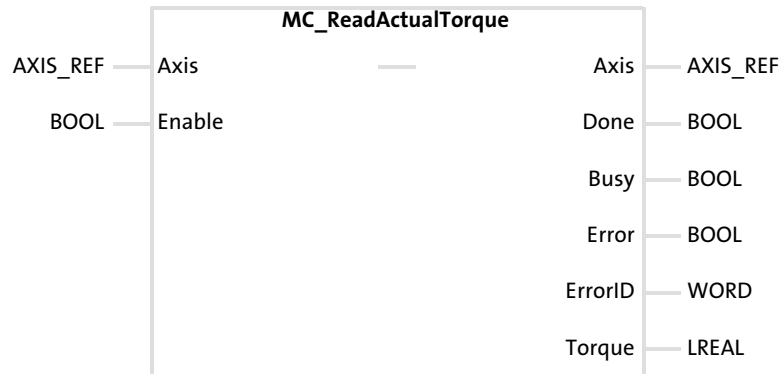
Dieser Funktionsbaustein setzt voraus, dass die CANopen-Antriebe die Regelung des Drehmoments über CANopen unterstützen.

Bei Achsen mit analoger Schnittstelle ist der FB nicht verwendbar.

Aufgabe

Mit diesem Funktionsbaustein kann das aktuelle vom Antrieb gemeldete Drehmoment (die aktuelle Kraft im Fall eines linearen Motors) der Achse gelesen werden.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Solange der Eingangsparameter **Enable** = TRUE ist, wird der Ausgangsparameter **Torque** auf den aktuellen vom Antrieb gemeldeten Drehmoment in [Nm] der programmierten Achse gesetzt. Im Fall eines linearen Motors wird die aktuelle vom Antrieb gemeldete Kraft in [N] geliefert.

Bei **Enable** = FALSE wird der Ausgangsparameter **Torque** auf Null gesetzt.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Axis	AXIS_REF	P	Index einer Achse. Wert 0 ... MaxAxis
VAR_INPUT			
Enable	BOOL	P	TRUE = aktueller Ist-Drehmoment wird kontinuierlich ausgegeben.
VAR_OUTPUT			
Done	BOOL	-	TRUE = Ist-Drehmoment wurde über den Ausgangsparameter Torque ausgegeben. Einheit [Nm].
Busy	BOOL	-	Funktion in Ausführung
Error	BOOL	-	TRUE, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode
Torque	LREAL	-	Aktuelles Drehmoment der Achse

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Fehlerbeschreibung

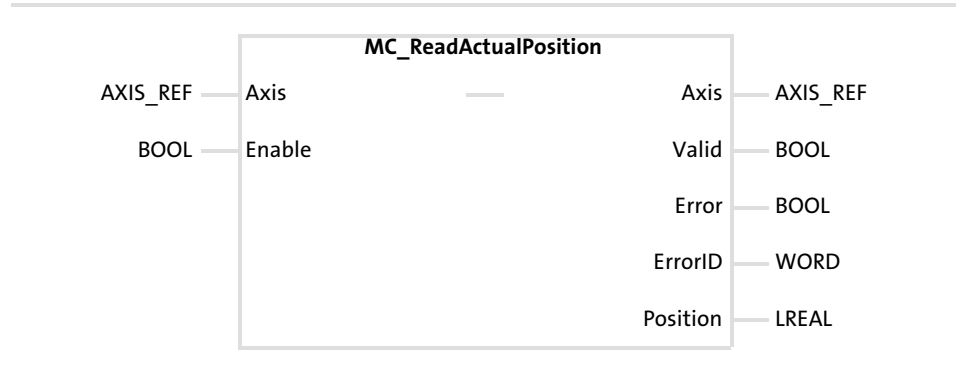
ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.

4.3.13 Funktionsbaustein MC_ReadActualPosition

Aufgabe

Mit diesem Funktionsbaustein kann die aktuelle Istposition der Achse gelesen werden.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Solange der Eingangsparameter **Enable** = TRUE ist, wird der Ausgangsparameter **Position** auf die aktuelle Ist-Position der programmierten Achse gesetzt.

Bei **Enable** = FALSE wird der Ausgang **Position** auf Null gesetzt.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Axis	AXIS_REF	P	Index einer Achse. Wert 0 ... MaxAxis
VAR_INPUT			
Enable	BOOL	P	TRUE = aktuelle Istposition wird kontinuierlich ausgegeben.
VAR_OUTPUT			
Valid	BOOL	-	TRUE = Istposition wurde über den Ausgang Valid ausgegeben.
Error	BOOL	-	TRUE, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode
Position	LREAL	-	Aktuelle Istposition der Achse

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Fehlerbeschreibung

ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.

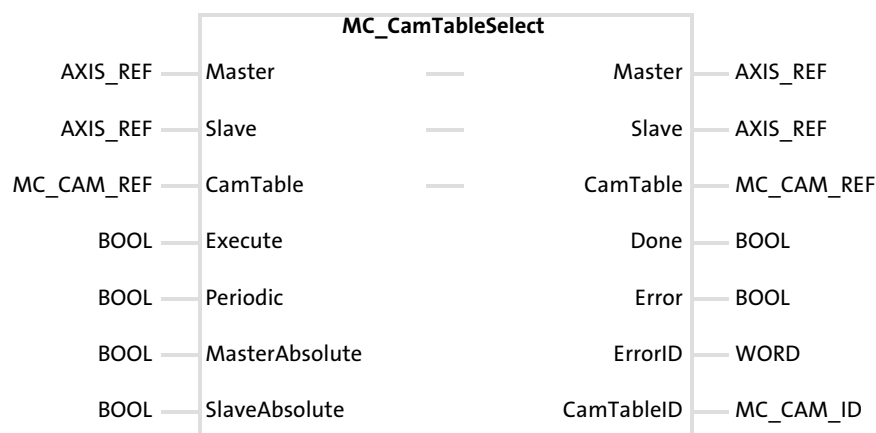
4.4 Achskopplung Funktionsbausteine nach PLCopen-Standard

4.4.1 Funktionsbaustein MC_CamTableSelect

Aufgabe

Mit diesem Funktionsbaustein wird eine Kurvenscheibentabelle (Kopplungstabelle) für die Kopplung zwischen einer Master- und einer Slave-Achse selektiert (beim Interpolator angemeldet).

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Vordefinierte Datentypen in mc.lib

```

TYPE MC_CAM_REF :
STRUCT
  NoOfRows:   DWORD;
  CamPosPtr:  DWORD;
END_STRUCT
END_TYPE
TYPE MC_POS_TR :
STRUCT
  master:     LREAL;
  slave:      LREAL;
  interpol:   MC_CAM_INT_ART;
END_STRUCT
END_TYPE
  
```

NoOfRows: Anzahl von Stützstellen

CamPosPtr: Adresse der Tabelle mit den Stützstellen. Die Tabelle muss in der Applikation als **ARRAY[0...NoOfRows-1] OF MC_POS_TR** deklariert werden. In dieser Array ist die erste Spalte eine Master Position, die zweite Spalte eine Slave-Position und die dritte Spalte die Interpolationsart. Die Tabelle enthält nur Binärdaten.

Die Positionen in der Tabelle sind einheitenneutral. Erst nach der Zuordnung der Kurvenscheibentabelle zu einer Master- und Slaveachse werden die jeweiligen Einheiten der Achse übernommen. Die Einheit einer Achse wird indirekt mit den Konfigurationsdaten **MK_DISTANCE** und **MK_INCREMENTS**

oder mit der Struktur **AXIS_CONFIG_CONST_TR** (**lrRoad** und **lrImpulse**) vorgegeben.

Die CAM-Tabelle kann einmalig oder zyklisch ausgeführt werden.

Bei aktiver Bearbeitung können die Tabellenwerte geändert werden. Man kann auch zwischen verschiedenen Kopplungstabellen umschalten.

Zwischen den Stützpunkten der Kopplungstabelle werden Segmente nach der gewählten Interpolationsart eingefügt:

0 - linear

1 - kubische Spline (Polynome 3. Grades)

2 - quintische Spline (Polynome 5. Grades)

Der Tabelleninhalt kann aus einer Datei eingelesen oder in der SPS berechnet und eingetragen werden.

Als Master kann eine virtuelle Achse benutzt werden.



Stop!

Die Selektierung (die Anmeldung beim Interpolator) der nächsten Tabelle für die gleiche Slave-Achse sollte erst erfolgen, wenn die vorherige Selektierung mit dem Ausgang **Done=TRUE** angezeigt wurde, sonst wird die Selektierung vorheriger Tabellen abgebrochen.

Die Eingangsparameter **Master** und **Slave** sind nur aus Kompatibilitätsgründen zu PLCopen notwendig. Die beim Interpolator angemeldeten Tabellen können bei **MC_CamIn** für beliebige zwei Achsen eingesetzt werden.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Master	AXIS_REF	P	Index einer Master-Achse. Wert: 0 ... MaxAxis
Slave	AXIS_REF	P	Index einer Slave-Achse. Wert: 0 ... MaxAxis
CamTable	MC_CAM_REF	P	Referenz auf die Kopplungstabelle
VAR_INPUT			
Execute	BOOL	P	Selektierung mit steigender Flanke
Periodic	BOOL	O	TRUE = periodische FALSE = nicht periodische Tabelle
MasterAbsolute	BOOL	O	TRUE oder FALSE hat die gleiche Bedeutung. Tabellendaten für den Master werden immer als relative Positionen in Bezug auf die erste Master-Position in der Tabelle gewertet, da die Tabelle immer mit Master-Position 0.0 beginnen muss.
SlaveAbsolute	BOOL	O	TRUE = Tabellendaten für den Slave als absolute Positionen werten. FALSE = Tabellendaten für den Slave als relative Positionen in Bezug auf die erste Slaveposition in der Tabelle werten.
VAR_OUTPUT			
Done	BOOL	-	Selektierung ausgeführt
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode
CamTableID	WORD	-	Identifikationsnr. der Tabelle über die gekoppelt wird; für MC_CamIn

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Initialisierungswerte

Periodic	FALSE
MasterAbsolute	TRUE
SlaveAbsolute	TRUE

Fehlerbeschreibung

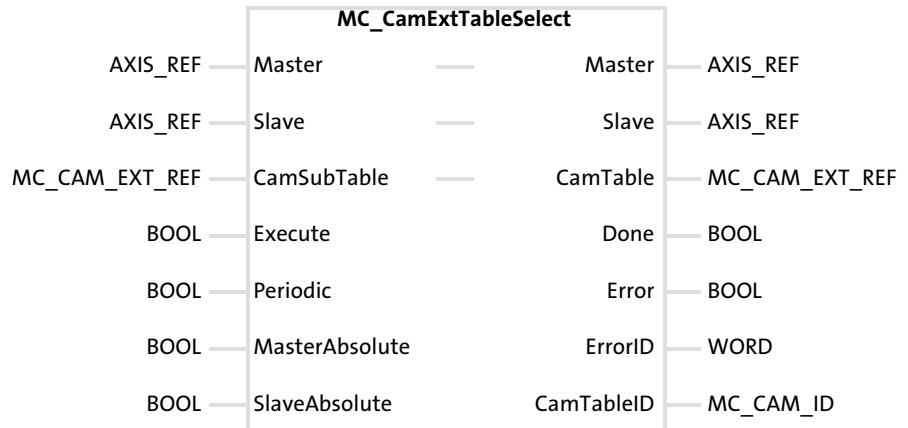
ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
1	Zustand nicht erlaubt	Der Funktionsbaustein kann nur im Interpolator-Zustand Init_Drive_Ok ausgeführt werden.
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
13	Zeiger CamPosPtr gleich Null	
14	Speicher für CAM-Tabellen voll	Zu viele Tabellen. Maximale Anzahl von CAM-Tabellen ist 16.
32	NoOfRows in CamTable -Tabelle < 2	Die Tabelle muss mindestens zwei Stützstellen beinhalten.
100	Achsfehler ist während der Ausführung des Befehls aufgetreten.	Achsfehler ist mit MC_ReadAxisError einzulesen.
101	Systemfehler ist während der Ausführung des Befehls aufgetreten.	Systemfehler ist mit MC_ReadSysError einzulesen.

4.4.2 Funktionsbaustein MC_CamExtTableSelect

Aufgabe

Mit diesem Funktionsbaustein wird eine erweiterte Kurvenscheibentabelle (Kopplungstabelle) für die Kopplung zwischen einer Master- und einer Slave-Achse selektiert.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Vordefinierte Datentypen in mc.lib

```

TYPE MC_CAM_ID : WORD; END_TYPE
TYPE MC_CAM_EXT_REF :
STRUCT
  NoOfRows:      DWORD;
  CamPosPtr:     DWORD;
END_STRUCT
END_TYPE
TYPE MC_POS_EXT_TR :
STRUCT
  master:        LREAL; //Master-Position (M)
  slave:         LREAL; //Slave-Position (S)
  firstDerivation: LREAL; //1.te Ableitung (dS/dM)
  secondDerivation: LREAL; //1.te Ableitung (dS²/dM²)
  interpol:      UINT
END_STRUCT
END_TYPE
    
```

NoOfRows: Anzahl von Stützstellen

CamPosPTR: Adresse der Tabelle mit den Stützstellen. Die Tabelle muss in der Applikation als **ARRAY[0...NoOfRows-1] OF MC_POS_EXT_TR** deklariert werden.

Die erste Spalte ist in der MC_POS_EXT_TR Struktur eine Master-Position, die zweite Spalte ist eine Slave-Position. In der dritten und vierten Spalte können die Übergangsbedingungen zum nächsten Segment vorgegeben werden.

Ob die Übergangsbedingungen, d. h. die erste und/oder die zweite Ableitung relevant sind wird in der fünften Spalte zusammen mit der Interpolationsart bestimmt.

Die Variable interpol in der Struktur wird wie folgt aufgebaut:

$\text{interpol} = n \cdot 256 + m$; wobei:

$m=0$ - lineare Interpolation

$m=1$ - Spline 3.ter Ordnung

$m=2$ - Spline 5.ter Ordnung

$n=0$ - nur die Positionen (master, slave) sind relevant

$n=1$ - die Positionen (master,slave) und die 1.te Ableitung sind relevant

$n=2$ - die Positionen (master,slave) und die 2.te Ableitung sind relevant

$n=3$ - alle Vorgaben sind relevant

Die Positionen der Master-Achse (**master**) müssen monoton wachsend sein. Abhängig von der angewählten Interpolationsart kann ggf. die 1. Ableitung (**firstDerivation**) und/oder die 2. Ableitung (**secondDerivation**) an der jeweiligen Position eingetragen werden. Die nicht vorgegebenen Ableitungen, die für die eingegebene Interpolationsart notwendig sind, werden durch den Funktionsbaustein berechnet. Die 1. Ableitung wird aus den eingegebenen Master- und Slave-Positionen berechnet. Die 2. Ableitung wird aus den eingegebenen oder der den berechneten 1. Ableitungen berechnet. Die eingegebenen Ableitungen, die für die eingegebene Interpolationsart nicht notwendig sind, werden ignoriert.

Die Position der Slave-Achse muss immer eingegeben werden, mit einer Ausnahme: wenn die 1. Ableitung (Steigung) bei linearer Interpolation eingegeben wird, wird die Slave-Position aus der vorheriger Slave-Position und der eingegebenen 1. Ableitung berechnet. Die eingegebene Slave-Position wird ignoriert. Die Spline-Interpolation 3. Ordnung erzeugt Kurven mit der stetigen 1. Ableitung. Die Spline-Interpolation 5. Ordnung erzeugt Kurven mit der stetigen 1. und 2. Ableitung. Die gewünschten Unstetigkeiten in der 1. und/oder in der 2. Ableitung können durch 2 Stützpunkte mit identischer Master-Position beschrieben werden. In dem Fall muss jeweils die 1. und/oder die 2. Ableitung eingegeben werden.

Die Tabelle enthält nur Binärdaten. Die Positionen in der Tabelle sind einheitsneutral. Erst nach Zuordnung der Kurvenscheibentabelle zu einer Master- und Slaveachse werden die jeweiligen Einheiten der Achse übernommen. Die Einheit einer Achse wird indirekt mit den Konfigurationsdaten **MK_DISTANCE** und **MK_INCREMENTS** oder mit der Struktur **AXIS_CONFIG_CONST_TR** (**lrRoad** und **lrImpulse**) vorgegeben.

Die CAM-Tabelle kann einmalig oder zyklisch ausgeführt werden.

Bei aktiver Bearbeitung können die Tabellenwerte geändert werden. Auch kann zwischen verschiedenen Kopplungstabellen umgeschaltet werden. Zwischen den Stützpunkten der Kopplungstabelle werden Segmente nach der gewählten Interpolationsart eingefügt:

0 - linear

1 - kubischer Spline (Polynome 3. Grades)

2 - quintischer Spline (Polynome 5. Grades)

Der Tabelleninhalt kann aus einer Datei eingelesen oder in der SPS berechnet und eingetragen werden. Als Master kann eine virtuelle Achse benutzt werden.

**Hinweis!**

Die Selektierung (die Anmeldung beim Interpolator) der nächsten Tabelle für die gleiche Slave-Achse sollte erst erfolgen, wenn die vorherige Selektierung mit dem Ausgang Do-ne=TRUE angezeigt wurde. Sonst wird die Selektierung der vorherigen Tabelle abgebrochen.

Die Eingangsparameter **Master** und **Slave** sind nur aus Kompatibilitätsgründen zu PLCopen notwendig. Die beim Interpolator angemeldeten Tabellen können bei **MC_CamIn** für beliebige zwei Achsen eingesetzt werden.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Master	AXIS_REF	P	Index einer Master-Achse. Wert: 0 ... MaxAxis
Slave	AXIS_REF	P	Index einer Slave-Achse. Wert: 0 ... MaxAxis
CamSubTable	MC_CAM_EXT_REF	P	Referenz auf die Kopplungssubtabelle
VAR_INPUT			
Execute	BOOL	P	Funktionsausführung mit steigender Flanke
Periodic	BOOL	O	TRUE = periodische FALSE = nicht periodische Tabelle
MasterAbsolute	BOOL	O	TRUE oder FALSE hat die gleiche Bedeutung. Tabellendaten für den Master werden immer als relative Positionen in Bezug auf die erste Master-Position in der Tabelle gewertet, da die Tabelle immer mit Master-Position 0.0 beginnen muss.
SlaveAbsolute	BOOL	O	TRUE = Tabellendaten für den Slave als absolute Positionen werten. FALSE = Tabellendaten für den Slave als relative Positionen in Bezug auf die erste Slave-Position in der Tabelle werten. Nur bei SubTableType = MC_SUBTABLE_FIRST_E von Bedeutung.
VAR_OUTPUT			
Done	BOOL	-	Selektierung ausgeführt
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode
CamTableID	MC_CAM_ID	-	Identifikationsnummer der Tabelle, die der Baustein MC_CamIn zum Koppeln der Achsen benutzt.

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Initialisierungswerte

Periodic	FALSE
MasterAbsolute	TRUE
SlaveAbsolute	TRUE

Fehlerbeschreibung

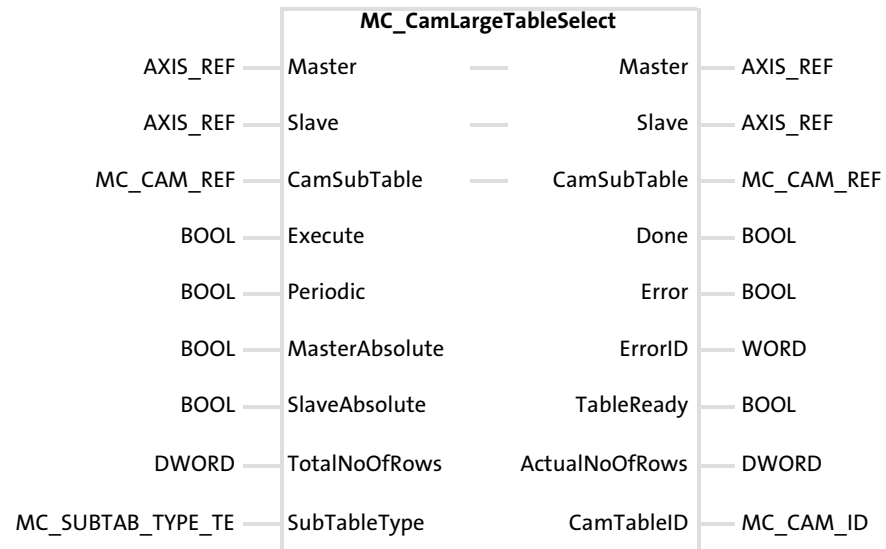
ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
1	Zustand nicht erlaubt	Der Funktionsbaustein kann nur im Interpolator-Zustand Init_Drive_Ok ausgeführt werden.
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
13	Zeiger CamPosPtr gleich Null	
14	Speicher für CAM-Tabellen voll	Zu viele Tabellen. Max. Anzahl Generation 1: 16 Max. Anzahl Generation 2: 60
32	NoOfRows in CamTable -Tabelle < 2	Die Tabelle muss mindestens zwei Stützstellen beinhalten.
100	Achsfehler ist während der Ausführung des Befehls aufgetreten.	Achsfehler ist mit MC_ReadAxisError einzulesen.
101	Systemfehler ist während der Ausführung des Befehls aufgetreten.	Systemfehler ist mit MC_ReadSysError einzulesen.
1200	Master-Periode = 0	Die erste und die letzte Masterposition in der Tabelle sind gleich
1201	Unzulässige Interpolationsart in Interpol	> 2
1202	Masterpositionen nicht aufsteigend	
1204	Vorgabe der 1.ten Ableitung bei den gleichen Masterpositionen fehlt	
1205	Vorgabe der 2. Ableitung bei den gleichen Masterpositionen fehlt	Nur bei m=2, d. h. Spline 5. Ordnung
1207	Vorgabe der 1. und der 2. Ableitung fehlt	Bei m=2, d. h. Spline 5. Ordnung muss bei der 1. Stützstelle die 1. und die 2. Ableitung vorgegeben werden
1208	Ungleiche Slavepositionen bei den zwei gleichen Masterpositionen	
1209	Zwei gleiche Masterpositionen bei Linearinterpolation	

4.4.3 Funktionsbaustein MC_CamLargeTableSelect

Aufgabe

Mit diesem Funktionsbaustein wird eine Kurvenscheibentabelle (Kopplungstabelle) für die Kopplung zwischen einer Master- und einer Slave-Achse selektiert. Die Tabelle wird in mehreren Schritten aus Subtabellen gebildet.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Vordefinierte Datentypen in mc.lib

```

TYPE MC_CAM_ID : WORD; END_TYPE
TYPE MC_CAM_REF :
STRUCT
  NoOfRows:      DWORD;
  CamPosPtr:     DWORD;
END_STRUCT
END_TYPE
TYPE MC_POS_TR :
STRUCT
  master:        LREAL;
  slave:         LREAL;
  interpol:      MC_CAM_INT_ART;
END_STRUCT
END_TYPE

```

NoOfRows: Anzahl von Stützstellen

CamPosPTR: Adresse der Tabelle mit den Stützstellen. Die Tabelle muss in der Applikation als **ARRAY[0...NoOfRows-1] OF MC_POS_TR** deklariert werden. In dieser Array ist die erste Spalte eine Master-Position, die zweite Spalte eine Slave-Position und die dritte Spalte die Interpolationsart. Die Tabelle enthält nur Binärdaten. Die Positionen in der Tabelle sind einheitenneutral. Erst nach der Zuordnung der Kurvenscheibentabelle zu einer Master- und Slave-Achse werden die jeweiligen Einheiten der Achse übernommen. Die Einheit einer Achse wird indirekt mit den Konfigurationsdaten **MK_DISTANCE** und **MK_INCREMENTS** oder mit der Struktur

AXIS_CONFIG_CONST_TR (IrRoad und IrImpulse) vorgegeben.

Die CAM-Tabelle kann einmalig oder zyklisch ausgeführt werden.

Bei aktiver Bearbeitung können die Tabellenwerte geändert werden, sowie man auch zwischen verschiedenen Kopplungstabellen umschalten kann.

Zwischen den Stützpunkten der Kopplungstabelle werden Segmente nach der gewählten Interpolationsart eingefügt:

0 - linear

1 - kubischer Spline (Polynome 3. Grades)

2 - quintischer Spline (Polynome 5. Grades)

Der Tabelleninhalt kann aus einer Datei eingelesen oder in der SPS berechnet und eingetragen werden.

Als Master kann eine virtuelle Achse benutzt werden.



Hinweis!

Die Eingangsparameter **Master** und **Slave** sind nur aus Kompatibilitätsgründen zu PLCopen notwendig. Die beim Interpolator angemeldeten Tabellen können bei **MC_CamIn** für beliebige zwei Achsen eingesetzt werden.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Master	AXIS_REF	P	Index einer Master-Achse. Wert: 0 ... MaxAxis
Slave	AXIS_REF	P	Index einer Slave-Achse. Wert: 0 ... MaxAxis
CamSubTable	MC_CAM_REF	P	Referenz auf die Kopplungssubtabelle
VAR_INPUT			
Execute	BOOL	P	Funktionsausführung mit steigender Flanke
Periodic	BOOL	O	TRUE = periodische FALSE = nicht periodische Tabelle Nur bei SubTableType = MC_SUBTABLE_FIRST_E von Bedeutung.
MasterAbsolute	BOOL	O	TRUE oder FALSE hat die gleiche Bedeutung. Tabellendaten für den Master werden immer als relative Positionen in Bezug auf die erste Master-Position in der Tabelle gewertet, da die Tabelle immer mit Master-Position 0.0 beginnen muss. Nur bei SubTableType = MC_SUBTABLE_FIRST_E von Bedeutung.
SlaveAbsolute	BOOL	O	TRUE = Tabellendaten für den Slave als absolute Positionen werten. FALSE = Tabellendaten für den Slave als relative Positionen in Bezug auf die erste Slave-Position in der Tabelle werten. Nur bei SubTableType = MC_SUBTABLE_FIRST_E von Bedeutung.
TotalNoOfRows	DWORD	O	Summe von Stützstellenanzahl aus allen Subtabellen. Nur bei SubTableType = MC_SUBTABLE_FIRST_E von Bedeutung.

Name	Datentyp	P/O ¹⁾	Bedeutung
SubTableType	MC_SUB-TAB_TYPE_TE		Enum-Typ. Da die resultierende Kopplungstabelle aus mehreren Subtabellen aufgebaut wird, wird es zwischen den einzelnen Subtabellen über diesen Eingangsparameter unterschieden. TYPE MC_SUBTAB_TYPE_TE : (MC_SUBTABLE_FIRST_E := 0, MC_SUBTABLE_NEXT_E := 1, MC_SUBTABLE_LAST_E := 2); END_TYPE
VAR_OUTPUT			
Done	BOOL	-	Ausführung der Funktion fertig
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode
TableReady	BOOL	-	Die gesamte Tabelle steht nach dem letzten Block (SubTableType = SUBTABLE_LAST_E) zur Verfügung.
ActualNoOfRows	DWORD	-	Zeigt nach dem Übertragen einer Subtabelle die aktuelle Anzahl der übertragenen Stützstellen (der gesamten Tabelle) an.
CamTableID	MC_CAM_ID	-	Identifikationsnummer der Tabelle, die der Baustein MC_CamIn zum Koppeln der Achsen benutzt. Die richtige Nummer steht nur an, wenn der Eingangsparameter SubTableType = SUBTABLE_LAST_E .

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Initialisierungswerte

Periodic	FALSE
MasterAbsolute	TRUE
SlaveAbsolute	TRUE

Fehlerbeschreibung

ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
1	Zustand nicht erlaubt	Der Funktionsbaustein kann nur im Interpolator-Zustand Init_Drive_Ok ausgeführt werden.
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
13	Zeiger CamPosPtr gleich Null	
14	Speicher für CAM-Tabellen voll	Zu viele Tabellen. Maximale Anzahl von CAM-Tabellen ist 16.
40	Unzulässiger Wert von SubTableType	
41	Speicher voll	Zu wenig Speicher für Anlegen einer neuen Tabelle.
100	Achsfehler ist während der Ausführung des Befehls aufgetreten.	Achsfehler ist mit MC_ReadAxisError einzulesen.
101	Systemfehler ist während der Ausführung des Befehls aufgetreten.	Systemfehler ist mit MC_ReadSysError einzulesen.

4.4.4 Funktionsbaustein MC_CamTableDelete

Aufgabe

Mit diesem Funktionsbaustein wird eine Kopplungstabelle für die Kopplung zwischen einer Master- und einer Slave-Achse gelöscht. Der mit MC_CamLargeTableSelect reservierte Speicher für die gelöschte Kopplungstabelle wird freigegeben.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Mit dem Funktionsbaustein können sowohl mit dem Funktionsbaustein MC_CamTableSelect und MA_CamExtTableSelect als auch mit dem Funktionsbaustein MC_CamLargeTableSelect beim Interpolator angemeldeten Kopplungstabellen gelöscht werden, wobei der mit MC_CamLargeTableSelect reservierten Speicher freigegeben wird.

Vordefinierte Datentypen in mc.lib

TYPE MC_CAM_ID : WORD; END_TYPE

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
CamTableID	MC_CAM_ID	O	Identifikationsnummer der Kopplungstabelle
Execute	BOOL	P	Löschen der Tabelle mit steigender Flanke.
VAR_OUTPUT			
Done	BOOL	-	Selektierung ausgeführt
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Fehlerbeschreibung

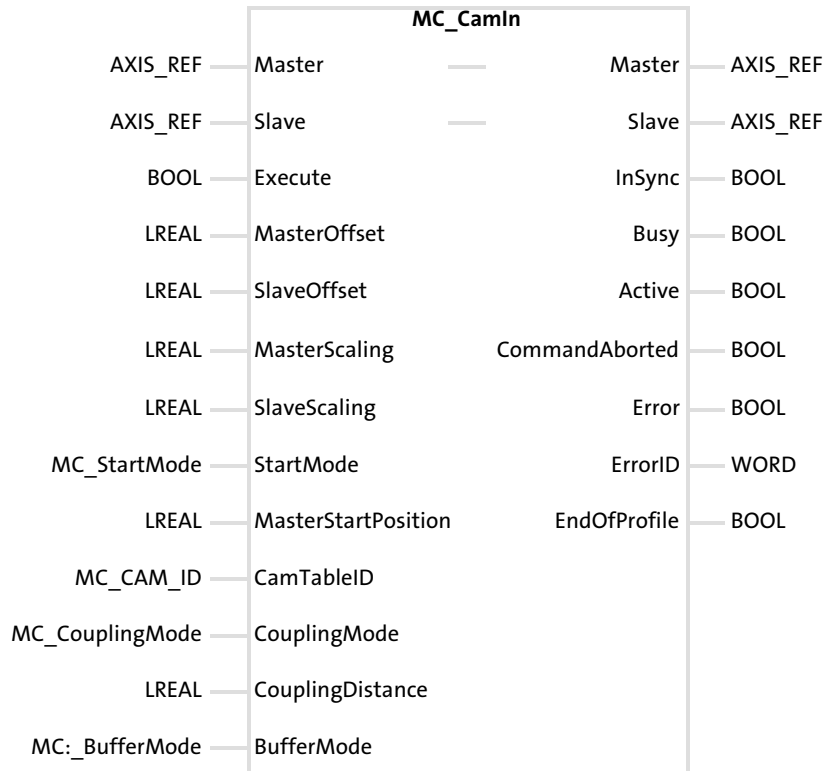
ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
1	Steuerung befindet sich in einem für diese Funktion unzulässigen Zustand.	
15	Die zu löschende Tabelle existiert nicht.	
42	Tabelle kann zurzeit nicht gelöscht werden.	Eine nach dieser Tabelle gekoppelte Bewegung ist gerade aktiv.

4.4.5 Funktionsbaustein MC_CamIn

Aufgabe

Mit diesem Funktionsbaustein wird die Kopplung über eine Kopplungstabelle zwischen einer Master- und einer Slave-Achse aktiviert.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Bei **CouplingMode = COUPLING_MODE_ERROR_E** wird über den Ausgang ErrorID ein Fehler gemeldet, wenn die Differenz zwischen der aktuellen Sollposition der Slave-Achse und der Position aus der Kopplungstabelle den Weg, der dem maximalen Ruck der Slave-Achse pro Interpolationstakt entspricht, überschreitet. Die Kopplung wird nicht aktiviert.

Bei **CouplingMode = COUPLING_MODE_SUPERIMPOSED_E** wird die Differenz zwischen der aktuellen Sollposition der Slave-Achse und der Position aus der Kopplungstabelle auf die Sollposition der überlagerten Bewegung aufaddiert und die Kopplung wird aktiviert. Die Slave-Achse hat damit einen zusätzlichen Offset. Der Offset kann mit MC_MoveSuperimposed auf 0 oder auf eine andere Distanz gefahren werden.

Bei **StartMode=STARTMODE_RAMP_IN_E** wird die Slave-Achse zunächst über mit dem Eingangsparameter **CouplingDistance** vorgegebene Master-Strecke auf die entsprechende Position aus der Tabelle gefahren. Erst danach wird die Kopplung aktiviert und beide Achsen fahren (lt. der Tabelle) synchron.

Vordefinierte Datentypen in mc.lib

```
TYPE MC_CAM_ID : WORD; END_TYPE
```

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Master	AXIS_REF	P	Index einer Master-Achse. Wert: 0 ... MaxAxis
Slave	AXIS_REF	P	Index einer Slave-Achse. Wert. Wert: 0 ... MaxAxis
VAR_INPUT			
Execute	BOOL	P	Aktivierung der Kopplung mit positiver Flanke
MasterOffset	LREAL	O	Master-Positionsoffset in [u]
SlaveOffset	LREAL	O	Slave-Positionsoffset in [u]
MasterScaling	LREAL	O	Master-Skalierungsfaktor
SlaveScaling	LREAL	O	Slave-Skalierungsfaktor
StartMode	MC_StartMode	O	Enum MC_StartMode: <pre> TYPE MC_StartMode : (STARTMODE_ABSOLUTE_E:=0, STARTMODE_RELATIVE_E:=1, STARTMODE_RAMP_IN_E:=2, STARTMODE_RAMP_IN_SHORTEST_WAY_E:=2, STARTMODE_RAMP_IN_POS_E:=3, STARTMODE_RAMP_IN_NEG_E:=4); END_TYPE </pre> Bei StartMode=STARTMODE_ABSOLUTE_E werden die Slave-Positionen in der Kopplungstabelle als die absoluten Positionen gewertet. Die Slavepositionen aus der Kopplungstabelle werden direkt als die Sollpositionen ausgegeben. Bei StartMode=STARTMODE_RELATIVE_E werden die Slave-Positionen als relative Positionen in der Kopplungstabelle gewertet. Die Slavepositionen aus der Kopplungstabelle werden auf die Koppelposition der Slave-Achse aufaddiert. Bei StartMode=STARTMODE_RAMP_IN_E wird die Slave-Achse über die mit dem Eingang CouplingDistance eingegebene Masterstrecke auf entsprechende Position aus der CAM-Tabelle gefahren. Erst danach werden die beiden Achsen synchron fahren. Bei StartMode=STARTMODE_RAMP_IN_SHORTEST_WAY_E wird eine Rotationschse als die Slave-Achse über die mit dem Eingang CouplingDistance eingegebene Masterstrecke auf entsprechende Position aus der CAM-Tabelle über den kürzesten Weg gefahren. Erst danach werden die beiden Achsen synchron fahren. Bei StartMode=STARTMODE_RAMP_IN_POS_E wird eine Rotationschse als die Slave-Achse über die mit dem Eingang Coupling-Distance eingegebene Masterstrecke auf entsprechende Position aus der CAM-Tabelle in die positive Richtung gefahren. Erst danach werden die beiden Achsen synchron fahren. Bei StartMode=STARTMODE_RAMP_IN_NEG_E wird eine Rotationschse als die Slave-Achse über die mit dem Eingang Coupling-Distance eingegebene Masterstrecke auf entsprechende Position aus der CAM-Tabelle in die negative Richtung gefahren. Erst danach werden die beiden Achsen synchron fahren.

Name	Datentyp	P/O ¹⁾	Bedeutung
MasterStartPos.	LREAL	O	Bei dieser Position der Master-Achse wird die Tabelle aufgesetzt, wenn BufferMode = BM_MASTER_POSITION_E programmiert wird.
CamTableD	MC_CAM_ID	O	Identifikationsnummer der Kopplungstabelle
CouplingMode	MC_Coupling-Mode	O	Enum MC_CouplingMode Typ: TYPE MC_CouplingMode : (COUPLING_MODE_ERROR_E :=0, COUPLING_MODE_SUPERIMPOSED_E :=1); END_TYPE Ankopplungsart der Slave-Achse an die Master-Achse. Bei CouplingMode=COUPLING_MODE_ERROR_E wird ein Fehler gemeldet, wenn die Differenz zwischen der aktuellen Slave-Position und der Slave-Position aus der CAM-Tabelle zum Zeitpunkt der Ankopplung größer als der Weg ist, der dem maximalen Ruck der Slave-Achse (MK_MAX_JERK) pro Interpolationstakt entspricht. Bei CouplingMode=COUPLING_MODE_SUPERIMPOSED wird die Differenz zwischen der aktuellen Slaveposition und der Slaveposition aus der CAM-Tabelle zum Zeitpunkt der Ankopplung auf die Superimposed-Position aufaddiert. Bei StartMode=STARTMODE_RAMP_IN_E hat CouplingMode keine Bedeutung, da keine Positionsdifferenz nach der Synchronisation vorhanden ist.
CouplingDistance	LREAL	P	Masterstrecke für die Ankopplung der Slave-Achse auf die entsprechende Position aus der CAM-Tabelle.
BufferMode	MC_BufferMode	O	Enum-Typ MC_BufferMode. TYPE MC_BufferMode : (BM_ABORTING_E :=0, BM_BUFFERED_E :=1, BM_BLENDING_LOW_E :=2, BM_BLENDING_PREVIOUS_E :=3, BM_BLENDING_NEXT_E :=4, BM_BLENDING_HIGH_E :=5, BM_MASTER_POSITION_E :=6); END_TYPE Bei BM_ABORTING_E wird die aktuelle Bewegung sofort abgelöst. Bei BM_BUFFERED_E wird die nächste Aufgabe erst gestartet, wenn die aktuelle beendet wird. Wenn die aktuelle Aufgabe MC_CamIn ist, dann wird die nächste gestartet, wenn der Zustand EndOfProfile erreicht wird. Bei BM_BLENDING_LOW_E, BM_BLENDING_PREVIOUS_E, BM_BLENDING_NEXT_E, BM_BLENDING_HIGH_E wird der Übergang wie bei BM_BUFFERED_E ausgeführt.
VAR_OUTPUT			
InSync	BOOL	-	Die Achsen sind gekoppelt
Busy	BOOL	-	Wird mit steigender Flanke von Execute gesetzt.
Active	BOOL	-	Bewegung in Ausführung
CommandAborted	BOOL	-	Die Kopplung wurde durch einen anderen Bewegungsbefehl abgebrochen.

Name	Datentyp	P/O ¹⁾	Bedeutung
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode
EndOfProfile	BOOL	-	Signalisiert das Ende der Kopplungstabelle (Bei zyklischen Tabellen bleibt mindestens einen Zyklus gesetzt).

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Initialisierungswerte

MasterOffset	0.0
SlaveOffset	0.0
MasterScaling	1.0
SlaveScaling	1.0
StartMode	STARTMODE_ABSOLUTE_E
CouplingMode	COUPLING_MODE_ERROR_E
CamTableID	0
BufferMode	BM_ABORTING_E

Fehlerbeschreibung

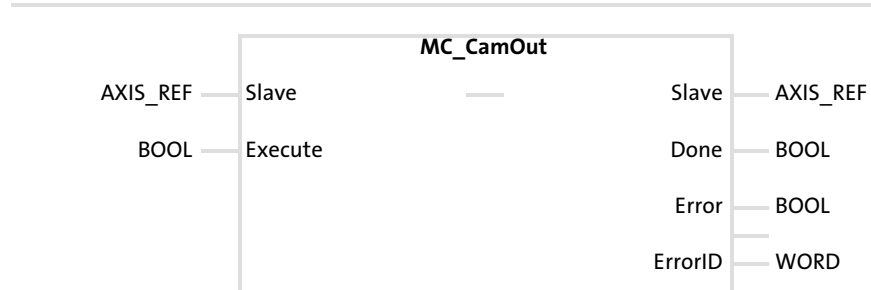
ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
1	Achse befindet sich in einem Zustand, aus dem diese Bewegung nicht gestartet werden kann.	Erlaubte Zustände: Standstill, Discrete Motion, Continuous Motion, Synchronized Motion
3	Slave- oder Master-Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
6	Reglerfreigabe der Slave-Achse fehlt	Mit MC_Power muss zunächst Reglerfreigabe gesetzt werden.
15	Unzulässiger Wert für CamTableID	
16	Unzulässiger Wert für StartMode	
17	Unzulässiger Wert für CouplingMode	
18	Differenz zwischen aktueller Position und Startposition aus der CAM-Tabelle der Slave-Achse zu groß.	Größer als maximaler Ruck der Slave-Achse.
19	Diese Instanz ist für eine andere Slave-Achse aktiv	
20	Unzulässiger Wert für BufferMode	
24	Für eine Achse von dieser Achsenart ist diese Bewegung nicht erlaubt.	
25	Negativer Softwareendschalter	
26	Positiver Softwareendschalter	
36	CouplingDistance = 0	
100	Achsfehler ist während der Ausführung des Befehls aufgetreten.	Achsfehler ist mit MC_ReadAxisError einzulesen.
101	Systemfehler ist während der Ausführung des Befehls aufgetreten.	Systemfehler ist mit MC_ReadSysError einzulesen.

4.4.6 Funktionsbaustein MC_CamOut

Aufgabe

Mit diesem Funktionsbaustein wird die Kopplung einer Slave-Achse an eine Master-Achse deaktiviert.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Nach Deaktivierung der Kopplung geht die Slave-Achse in den Zustand **Continuous Motion** und fährt mit der aktuellen Geschwindigkeit weiter.



Stop!

Nach MC_CamOut befindet sich die Achse in einer Bewegung ohne einen aktiven FB, der diese Bewegung angestoßen hat. Der aktuelle Zustand der Achse kann nur mit dem FB MC_ReadStatus erfasst werden.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Slave	AXIS_REF	P	Index der Slave-Achse. Wert: 0 ... MaxAxis
VAR_INPUT			
Execute	BOOL	P	Deaktivierung der Kopplung mit steigender Flanke
VAR_OUTPUT			
Done	BOOL	-	Kopplung deaktiviert
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Fehlerbeschreibung

ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
1	Zustand nicht zulässig	Nur im Zustand Synchronized Motion zulässig.
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
100	Achsfehler ist während der Ausführung des Befehls aufgetreten.	Achsfehler ist mit MC_ReadAxisError einzulesen.
101	Systemfehler ist während der Ausführung des Befehls aufgetreten.	Systemfehler ist mit MC_ReadSysError einzulesen.

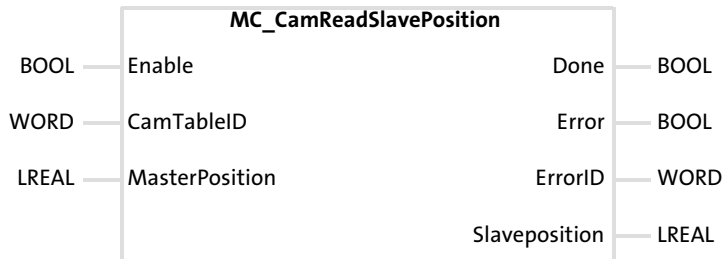
4.4.7 Funktionsbaustein MC_CamReadSlavePosition

Aufgabe

Der Funktionsbaustein liefert die Position der Slave-Achse aus der vorgegebenen CAM-Tabelle für die vorgegebene Position der Master-Achse.

Die vorgegebene Master-Position wird immer modulo Master-Periode (letzte Master-Position in der Tabelle) berechnet.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Vordefinierte Datentypen in mc.lib

```
TYPE MC_CAM_ID : WORD; END_TYPE
```

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Enable	BOOL	P	Bei True wird die Slaveposition geliefert.
CamTableID	WORD	O	Identifikationsnummer der Tabelle.
MasterPosition	LREAL	O	Master-Position für die Berechnung der Slaveposition aus der Tabelle.
VAR_OUTPUT			
Done	BOOL	-	TRUE=ein gültiger Wert steht am Ausgang Slaveposition an.
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode
SlavePosition	LREAL	-	Slaveposition aus der CAM-Tabelle

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Initialisierungswerte

MasterPosition	0.0
CamTableID	0

Fehlerbeschreibung

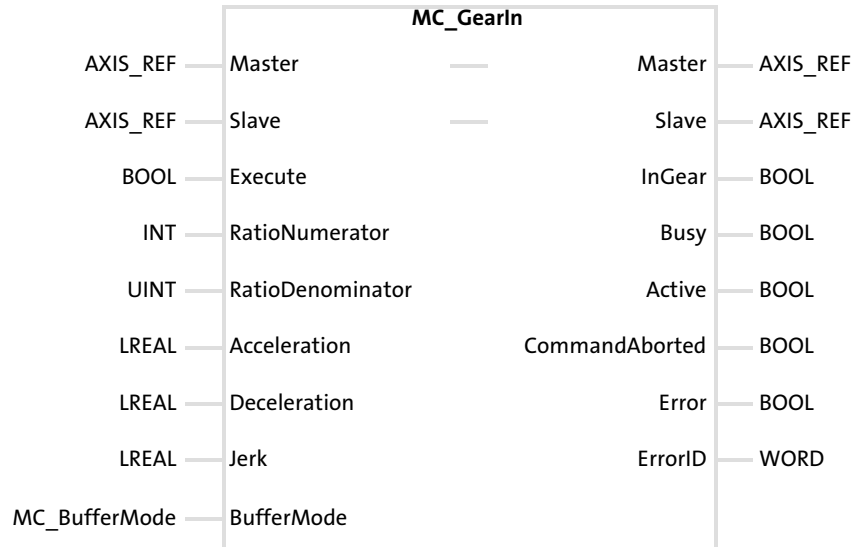
ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
15	Unzulässiger Wert für CamTableID	Die Tabelle muss vorher mit MC_CamTableSelect beim Interpolator angemeldet werden.

4.4.8 Funktionsbaustein MC_GearIn

Aufgabe

Mit diesem Funktionsbaustein wird ein elektronisches Getriebe zwischen einer Slave- und einer Master-Achse aktiviert.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Das Getriebeverhältnis kann auch bei eingeschaltetem Getriebe geändert werden, indem **MC_GearIn** mehrmals nacheinander aufgerufen wird; das heißt, die Kopplung muss vorher nicht mit **GearOut** ausgeschaltet werden.

Die Slave-Achse kann mit **MC_Stop** angehalten werden. Die bestehende Kopplung zwischen Slave- und Master-Achse kann mit **MC_GearOut** aufgehoben werden, wobei die Slave-Achse sich mit der aktuellen Geschwindigkeit weiter im Zustand **ContinuousMotion** bewegen wird.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Master	AXIS_REF	P	Index einer Master-Achse. Wert: 0 ... MaxAxis
Slave	AXIS_REF	P	Index einer Slave-Achse. Wert: 0 ... MaxAxis
VAR_INPUT			
Execute	BOOL	P	Mit steigender Flanke wird die Ausführung der Funktion gestartet.
RatioNumerator	INT	O	Zähler des Getriebeverhältnisses
RatioDenominator	INT	O	Nenner des Getriebeverhältnisses
Acceleration	LREAL	O	Sollbeschleunigung für die Ankopplungsphase in [u/s ²] (nur positiv)
Deceleration	LREAL	O	Sollverzögerung für die Ankopplungsphase in [u/s ²] (nur positiv)
Jerk	LREAL	O	Ruck für die Ankopplungsphase in [u/s ³] (nur positiv)
BufferMode	MC_BufferMode	O	Enum-Typ MC_BufferMode. TYPE MC_BufferMode : (BM_ABORTING_E := 0, BM_BUFFERED_E := 1, BM_BLENDING_LOW_E := 2, BM_BLENDING_PREVIOUS_E := 3, BM_BLENDING_NEXT_E := 4, BM_BLENDING_HIGH_E := 5, BM_MASTER_POSITION_E := 6); END_TYPE Bei BM_ABORTING_E wird die aktuelle Bewegung sofort abgelöst. Bei BM_BUFFERED_E wird die nächste Aufgabe erst gestartet, wenn die aktuelle beendet wird. Wenn die aktuelle Aufgabe MC_GearIn ist, dann wird die nächste erst gestartet, wenn der Zustand InGear erreicht wird. Bei BM_BLENDING_LOW_E , BM_BLENDING_PREVIOUS_E , BM_BLENDING_NEXT_E , BM_BLENDING_HIGH_E wird der Übergang wie bei BM_BUFFERED_E ausgeführt.
VAR_OUTPUT			
InGear	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn das Geschwindigkeitsverhältnis erreicht wurde.
Busy	BOOL	-	Wird mit steigender Flanke von Execute gesetzt.
Active	BOOL	-	Bewegung in Ausführung
CommandAborted	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn die Ausführung des Befehls durch eine andere Befehlsanforderung abgebrochen wurde.
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Initialisierungswerte

RatioNumerator	1
RatioDenominator	1
Acceleration	1000
Deceleration	1000
Jerk	10000
BufferMode	BM_ABORTING_E

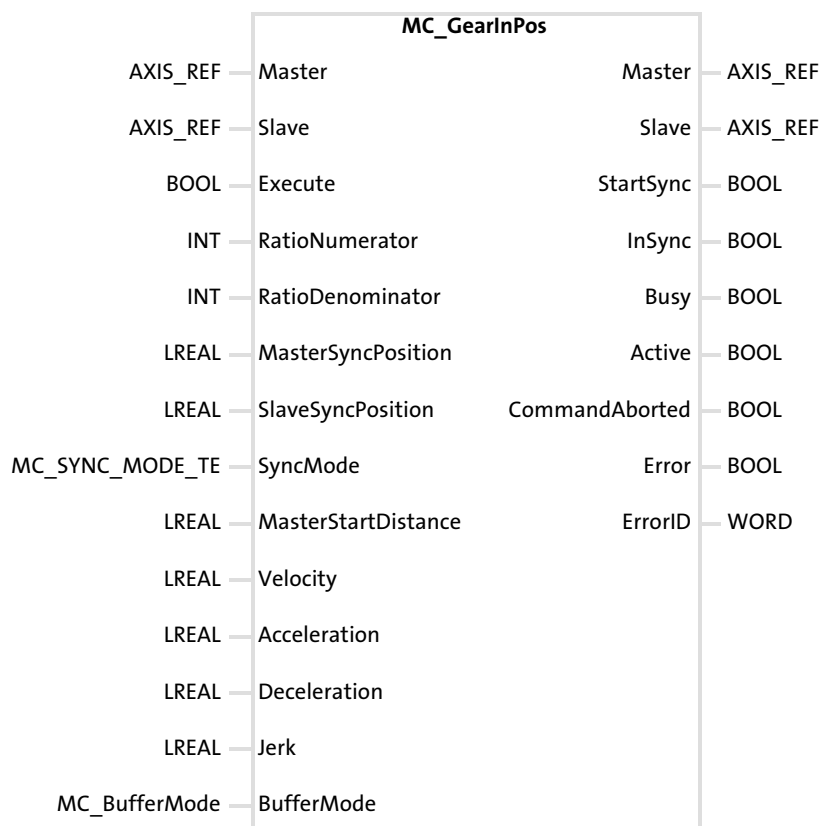
Fehlerbeschreibung

ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
1	Zustand nicht zulässig	Erlaubte Zustände: Standstill, Discrete Motion, Continuous Motion, Synchronized Motion
4	Slave-Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
5	Master-Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
6	Reglerfreigabe der Slave-Achse fehlt	Mit MC_Power muss zunächst Reglerfreigabe gesetzt werden.
8	Acceleration zu groß oder ≤ 0	Acceleration > MAX_ACCELERATION_APPL (206)
9	Deceleration zu groß oder ≤ 0	Deceleration > MAX_DECELERATION_APPL (206)
10	Jerk zu groß oder ≤ 0	Jerk > MAX_JERK_APPL (206)
12	RatioDenominator = 0	
19	Instanz schon für eine andere Slave-Achse aktiv.	
20	Unzulässiger Wert für BufferMode	
24	Für diese Achsensart ist diese Bewegung nicht erlaubt.	
25	Negativer Softwareendschalter	
26	Positiver Softwareendschalter	
100	Achsfehler ist während der Ausführung des Befehls aufgetreten.	Achsfehler ist mit MC_ReadAxisError einzulesen.
101	Systemfehler ist während der Ausführung des Befehls aufgetreten.	Systemfehler ist mit MC_ReadSysError einzulesen.

4.4.9 Funktionsbaustein MC_GearInPos

Aufgabe

Mit diesem Funktionsbaustein wird ein elektronisches Geschwindigkeitsgetriebe zwischen einer Slave- und einer Master-Achse aktiviert, wobei die volle Synchronisation ab vorgegebenen Positionen der Slave- und Master-Achse gewährleistet wird.

Schematische
KOP/FUP-Darstellung

Funktionsbeschreibung

Die aktuelle Bewegung der Slave-Achse wird fortgesetzt, bis die Synchronisationsphase gestartet wird; d. h., bis die Master-Achse die Position **MasterSyncPosition-MasterStartDistance** erreicht.

Wenn während der Synchronisationsphase die über Eingangsparameter **Velocity**, **Acceleration**, **Deceleration** und **Jerk** programmierte Werte überschritten werden, wird im Falle einer Modulo-Master-Achse der **MasterStartDistance** bis zu 5x um den Master-Modulo-Wert (**IrRoad** in Konfigurationsdaten) vergrößert. Wenn alle Versuche misslungen sind, wird ein Fehler gemeldet und die Bewegung nicht ausgeführt.

Bei einer nicht Modulo-Master-Achse wird bei möglicher Verletzung dieser Parameter sofort Fehler gemeldet und die Bewegung nicht ausgeführt.

Die Slave-Achse kann mit **MC_Stop** angehalten werden. Die bestehende Kopplung zwischen Slave- und Master-Achse kann mit **MC_GearOut** aufgehoben werden, wobei die Slave-Achse sich mit der aktuellen Geschwindigkeit weiter im Zustand **ContinuousMotion** bewegen wird.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Master	AXIS_REF	P	Index einer Master-Achse. Wert: 0 ... MaxAxis
Slave	AXIS_REF	P	Index einer Slave-Achse. Wert: 0 ... MaxAxis
VAR_INPUT			
Execute	BOOL	P	Mit steigender Flanke wird die Ausführung der Funktion gestartet.
RatioNumerator	INT	O	Zähler des Getriebeverhältnisses
RatioDenominator	INT	O	Nenner des Getriebeverhältnisses
MasterSyncPosition	LREAL	P	Position der Master-Achse, ab der die Achsen synchron laufen müssen.
SlaveSyncPosition	LREAL	P	Position der Slave-Achse, ab der die Achsen synchron laufen müssen.
SyncMode	MC_SYNC_MODE_TE	O	Nur von Bedeutung, wenn die Slave-Achse eine Modulo-Achse (Rotationsachse) ist. Enum MC_SYNC_MODE_TE: TYPE MC_SYNC_MODE_TE : (SYNC_MODE_SHORTEST_WAY_E:=0, SYNC_MODE_CATCH_UP_E :=1, SYNC_MODE_SLOW_DOWN_E :=2); END_TYPE - Bei SyncMode = SYNC_MODE_SHORTEST_WAY_E wird die Slave-Achse über den kürzesten Weg auf SlaveSyncPosition positioniert. - Bei SyncMode=SYNC_MODE_CATCH_UP_E wird die Slave-Achse über die aktuelle Richtung auf SlaveSyncPosition positioniert. - Bei SyncMode= SYNC_MODE_SLOW_DOWN_E wird die Slave-Achse über die entgegengesetzte Richtung zu der aktuellen Richtung auf SlaveSyncPosition positioniert.
MasterStartDistance	LREAL	P	Abstand der Master vor MasterSyncPosition , auf dem die Slave-Achse mit der Master-Achse synchronisiert wird.
Velocity	LREAL	O	Maximale Geschwindigkeit der Slave-Achse für die Synchronisationsphase in [u/s] (nur positive).
Acceleration	LREAL	O	Maximale Beschleunigung für die Synchronisationsphase in [u/s ²] (nur positiv).
Deceleration	LREAL	O	Sollverzögerung für die Synchronisationsphase in [u/s ²] (nur positiv).
Jerk	LREAL	O	Sollbeschleunigungs-/verzögerungsanstieg (Ruck) für die Synchronisationsphase in [u/s ³] (nur positiv).
VAR_OUTPUT			
StartSync	BOOL	-	Synchronisation wurde gestartet.
InSync	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn die Synchronisation erfolgreich beendet wurde.
Busy	BOOL	-	Wird mit steigender Flanke von Execute gesetzt.
Active	BOOL		Bewegung in Ausführung

Name	Datentyp	P/O ¹⁾	Bedeutung
CommandAborted	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn die Ausführung des Befehls durch eine andere Befehlsanforderung abgebrochen wurde.
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Initialisierungswerte

RatioNumerator	1
RatioDenominator	1
SyncMode	SYNC_MODE_SHORTEST_WAY_E
Velocity	100
Acceleration	1000
Deceleration	1000
Jerk	10000
BufferMode	BM_ABORTING_E

Fehlerbeschreibung

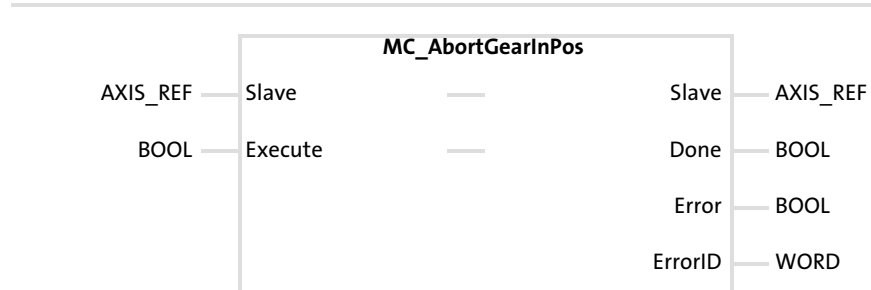
ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
1	Zustand nicht zulässig	Erlaubte Zustände: Standstill, Discrete Motion, Continuous Motion, Synchronized Motion
4	Slave-Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
5	Master-Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
6	Reglerfreigabe der Slave-Achse fehlt	Mit MC_Power muss zunächst Reglerfreigabe gesetzt werden.
7	Velocity zu groß oder < 0	Velocity > MAX_VELOCITY_APPL (206)
8	Acceleration zu groß oder <= 0	Acceleration > MAX_ACCELERATION_APPL (206)
9	Deceleration zu groß oder <= 0	Deceleration > MAX_DECELERATION_APPL (206)
10	Jerk zu groß oder <= 0	Jerk > MAX_JERK_APPL (206)
12	RatioDenominator = 0	
19	Instanz schon für eine andere Slave-Achse aktiv	
25	Negativer Softwareendschalter	
26	Positiver Softwareendschalter	
24	Für diese Achsensart ist diese Bewegung nicht erlaubt.	
100	Achsfehler ist während der Ausführung des Befehls aufgetreten.	Achsfehler ist mit MC_ReadAxisError einzulesen.
101	Systemfehler ist während der Ausführung des Befehls aufgetreten.	Systemfehler ist mit MC_ReadSysError einzulesen.
1110	Geschwindigkeit während der Synchronisationsphase zu groß.	
1111	Beschleunigung während der Synchronisationsphase zu groß.	
1112	Verzögerung während der Synchronisationsphase zu groß.	
1113	Ruck während der Synchronisationsphase zu groß.	
1114	Ausführung der Synchronisationsphase, ohne Geschwindigkeit oder Beschleunigung oder Verzögerung oder Ruck zu verletzen, nicht möglich.	
1115	MasterStartDistance = 0	

4.4.10 Funktionsbaustein MC_AbortGearInPos

Aufgabe

Mit diesem Funktionsbaustein kann der Funktionsbaustein **MC_GearInPos** beim Warten auf die Startposition der Master-Achse deaktiviert werden.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Wenn der Funktionsbaustein **MC_GearInPos** aktiviert wird, befindet er sich zunächst in einer Ausführungs-phase, in der er auf die Startposition von Master-Achse wartet (Ausgangsparameter Busy=TRUE). In dieser Ausführungs-phase kann der Funktionsbaustein **MC_GearInPos** mit dem Funktionsbaustein **MC_AbortGearInPos** deaktiviert werden. Der Funktionsbaustein **MC_GearInPos** zeigt danach über den Ausgangsparameter **CommandAborted**, dass seine Ausführung abgebrochen wurde.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Slave	AXIS_REF	P	Index der Slave-Achse. Wert: 0 ... MaxAxis
VAR_INPUT			
Execute	BOOL	P	Mit steigender Flanke wird die Ausführung der Funktion gestartet
VAR_OUTPUT			
Done	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn der FB MC_GearInPos beim Warten auf die Startposition der Master-Achse abgebrochen wurde.
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Fehlerbeschreibung

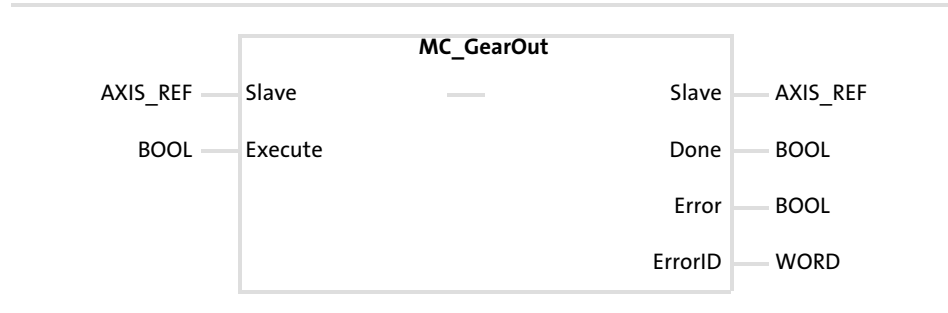
ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
1160	MC_GearInPos für vorgegebene Achse nicht beim Warten auf die Startposition der Master-Achse.	Entweder MC_GearInPos nicht aktiviert oder befindet sich schon mindestens in der Synchronisationsphase.

4.4.11 Funktionsbaustein MC_GearOut

Aufgabe

Mit diesem Funktionsbaustein wird ein elektronisches Getriebe zwischen einer Slave- und einer Master-Achse deaktiviert.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Nach der Deaktivierung der Getriebe geht die Slave-Achse in den Zustand **Continuous Motion** und fährt mit der aktuellen Geschwindigkeit weiter.



Stop!

Nach MC_GearOut befindet sich die Achse in einer Bewegung ohne einen aktiven FB, der diese Bewegung angestoßen hat. Der aktuelle Zustand der Achse kann nur mit dem FB MC_ReadStatus erfasst werden.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Slave	AXIS_REF	P	Index der Slave-Achse. Wert: 0 ... MaxAxis
VAR_INPUT			
Execute	BOOL	P	Mit steigender Flanke wird die Ausführung der Funktion gestartet
VAR_OUTPUT			
Done	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn die Slave-Achse abgekoppelt wurde.
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Fehlerbeschreibung

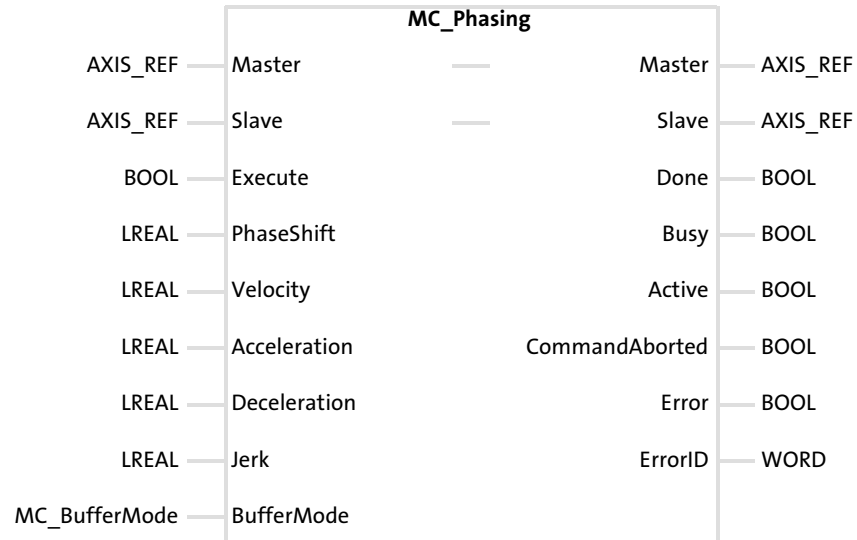
ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
1	Zustand nicht zulässig	Nur im Zustand Synchronized Motion wird der FB ausgeführt.
3	Slave-Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
6	Reglerfreigabe der Achse fehlt	Mit MC_Power muss zunächst Reglerfreigabe gesetzt werden.
100	Achsfehler ist während der Ausführung des Befehls aufgetreten.	Achsfehler ist mit MC_ReadAxisError einzulesen.
101	Systemfehler ist während der Ausführung des Befehls aufgetreten.	Systemfehler ist mit MC_ReadSysError einzulesen.

4.4.12 Funktionsbaustein MC_Phasing

Aufgabe

Dieser Funktionsbaustein erzeugt eine Phasenverschiebung in der Master-Position für eine Slave-Achse.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Dieser Funktionsbaustein erzeugt eine absolute Phasenverschiebung in der Master-Position für die vorgegebene Slave-Achse. Bei Berechnung der Slave-position aus einer CAM-Tabelle wird die Master-Position in Bezug auf die physikalische Position um den aktuellen **PhaseShift** verschoben. Diese Phasenverschiebung bleibt solange erhalten, bis ein neuer Phasing-Befehl das ändert oder MC_ResetPhaseShift auf Null setzt.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Master	AXIS_REF	P	Index einer Master-Achse. Wert: 0 ... MaxAxis
Slave	AXIS_REF	P	Index einer Slave-Achse. Wert: 0 ... MaxAxis
VAR_INPUT			
Execute	BOOL	P	Mit steigender Flanke wird die Phasenverschiebung gestartet.
PhaseShift	LREAL	O	Absolute Phasenverschiebung der Masterposition [u]
Velocity	LREAL	O	Geschwindigkeit zum Erreichen der Phasenverschiebung [u/s]
Acceleration	LREAL	O	Sollbeschleunigung zum Erreichen der Phasenverschiebung [u/s ²] (nur positiv)
Deceleration	LREAL	O	Sollverzögerung zum Erreichen der Phasenverschiebung [u/s ²] (nur positiv)
Jerk	LREAL	O	Sollbeschleunigungs-/verzögerungsanstieg (Ruck) zum Erreichen der Phasenverschiebung [u/s ³] (nur positiv)

Name	Datentyp	P/O ¹⁾	Bedeutung
BufferMode	MC_BufferMode	O	Enum-Typ MC_BufferMode. Gilt nur für die Übergänge zwischen zwei MC_Phasing. <pre> TYPE MC_BufferMode : (BM_ABORTING_E :=0, BM_BUFFERED_E :=1, BM_BLENDING_LOW_E :=2, BM_BLENDING_PREVIOUS_E :=3, BM_BLENDING_NEXT_E :=4, BM_BLENDING_HIGH_E :=5, BM_MASTER_POSITION_E :=6); END_TYPE </pre> Bei BM_ABORTING_E wird die aktuelle Bewegung sofort abgelöst. Bei BM_BUFFERED_E wird die MC-Phasing-Aufgabe erst gestartet, wenn die aktuelle beendet wird. Die Achse wird zwischen beiden Bewegungen angehalten Bei BM_BLENDING_LOW_E wird die MC-Phasing-Aufgabe erst gestartet, wenn die aktuelle beendet wird. Die Achse wird zwischen beiden Bewegungen nicht angehalten. Der Übergang wird mit der kleineren Geschwindigkeit von beiden Bewegungen ausgeführt. Bei BM_BLENDING_PREVIOUS_E wird die MC-Phasing-Aufgabe erst gestartet, wenn die aktuelle beendet wird. Die Achse wird zwischen beiden Bewegungen nicht angehalten. Der Übergang wird mit der Geschwindigkeit der ersten Bewegung ausgeführt. Bei BM_BLENDING_NEXT_E wird die MC-Phasing-Aufgabe erst gestartet, wenn die aktuelle beendet wird. Die Achse wird zwischen beiden Bewegungen nicht angehalten. Der Übergang wird mit der Geschwindigkeit der nächsten Bewegung ausgeführt. Bei BM_BLENDING_HIGH_E wird die MC-Phasing-Aufgabe erst gestartet, wenn die aktuelle beendet wird. Die Achse wird zwischen beiden Bewegungen nicht angehalten. Der Übergang wird mit der höheren Geschwindigkeit von beiden Bewegungen ausgeführt.
VAR_OUTPUT			
Done	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn die Phasenverschiebung erreicht wurde.
Busy	BOOL	-	Befehl in Ausführung Von der Bedeutung vor allem bei Verfahrenskette mit Merge = MERGE_MODE_BLENDING_E
Active	BOOL	-	Bewegung in Ausführung
CommandAborted	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn die Ausführung des Befehls durch einen anderen Befehl abgebrochen wurde.
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Initialisierungswerte

PhaseShift	0.0
Velocity	50
Acceleration	1000
Deceleration	1000
Jerk	10000
BufferMode	BM_ABORTING_E

Fehlerbeschreibung

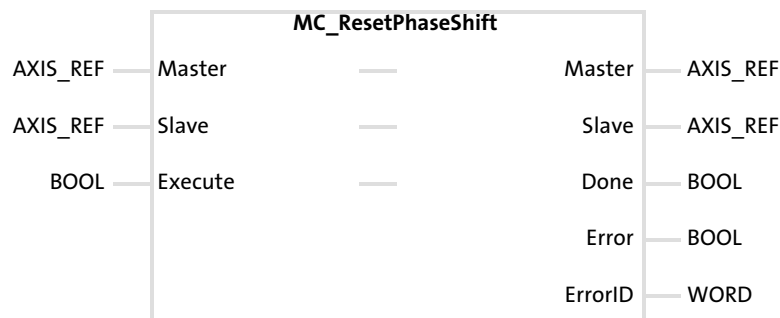
ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
1	Zustand nicht erlaubt	Nur im Zustand Synchronized Motion wird der FB ausgeführt.
3	Slave- oder Master-Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
6	Reglerfreigabe der Slave-Achse fehlt	Mit MC_Power muss zunächst Reglerfreigabe gesetzt werden.
7	VelocityDiff zu groß oder < 0	Velocity > MAX_VELOCITY_APPL (206)
8	Acceleration zu groß oder <= 0	Acceleration > MAX_ACCELERATION_APPL (206)
9	Deceleration zu groß oder <= 0	Deceleration > MAX_DECELERATION_APPL (206)
10	Jerk zu groß oder <= 0	Jerk > MAX_JERK_APPL (206)
19	Instanz schon für eine andere Slave-Achse aktiv	
20	Unzulässiger Wert für BufferMode	
100	Achsfehler ist während der Ausführung des Befehls aufgetreten.	Achsfehler ist mit MC_ReadAxisError einzulesen.
101	Systemfehler ist während der Ausführung des Befehls aufgetreten.	Systemfehler ist mit MC_ReadSysError einzulesen.

4.4.13 Funktionsbaustein MC_ResetPhaseShift

Aufgabe

Dieser Funktionsbaustein setzt die mit dem Funktionsbaustein **MC_Phasing** erzeugte Phasenverschiebung der Masterposition auf Null.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Der Funktionsbaustein **MC_Phasing** erzeugt eine Phasenverschiebung in der Master-Position für die vorgegebene Slave-Achse. Bei Berechnung der Slave-Position aus einer CAM-Tabelle wird die Master-Position in Bezug auf die physikalische Position um den aktuellen "PhaseShift" verschoben. Diese Phasenverschiebung bleibt solange erhalten, bis ein neuer Phasing-Befehl das ändert oder mit dem Funktionsbaustein **MC_ResetPhaseShift** auf Null gesetzt wird. Der Funktionsbaustein kann nur im Achszustand "Stand-Still" ausgeführt werden.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Master	AXIS_REF	P	Index der Master-Achse. Wert: 0 ... MaxAxis
Slave	AXIS_REF	P	Index der Slave-Achse. Wert: 0 ... MaxAxis
VAR_INPUT			
Execute	BOOL	P	Mit steigender Flanke wird die Ausführung der Funktion gestartet
VAR_OUTPUT			
Done	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn die Phasenverschiebung auf Null gesetzt wurde.
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Fehlerbeschreibung

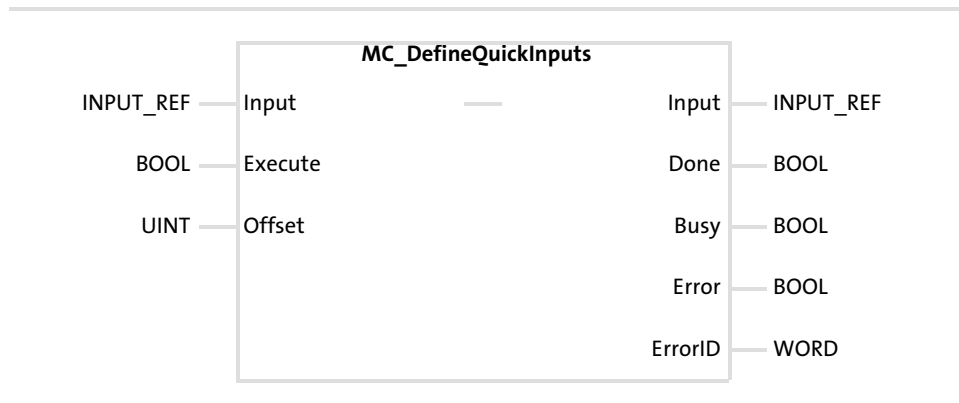
ErrorID	Bedeutung	Hinweis
0	Kein Fehler	
1	Zustand nicht erlaubt	Nur im Zustand "Synchronized Motion" wird der FB aus-geführt.
3	Slave- oder Master-Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.

4.5 Funktionsbausteine für schnelle Ein-/Ausgänge

4.5.1 Funktionsbaustein MC_DefineQuickInputs

Aufgabe Mit diesem Funktionsbaustein können digitale Eingänge dem Interpolator zur Verfügung gestellt werden.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Maximal 32 digitale Eingänge (d. h. 2 Eingangsmodul von jeweils bis zu 16 digitalen Eingängen) können dem Interpolator zur Verfügung gestellt werden. Über den Parameter **INPUT_REF** wird das entsprechende Eingangsmodul ausgewählt.

Die Adressen der digitalen Eingänge in den Prozessdaten können mit dem Eingangsparameter **Offset** eingestellt werden.

Ab dem Offset werden immer 16 Eingänge nacheinander in Betracht genommen.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Input	INPUT_REF	P	Referenz auf ein I/O-Modul (0 oder 1)
VAR_INPUT			
Execute	BOOL	P	Mit steigender Flanke wird die Funktion gestartet.
Offset	UINT	O	Anfangsindex in Prozessdaten für Eingangsmodul
VAR_OUTPUT			
Done	BOOL	-	Kommando ausgeführt
Busy	BOOL	-	TRUE = Funktionsbaustein ist aktiv
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Initialisierungswerte

Offset	0, 0
--------	------

Fehlerbeschreibung

Fehler-Nr.	Bedeutung	Hinweis
0	Kein Fehler	-
1060	Unzulässiger Wert für Input	> 1
1061	Unzulässiger Wert für Offset	> 2047

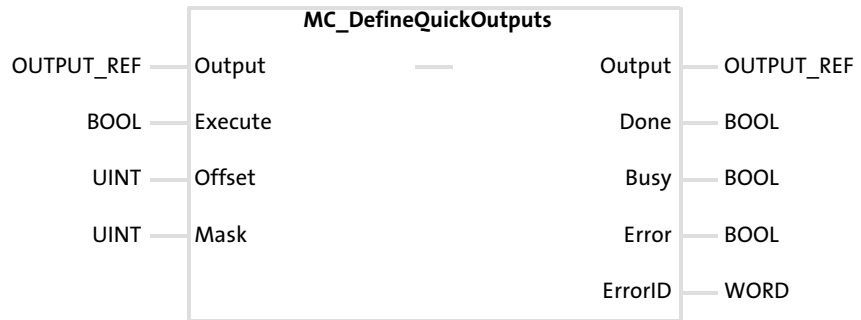
4.5.2 Funktionsbaustein MC_DefineQuickOutputs

Aufgabe

Mit diesem Funktionsbaustein können dem Interpolator digitale Ausgänge zur Verfügung gestellt werden.

Ausgänge von 2 Ausgangsmodulen von jeweils bis zu 16 digitalen Ausgängen können dem Interpolator freigegeben werden.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Maximal 32 digitale Ausgänge (d. h. 2 Ausgangsmodule von jeweils bis zu 16 digitalen Ausgängen) können dem Interpolator zur Verfügung gestellt werden. Über den Parameter **OUTPUT_REF** wird das entsprechende Ausgangsmodul ausgewählt.

Die Adressen der digitalen Ausgänge in Prozessdaten können mit dem Eingangsparameter **Offset** eingestellt werden.

Ab dem Offset werden immer 16 Ausgänge nacheinander in Betracht genommen. Welche von den 16 Ausgängen dem Interpolator als schnelle Ausgänge zur Verfügung stehen, kann man mit dem Eingangsparameter **Mask** bestimmen.

Ein entsprechendes Bit in der Maske gleich 1 bedeutet, dass der Ausgang dem Interpolator freigegeben ist; d.h. der Interpolator kann den Ausgang auf 0 oder 1 setzen.

Ein entsprechendes Bit in der Maske gleich 0 bedeutet, dass der Ausgang der SPS zur Verfügung steht; d.h. der Interpolator kann den Ausgang nicht setzen.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Output	OUTPUT_REF	P	Referenz auf ein I/O-Modul (0 oder 1)
VAR_INPUT			
Execute	BOOL	P	Mit steigender Flanke wird die Funktion gestartet.
Offset	UINT	O	Anfangsindex in Prozessdaten für Ausgangsmodul 0 und 1.
Mask	UINT	O	Freigabemaske für Ausgangsmodul 0 und 1. Eine 1 gibt dem Interpolator den Zugriff auf den entsprechenden Ausgang.
VAR_OUTPUT			
Done	BOOL	-	Kommando ausgeführt
Busy	BOOL	-	TRUE = Funktionsbaustein ist aktiv
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Initialisierungswerte

Offset	0, 0
Mask	0,0

Fehlerbeschreibung

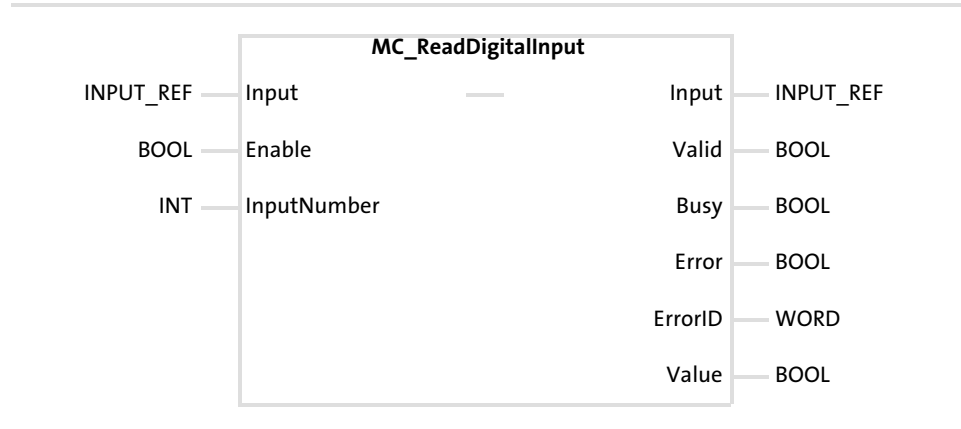
Fehler-Nr.	Bedeutung	Hinweis
0	Kein Fehler	-
1060	Unzulässiger Wert für Input	> 1
1061	Unzulässiger Wert für Offset	> 2047

4.5.3 Funktionsbaustein MC_ReadDigitalInput

Aufgabe

Mit diesem Funktionsbaustein kann der Zustand eines schnellen digitalen Eingangs gelesen werden.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Solange der Eingangsparameter **Enable** = TRUE, wird der Ausgangsparameter **Value** mit dem aktuellen Zustand des vorgegebenen Eingangs zyklisch versorgt. Bei **Enable** = FALSE wird der Ausgangsparameter **Value** auf FALSE gesetzt.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Input	INPUT_REF	P	Referenz auf ein I/O-Modul (0 oder 1)
VAR_INPUT			
Enable	BOOL	P	TRUE = Parameterwert wird kontinuierlich ausgegeben.
InputNumber	INT	P	Eingangsnummer
VAR_OUTPUT			
Valid	BOOL	-	TRUE = Zustand des vorgegebenen Eingangs wurde über den Ausgang Value ausgegeben.
Busy	BOOL	-	TRUE = Funktionsbaustein ist aktiv.
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode
Value	LREAL	-	Zustand des spezifizierten Eingangs

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Fehlerbeschreibung

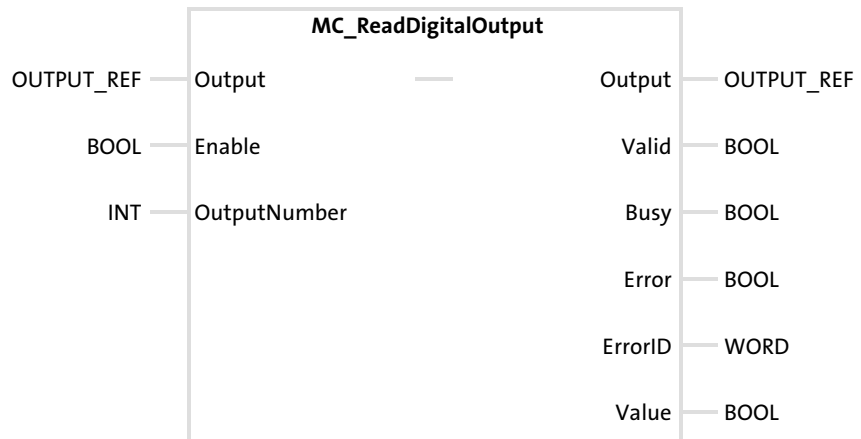
Fehler-Nr.	Bedeutung	Hinweis
0	Kein Fehler	-
1060	Unzulässiger Wert für Input	> 1
1062	Unzulässiger Wert von InputNumber	> 15
1063	Digitaler Eingang nicht als schneller Ausgang freigegeben.	

4.5.4 Funktionsbaustein MC_ReadDigitalOutput

Aufgabe

Mit diesem Funktionsbaustein kann der Zustand eines schnellen digitalen Ausgangs gelesen werden.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Solange der Eingangsparameter **Enable** = TRUE, wird der Ausgangsparameter **Value** mit dem aktuellen Zustand des vorgegebenen Ausgangs zyklisch versorgt. Bei **Enable** = FALSE wird der Ausgangsparameter **Value** auf FALSE gesetzt.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Output	OUTPUT_REF	P	Referenz auf ein I/O-Modul (0 oder 1)
VAR_INPUT			
Enable	BOOL	P	TRUE = Parameterwert wird kontinuierlich ausgegeben
OutputNumber	INT	P	Ausgangsnummer
VAR_OUTPUT			
Valid	BOOL	-	TRUE = Zustand des vorgegebenen Ausgangs wurde über den Ausgang Value ausgegeben
Busy	BOOL	-	TRUE = Funktionsbaustein ist aktiv
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode
Value	LREAL	-	Zustand des spezifizierten Ausgangs

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Fehlerbeschreibung

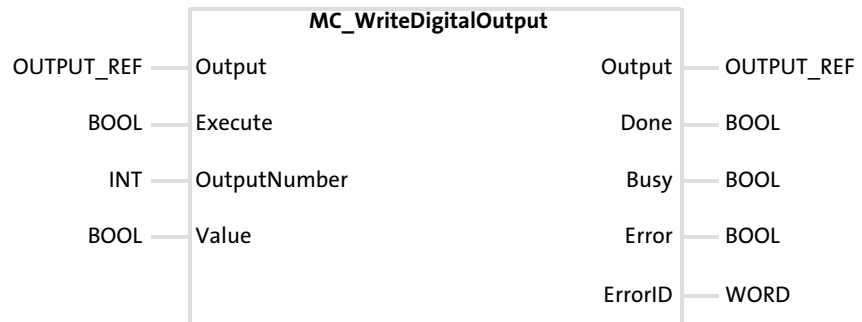
Fehler-Nr.	Bedeutung	Hinweis
0	Kein Fehler	-
1060	Unzulässiger Wert für Output	> 1
1062	Unzulässiger Wert von OutputNumber	> 15
1063	Digitaler Ausgang nicht als schneller Ausgang freigegeben	

4.5.5 Funktionsbaustein MC_WriteDigitalOutput

Aufgabe

Mit diesem Funktionsbaustein kann der Zustand eines schnellen digitalen Ausganges auf den vorgegebenen Wert gesetzt werden.

Schematische KOP/FUP-Darstellung



Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Output	OUTPUT_REF	P	Referenz auf ein I/O-Modul (0 oder 1)
VAR_INPUT			
Execte	BOOL	P	Mit steigender Flanke wird der vorgegebene Zustand geschrieben.
OutputNumber	INT	P	Ausgangsnummer
Value	BOOL	P	Erforderter Zustand des Ausganges
VAR_OUTPUT			
Done	BOOL	-	Schreiben des vorgegebenen Zustands auf den vorgegebenen Ausgang wurde abgeschlossen
Busy	BOOL	-	TRUE = Funktionsbaustein ist aktiv
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn bei der Ausführung dieses Befehls ein Fehler aufgetreten ist.
ErrorID	WORD	-	Fehlercode

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Fehlerbeschreibung

Fehler-Nr.	Bedeutung	Hinweis
0	Kein Fehler	-
1060	Unzulässiger Wert für Output	> 1
1062	Unzulässiger Wert von OutputNumber	> 15
1063	Digitaler Ausgang nicht als schneller Ausgang freigegeben	

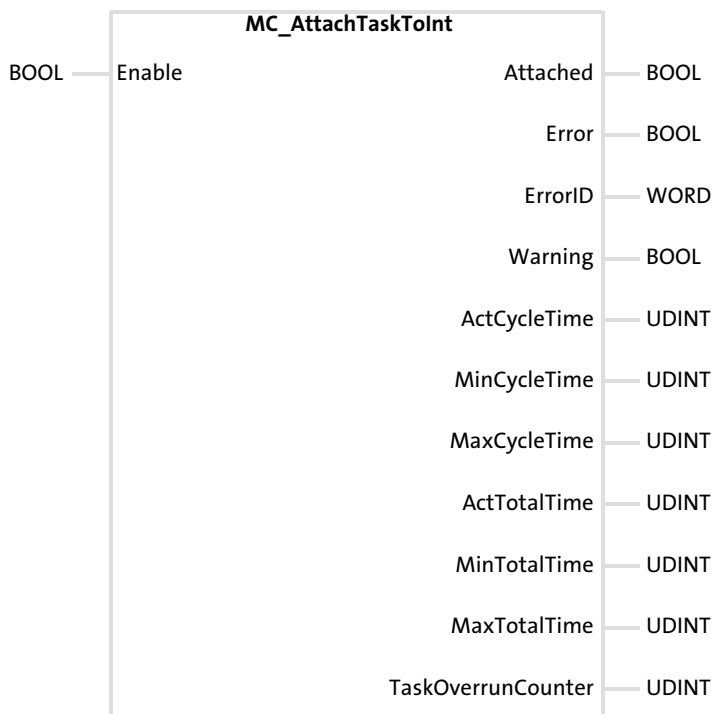
4.6 Support-Funktionsbausteine

4.6.1 Funktionsbaustein MC_AttachTaskToInt

Aufgabe

Mit diesem Funktionsbaustein wird die den Baustein aufrufende Task direkt an den Interpolator angekoppelt, d. h. die Task wird immer am Ende vom Interpolator aufgerufen und ohne Unterbrechung durch die anderen PLC-Tasks oder System-Task ausgeführt.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Die den Funktionsbaustein aufrufende PLC-Task wird bei steigender Flanke am Eingangsparameter **Enable** an den Interpolator angekoppelt. Das heißt, die Task wird am Ende des Interpolatorzyklus aufgerufen.

Die Priorität dieser Task wird für die Zeit der Ankopplung auf die höchste Priorität im System gesetzt, so dass die Task nie durch eine andere PLC-Task oder System-Task unterbrochen wird. Das gewährleistet, dass die Task einmal innerhalb des Interpolationstakts ausgeführt wird, wenn der eigene Rechenbedarf nicht zu groß ist. Die über den Ausgangsparameter **ActCycleTime** angezeigte Zykluszeit muss gleich dem Interpolationstakt sein.

Beachten Sie, dass die Summe der Laufzeit vom Interpolator und der Laufzeit der an Interpolator angekoppelten PLC-Task nicht zu nahe an den Interpolationstakt kommt. Die anderen PLC-Task sowie System-Task müssen auch ausgeführt werden.

Wenn die Zykluszeit (Ausgangsparameter **ActCycleTime**) durch die Laufzeit (Ausgangsparameter **ActTotalTime**) mindestens einmal seit der Ankopplung überschritten wurde, dann wird dieses über den Ausgangsparameter

Warning signalisiert. Wenn das mehr als 5 mal nacheinander vorkommt, wird die Task vom Interpolator entkoppelt und eine Fehlermeldung ausgegeben.

Bei fallender Flanke am Eingangsparameter **Enable** wird die Task vom Interpolator entkoppelt und wieder, wie vor der Ankopplung, mit eigenen konfigurierten Parametern (Priorität, Zykluszeit) laufen.

Der Baustein soll in der Task aufgerufen werden, die die zeitkritischen Aufgaben in Verbindung mit dem Interpolator zu erledigen hat, z. B. Lesen von Zuständen mehrerer Achsen aus einem Interpolationstakt oder der Start einer Aufgabe, abhängig von Parametern aus dem Interpolator.

Der Funktionsbaustein kann nur zur Ausführung von solchen Aufgaben aktiviert werden und danach wieder deaktiviert werden. Der Funktionsbaustein liefert immer **ActCycleTime**, **MinCycleTime**, **MaxCycleTime** und **ActTotalTime**, **MinTotalTime**, **MaxTotalTime** der den Funktionsbaustein aufrufenden Task, unabhängig vom Zustand des Eingangsparameters **Enable**.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_INPUT			
Enable	BOOL	P	Bei steigender Flanke wird die Task an den Interpolator angekoppelt. Bei fallender Flanke wird die Task vom Interpolator entkoppelt und weiter mit den eigenen konfigurierten Parametern laufen.
VAR_OUTPUT			
Attached	BOOL	-	Die Task ist am Interpolator angekoppelt.
Error	BOOL	-	Fehler wird gemeldet.
ErrorID	WORD	-	Fehlercode
Warning	BOOL	-	Gleich TRUE , wenn mindestens einmal seit der Ankopplung TotalTime > CycleTime
CycleTimeAct	UDINT	-	Aktuelle Zykluszeit der Task ≈ Interpolationstakt
CycleTimeMin	UDINT	-	Minimale Zykluszeit während der letzten Sekunde
CycleTimeMax	UDINT	-	Maximale Zykluszeit während der letzten Sekunde
TotalTimeAct	UDINT	-	Aktuelle Ausführungszeit der Task
TotalTimeMin	UDINT	-	Minimale Ausführungszeit der Task während der letzten Sekunde
TotalTimeMax	UDINT	-	Maximale Ausführungszeit der Task während der letzten Sekunde
TaskOverrunCounter	UDINT	-	Der Ausgangsparameter wird inkrementiert, wenn die aktuelle Ausführungszeit größer als die aktuelle Zykluszeit ist. Der Zähler wird bei steigender und fallender Flanke von dem Eingangsparameter Enable auf Null zurückgesetzt.

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Fehlerbeschreibung

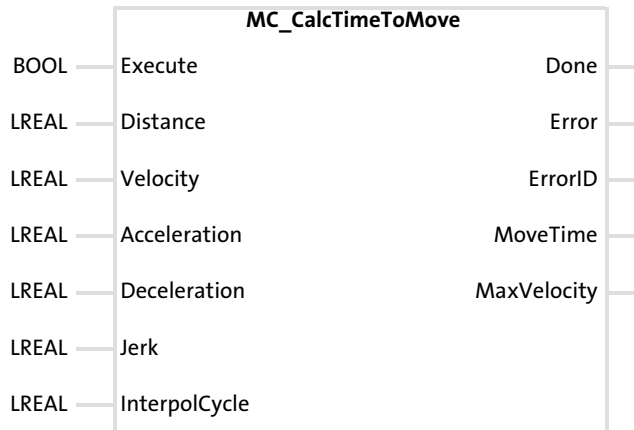
Fehler-Nr.	Bedeutung	Hinweis
0	Kein Fehler	
1070	Eine andere PLC-Task ist schon am Interpolator angekoppelt.	
1071	Nach der Ankopplung der Task an den Interpolator ist es 5 mal nacheinander zum Overrun der Task gekommen.	Mit dieser Fehlermeldung wird die Task vom Interpolator entkoppelt.

4.6.2 Funktionsbaustein MC_CalcTimeToMove

Aufgabe

Mit diesem Funktionsbaustein wird die Bewegungszeit berechnet.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Der Baustein berechnet die Dauer einer mit Eingangsparameter vorgegebenen absoluten Bewegung. Weiterhin liefert er auch die maximal erreichbare Geschwindigkeit während der Bewegung über den Ausgangsparameter **Max-Velocity**.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_INPUT			
Execute	BOOL	P	Bei steigender Flanke wird die Berechnung gestartet.
Distance	LREAL	P	Distanz/Strecke in [u] (negativ oder positiv)
Velocity	LREAL	P	Sollgeschwindigkeit in [u/s] (nur positiv)
Acceleration	LREAL	P	Sollbeschleunigung in [u/s ²] (nur positiv)
Deceleration	LREAL	P	Sollverzögerung in [u/s ²] (nur positiv)
Jerk	LREAL	P	Sollbeschleunigungs-/verzögerungsanstieg (Ruck) in [u/s ³] (nur positiv)
InterpolCycle	LREAL	P	Interpolationstakt in [ms]. Wenn = 0, wird der konfigurierte Interpolationstakt für die Berechnung eingesetzt.
VAR_OUTPUT			
Done	BOOL	-	Berechnung ist abgeschlossen.
Error	BOOL	-	Fehlerhafte Eingangsparameter oder Fehler bei der Berechnung
ErrorID	WORD	-	Fehlercode
MoveTime	TIME	-	Dauer der Bewegung
MaxVelocity	LREAL	-	Maximal erreichbare Geschwindigkeit der Bewegung

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Fehlerbeschreibung

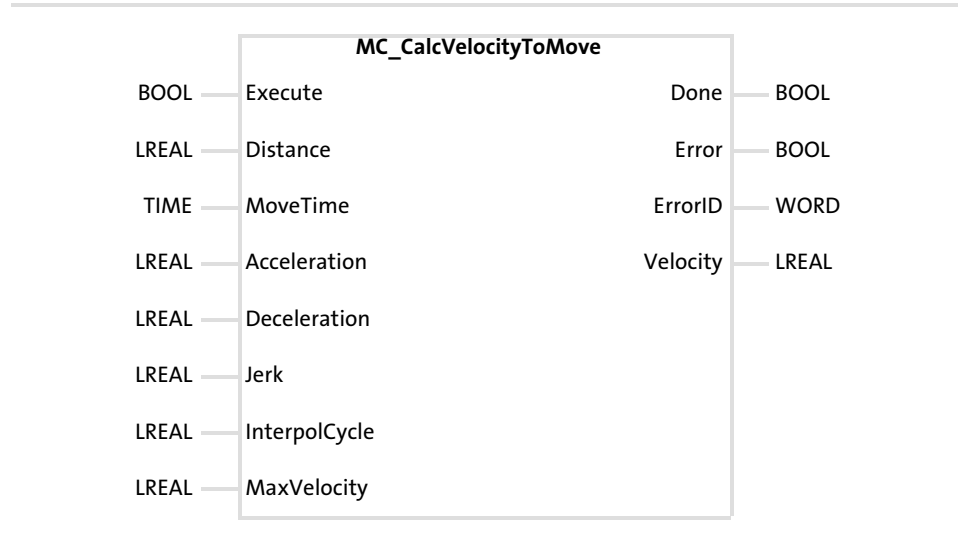
Fehler-Nr.	Bedeutung	Hinweis
0	Kein Fehler	-
1002	Eingangsparameter Velocity ≤ 0	
1003	Eingangsparameter Acceleration ≤ 0	
1004	Eingangsparameter Deceleration_d ≤ 0	
1005	Eingangsparameter Jerk ≤ 0	
1150	Berechnung nicht möglich	Die Bewegung ist bei vorgegebenen Parametern nicht möglich.
1151	Eingangsparameter Distance ≤ 0	

4.6.3 Funktionsbaustein MC_CalcVelocityToMove

Aufgabe

Mit diesem Funktionsbaustein wird die notwendige Geschwindigkeit einer Bewegung bei vorgegebener Zeit der Bewegung berechnet.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Der Baustein berechnet die notwendige Sollgeschwindigkeit einer absoluten Bewegung, die die vorgegebene Strecke in der vorgegebenen Zeit ablegen soll.

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_INPUT			
Execute	BOOL	P	Bei steigender Flanke wird die Berechnung gestartet.
Distance	LREAL	P	Distanz/Strecke in [u] (negativ oder positiv)
MoveTime	TIME	P	Dauer der Bewegung
Acceleration	LREAL	P	Sollbeschleunigung in [u/s ²] (nur positiv)
Deceleration	LREAL	P	Sollverzögerung in [u/s ²] (nur positiv)
Jerk	LREAL	P	Sollbeschleunigungs-/verzögerungsanstieg (Ruck) in [u/s ³] (nur positiv)
InterpolCycle	LREAL	P	Interpolationstakt in [ms]. Wenn = 0, wird der konfigurierte Interpolationstakt für die Berechnung eingesetzt.
MaxVelocity	LREAL	P	Maximal erlaubte Geschwindigkeit der Bewegung
VAR_OUTPUT			
Done	BOOL	-	Berechnung ist abgeschlossen.
Error	BOOL	-	Fehlerhafte Eingangsparameter oder Fehler bei der Berechnung
ErrorID	WORD	-	Fehlercode
Velocity	LREAL	-	Notwendige Sollgeschwindigkeit der Bewegung

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Fehlerbeschreibung

Fehler-Nr.	Bedeutung	Hinweis
0	Kein Fehler	-
1002	Eingangsparameter Velocity ≤ 0	
1003	Eingangsparameter Acceleration ≤ 0	
1004	Eingangsparameter Deceleration_d ≤ 0	
1005	Eingangsparameter Jerk ≤ 0	
1150	Berechnung nicht möglich	Die Bewegung ist bei vorgegebenen Parametern nicht möglich.
1151	Eingangsparameter Distance ≤ 0	

4.7 Kundenspezifische Funktionsbausteine



Hinweis!

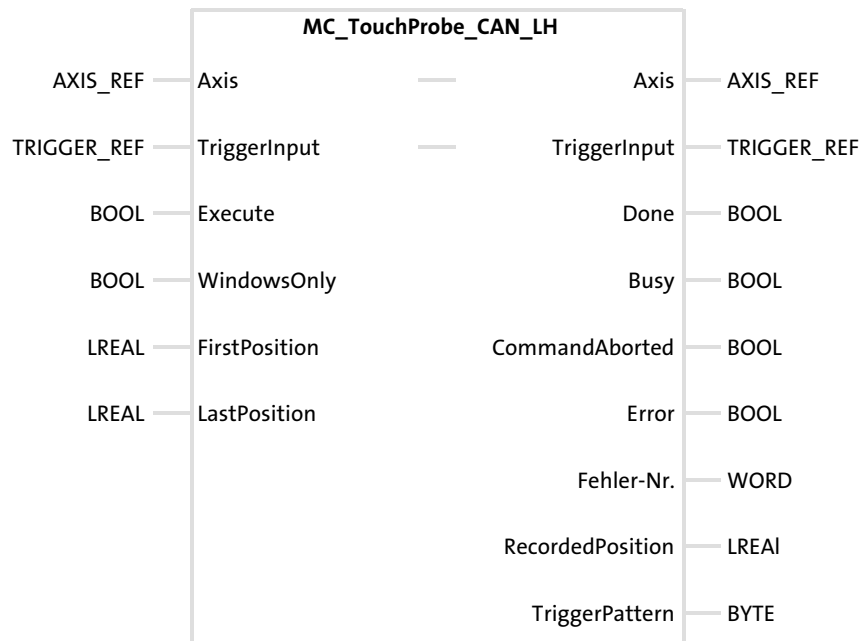
Namen von Funktionsbausteinen, die antriebsbus- und antriebsherstellerspezifische Eigenschaften haben, werden um Antriebsbus- und Antriebshersteller-Kürzel ergänzt.

4.7.1 Funktionsbaustein MC_TouchProbe_CAN_LH

Aufgabe

Der Funktionsbaustein ist äquivalent zum Funktionsbaustein MC_TouchProbe mit der Ausnahme: Wenn die Position im CAN-Antrieb aufgenommen werden soll, können nur die Servoantriebe der Firma Lenze benutzt werden.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung**Vordefinierte Datentypen in mc.lib**

```

TYPE INPUT_REF : WORD; END_TYPE
TYPE TRIGGER_REF :
STRUCT
    Inputs                : INPUT_REF; // Referenz auf Eingänge:
                                // 0- schnelle Eingänge des Interpolators:
                                // Modul 0
                                // 1 - schnelle Eingänge des Interpolators:
                                // Modul 1
                                // 2 - Antriebsmodul mit CAN-Bus
                                // 3- Antriebsmodul mit analoger Schnittstelle
                                // (LBM-ARI)
    InputNumber            : BYTE;      // Nummer von digitalen Triggereingang
                                // (0-15)
    TriggerNumber          : BYTE;      // Triggernummer (0-2)
    TriggerPattern         : BYTE;      // 0-steigende Flanke; 1-fallende Flanke;
                                // 2-beide
    TriggerObjectDataIdx   : WORD;      // Codestelle 6001
    TriggerDoneObjectDataIdx : WORD;    // Codestelle 6002
    RecPosObjectDataIdx    : WORD;      // Codestelle 6000
    AxisPeriodeObjectDataIdx : WORD;    // noch nicht vereinbart
    FirstPosObjectDataIdx  : WORD;      // noch nicht vereinbart
    LastPosObjectDataIdx   : WORD;      // noch nicht vereinbart
END_STRUCT
END_TYPE

```

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Axis	AXIS_REF	P	Index einer Achse. Wert: 0 ... MaxAxis
TriggerInput	TRIGGER_REF	P	Referenz auf Triggerbeschreibung
VAR_INPUT			
Execute	BOOL	P	Mit steigender Flanke wird der Trigger aktiviert.
WindowsOnly	BOOL	O	Triggerereignis nur innerhalb vom vorgegebenen Fenster akzeptieren.
FirstPosition	LREAL	O	Startposition des Fensters, ab welcher (inclusive) das Triggerereignis akzeptiert werden soll in [u].
LastPosition	LREAL	P	Endposition des Fensters, bis welcher (inclusive) das Triggerereignis akzeptiert werden soll in [u].
VAR_OUTPUT			
Done	BOOL	-	TRUE zeigt, dass die Position aufgenommen ist. Der gültige Wert ist über den Ausgangsparameter RecordedPosition geliefert.
Busy	BOOL	-	Funktion in Ausführung
CommandAborted	BOOL	-	TRUE zeigt, dass die Ausführung der Funktion durch MC_AbortTrigger_CAN_LH abgebrochen wurde.
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn ein Fehler bei der Ausführung dieses Befehls aufgetreten ist.
ErrorID	WORD	-	Fehlercode
RecordedPosition	LREAL	-	Aufgenommene Position
TriggerPattern	Byte	-	Triggerpattern, bei dem die Position aufgenommen ist 0 – steigende Flanke, 1 – fallende Flanke, 2 – beide Flanken

¹⁾ P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Initialisierungswerte

WindowsOnly	FALSE
FirstPosition	0

Fehlerbeschreibung

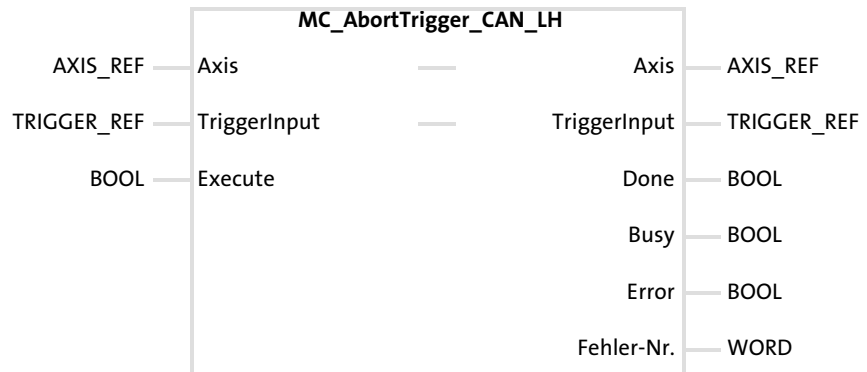
Fehler-Nr.	Bedeutung	Hinweis
0	Kein Fehler	
1	Unzulässiger Zustand der Achse	Achse noch nicht initialisiert oder im Fehlerzustand
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
19	Diese Instanz ist für eine andere Achse aktiv	
1064	Unzulässiger Wert für TriggerNumber	TriggerNumber > 2
1066	Unzulässiger Wert für TriggerInput	Inputs > 3
1067	Unzulässiger Wert für TriggerPattern	TriggerPattern > 2
1068	Datentransfer bei SDO abgebrochen	

4.7.2 Funktionsbaustein MC_Abort_Trigger_CAN_LH

Aufgabe

Der Funktionsbaustein ist äquivalent zu dem Funktionsbaustein MC_Abort_Trigger mit der Ausnahme: Wenn die Position im CAN-Antrieb aufgenommen werden soll, können nur Servoantriebe der Firma Lenze benutzt werden, die über CAN-Drivebus an die Steuerung angeschlossen sind.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Vordefinierte Datentypen in mc.lib

```

TYPE INPUT_REF : WORD; END_TYPE
TYPE TRIGGER_REF :
STRUCT
    Inputs                : INPUT_REF; // Referenz auf Eingänge:
                                // 0- schnelle Eingänge des Interpolators:
                                // Modul 0
                                // 1 - schnelle Eingänge des Interpolators:
                                // Modul 1
                                // 2 - Antriebsmodul mit CAN-Bus
                                // 3 - Antriebsmodul mit analoger
                                // Schnittstelle (LBM-ARI)

    InputNumber            : BYTE;      // Nummer von digitalen Triggereingang
                                // (0-15)

    TriggerNumber          : BYTE;      // Triggernummer (0-2)

    TriggerPaOttern        : BYTE;      // 0-steigende Flanke; 1-fallende Flanke;
                                // 2-beide

    TriggerObjectDataIdx   : WORD;      // Codestelle 6001

    TriggerDoneObjectDataIdx : WORD;     // Codestelle 6002

    RecPosObjectDataIdx    : WORD;      // Codestelle 6000

    AxisPeriodeObjectDataIdx : WORD;     // noch nicht vereinbart

    FirstPosObjectDataIdx  : WORD;      // noch nicht vereinbart

    LastPosObjectDataIdx   : WORD;      // noch nicht vereinbart
END_STRUCT
END_TYPE

```

Parameter

Name	Datentyp	P/O ¹⁾	Bedeutung
VAR_IN_OUT			
Axis	AXIS_REF	P	Index einer Achse. Wert: 0 ... MaxAxis
TriggerInput	TRIGGER_REF	P	Referenz auf Triggerbeschreibung
VAR_INPUT			
Execute	BOOL	P	Mit steigender Flanke wird die aktive Touch-Probe Funktion deaktiviert.
VAR_OUTPUT			
Done	BOOL	-	TRUE zeigt, dass die Touch Probe deaktiviert wurde.
Busy	BOOL	-	Funktion in Ausführung
Error	BOOL	-	Dieser Ausgang wird auf TRUE gesetzt, wenn ein Fehler bei der Ausführung dieses Befehls aufgetreten ist.
ErrorID	WORD	-	Fehlercode

1) P = Pflichtparameter, O = optionaler Parameter (vorinitialisiert)

Fehlerbeschreibung

Fehler-Nr.	Bedeutung	Hinweis
0	Kein Fehler	
1	Unzulässiger Zustand der Achse	Achse noch nicht initialisiert oder im Fehlerzustand.
3	Achse nicht konfiguriert	Achse wurde mit MC_DefineAxis nicht definiert.
19	Diese Instanz ist für eine andere Achse aktiv.	
1064	Unzulässiger Wert für TriggerNumber	TriggerNumber > 2
1066	Unzulässiger Wert für TriggerInput	Inputs > 3

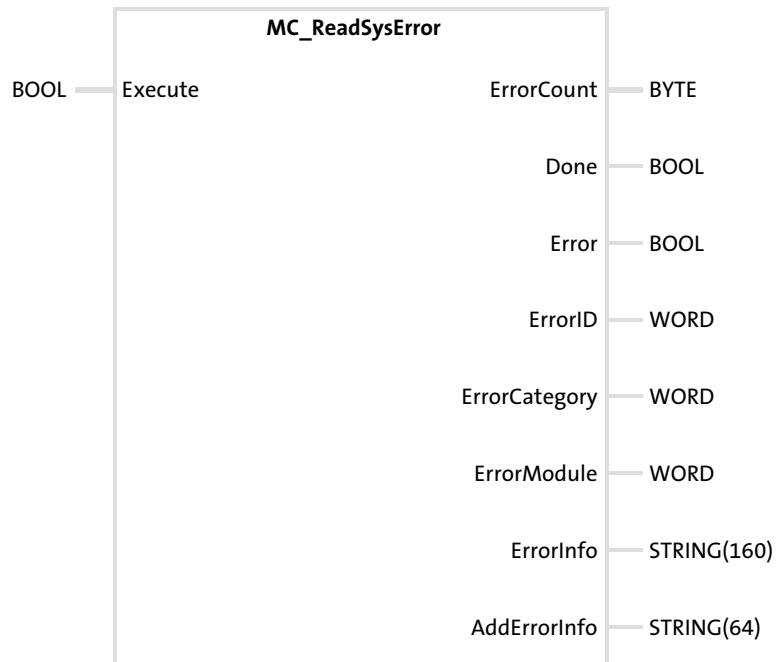
4.8 Funktionsbausteine zum Fehlerhandling

4.8.1 Funktionsbaustein MC_ReadSysError

Aufgabe

Mit diesem Funktionsbaustein wird ein Systemfehler angezeigt.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Bei **ErrorCategory = 4** werden alle Achsen angehalten und in den Zustand **SysErrorStop** versetzt.

Bei **ErrorCategory = 3** wird das weitere Vorgehen der SPS überlassen. Grundsätzlich sollten in dem Fall auch alle Achsen angehalten werden, aber über die Reihenfolge und andere Maßnahmen soll die SPS selbst entscheiden.

Fehler der **ErrorCategory = 1** und **2** sind als Hinweise oder Warnungen zu betrachten. Fehler der **ErrorCategory = 1 ... 3** führen nicht zum Abbruch der laufenden Bewegung und Übergang der Achse in den Zustand **SysErrorStop**. Sie müssen also nicht quittiert werden und werden nur über einen SPS-Zyklus angezeigt.

Fehler **ErrorCategory = 4** wird solange angezeigt, bis er mit **MC_ResetSysError** quittiert wird.

Parameter

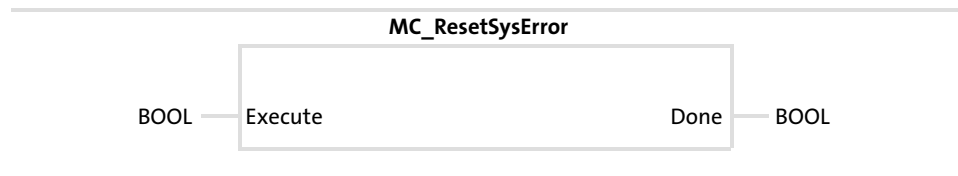
Name	Datentyp	Bedeutung
VAR_INPUT		
Enable	BOOL	TRUE = Systemfehler anzeigen
VAR_OUTPUT		
ErrorCount	BYTE	Anzahl von anstehenden Fehler Kategorie 4
Done	BOOL	TRUE = Funktion ausgeführt
Error	BOOL	TRUE = über den Ausgang ErroID wird ein Fehlercode angezeigt. FALSE = kein Fehler wird angezeigt
ErrorID	WORD	Fehlercode
ErrorCategory	WORD	Fehlerklasse: 1: Lokal leicht 2: Lokal schwer 3: Global leicht 4: Global schwer
ErrorModule	WORD	Modul (Subsystem)
ErrorInfo	STRING(160)	Fehlerbeschreibung
AddErrorInfo	STRING(64)	Zusatzinformation

4.8.2 Funktionsbaustein MC_ResetSysError

Aufgabe

Mit diesem Funktionsbaustein wird die anstehende Systemfehlermeldung quittiert.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Alle Systemfehler werden in einem FIFO-Puffer gespeichert und müssen nacheinander mit dem **MC_ResetSysError** quittiert werden. Nach dem Quittieren des zuletzt gelesenen Systemfehlers, kann der nächste Systemfehler gelesen werden.

Wenn sich Achsen im Zustand **SysErrorStop** befinden, werden sie in den Zustand **Standstill** versetzt, wenn keine weiteren Fehler von der Category 4 im FIFO-Puffer anstehen.

Parameter

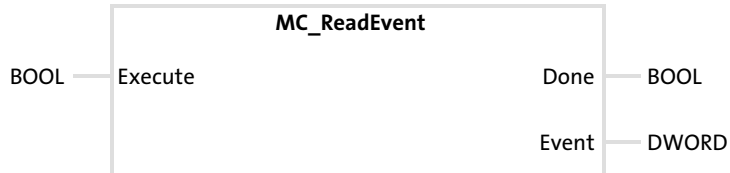
Name	Datentyp	Bedeutung
VAR_INPUT		
Execute	BOOL	Mit positiver Flanke wird der Fehler quittiert.
VAR_OUTPUT		
Done	BOOL	TRUE = Zustand StandStill wurde erreicht FALSE = Achse bleibt im Zustand SysErrorStop , weitere Fehler stehen an.

4.8.3 Funktionsbaustein MC_ReadEvent

Aufgabe

Der Funktionsbaustein **MC_ReadEvent** liefert Information darüber, was für ein Ereignis den Aufruf der Alarm-Task verursacht hat.

Schematische KOP/FUP-Darstellung



Funktionsbeschreibung

Alle gemeldeten Fehler, unabhängig von der Fehlerklasse, lösen an die PLC-Task, die als PLC_PRG4 in der SPS konfiguriert wird, ein Event (Signal) aus. Demzufolge wird diese Task, außerhalb des normalen Zyklus, sofort ausgeführt.

Mit MC_ReadEvent kann die Art des Events festgestellt werden.

Wenn Event = SIG_AXE_ERROR, muss mit dem **MC_ReadAxisError** festgestellt werden, welche Achse den Fehler gemeldet hat. Die betroffene Achse befindet sich im Zustand **Errorstop**, wenn Fehlerklasse gleich 3 oder 4 ist und wenn kein Systemfehler gleichzeitig gemeldet wird.

Wenn Event = SIG_SYS_ERROR, muss zunächst die Fehlerklasse mit dem **MC_ReadSysError** festgestellt werden. Bei Fehlerklasse 4 befinden sich alle Achsen im Zustand **SysErrorStop**. Bei Fehlerklasse 3 müssen alle Achsen von der SPS aus schnellstmöglich angehalten werden.

Bei Event= SIG_AXE_ERROR oder Event= SIG_SYS_ERROR sind Fehler der Klasse 1 oder 2 als Hinweise oder Warnungen zu betrachten.

Vordefinierte Konstanten für Events in mc.lib:

```
VAR_GLOBAL CONSTANT
    SIG_AXE_ERROR      :DWORD := 1; (* Achsfehler *)
    SIG_SYS_ERROR      :DWORD := 2; (* Systemfehler *)
    SIG_MK_LOADED      :DWORD := 4; (* MK's geladen *)
END_VAR
```

Parameter

Name	Datentyp	Bedeutung
VAR_INPUT		
Enable	BOOL	TRUE = Ereignis anzeigen
VAR_OUTPUT		
Done	BOOL	TRUE = ein gültiger Wert für Ereignis steht am Ausgang Event an
Event	DWORD	Event= SIG_AXE_ERROR: mindestens eine Achse hat Fehler gemeldet Event= SIG_SYS_ERROR: Systemfehler ist aufgetreten Event= SIG_MK_LOADED: Maschinenkonstanten sind geladen

5 Parameterliste

Reservierte Parameter nach PLCopen-Standard

5 Parameterliste

5.1 Reservierte Parameter nach PLCopen-Standard

Nr.	B/E/V ¹⁾	Name	R/W ²⁾	TYP	Bedeutung
1	B	COMMANDED_POSITION	R	LREAL	Absolute Sollposition

Softwareendschalter:

Parameter SW_LIMIT_POS und SW_LIMIT_NEG werden mit in MC_DefineAxis konfigurierten Werten initialisiert.

2	E	SW_LIMIT_POS	R/W	LREAL	Software-Endschalter Plus in [U]
3	E	SW_LIMIT_NEG	R/W	LREAL	Software-Endschalter Minus in [U]
4	E	ENABLE_LIMIT_POS	R/W	BOOL	Software-Endschalter Plus: TRUE=aktiviert; FALSE=deaktiviert
5	E	ENABLE_LIMIT_NEG	R/W	BOOL	Software-Endschalter Minus: TRUE=aktiviert; FALSE=deaktiviert

Maximale programmierbare Verfahrensparameter:

MAX_VELOCITY_APPL, MAX_ACCELERATION_APPL, MAX_DECELERATION_APPL, MAX_JERK_APPL, MAX_TORQUE_APPL, MAX_TORQUE_RAMP_APPL.

Diese Parameter werden mit MC_DefineAxis konfigurierten Werten initialisiert (MAX_VELOCITY_CFG, MAX_ACCELERATION_CFG, MAX_DECELERATION_CFG, MAX_JERK_CFG, MAX_TORQUE_CFG, MAX_TORQUE_RAMP_CFG).

Sie können mit MC_WriteParameter geändert werden. Wenn die mit FB programmierten Werte diese Parameter überschreiten, werden entsprechende Fehlermeldungen angezeigt.

9	B	MAX_VELOCITY_APPL	R/W	LREAL	maximale programmierbare Geschwindigkeit in [U/s]
10	B	ACTUAL_VELOCITY	R	LREAL	Istgeschwindigkeit in [U/s]
11	B	COMMANDED_VELOCITY	R	LREAL	Absolute Sollgeschwindigkeit in [U/s]
13	E	MAX_ACCELERATION_APPL	R/W	LREAL	Maximale programmierbare Beschleunigung in [U/s ²]
15	E	MAX_DECELERATION_APPL	R/W	LREAL	Maximale programmierbare Verzögerung in [U/s ²]
16	E	MAX_JERK_APPL	R/W	LREAL	Maximaler programmierbarer Ruck in [U/s ³]
17	V	MAX_TORQUE_APPL	R/W	LREAL	Maximales programmierbares Drehmoment in [Nm] oder [N]
18	V	MAX_TORQUE_RAMP_APPL	R/W	LREAL	Maximale programmierbare Drehmomentrampe in [Nm] oder [N]
50	V	ACTUAL_POSITION_LAG	R	LREAL	Aktueller Schleppabstand in [U]
60	V	ENABLE_QUICK_STOP	R	BOOL	Quickstop TRUE=aktiviert; FALSE=deaktiviert
61	V	QUICK_STOP_DECLARATION	R	LREAL	Verzögerung für Quickstop in [U/s ²]
62	V	QUICK_STOP_JERK	R	LREAL	Ruck für Quickstop in [U/s ³]

Mit MC_DefineAxis konfigurierte Achsparameter:

MAX_VELOCITY_CFG, MAX_ACCELERATION_CFG, MAX_DECELERATION_CFG, MAX_JERK_CFG): Diese Parameter können mit MC_WriteParameter geändert werden.

100	V	AXIS_TYPE_CFG	R	LREAL	Achsenart
101	V	INCREMENTS_CFG	R	LREAL	Anzahl von Impulsen für Impulsbewertung
102	V	DISTANCE_CFG	R	LREAL	Äquivalenter Weg für Impulsbewertung (MC_DefineAxis) in [U]
103	V	HOME_OFFSET_CFG	R	LREAL	Grundoffset in [U]
104	V	SW_LIMIT_NEG_CFG	R	LREAL	Software-Endschalter Minus in [U]
105	V	SW_LIMIT_POS_CFG	R	LREAL	Software-Endschalter Plus in [U]
108	V	MAX_VELOCITY_CFG	R	LREAL	Maximale Geschwindigkeit in [U/s]
109	V	MAX_ACCELERATION_CFG	R	LREAL	Maximale Beschleunigung in [U/s ²]
110	V	MAX_DECELERATION_CFG	R	LREAL	Maximale Verzögerung in [U/s ²]
111	V	MAX_JERK_CFG	R	LREAL	maximaler Ruck in [U/s ³]
112	V	MAX_TORQUE_CFG	R	LREAL	Maximaler Drehmoment in [Nm] oder in [N] für lineare Motoren

Nr.	B/E/V ¹⁾	Name	R/W ²⁾	TYP	Bedeutung
113	V	MAX_TORQUE_RAMP_CFG	R	LREAL	Maximaler Anstieg vom Drehmoment in [Nm/s] oder in [N/s] für lineare Motoren
114	V	ANALOG_DRIVE_CHANNEL_CFG	R	LREAL	Nummer des LBM-ARI-Kannels
115	V	CAN_DRIVE_NODE_ID_CFG	R	LREAL	CAN-Antriebsadresse/Knotennummer
116	V	US_CFG	R	LREAL	Überwachung des Signals "Weggeber gestört"
117	V	UA0_CFG	R	LREAL	Überwachung des Signals 'Nullimpuls'
118	V	UA12_CFG	R	LREAL	Überwachung des Weggeberssignals
119	V	REGLER_MODE_CFG	R	LREAL	Die Art des Lagereglers
120	V	SSI_ENCODER_CFG	R	LREAL	Übertragungsparameter der SSI-Schnittstelle
121	V	AXIS_DIGITAL_INPUTS_CFG	R	LREAL	Ordnet logische Achsrechner-Eingänge
122	V	MAX_POSITION_LAG_CFG	R	LREAL	Grenze für Schleppfehlermeldung
123	V	MAX_POSITION_LAG_COUNTER_CFG	R	LREAL	Schleppfehlerzähler bis zur Meldung
124	V	DRIFT_COMPENSATION_CFG	R	LREAL	Offset für den Analogeingang des Antriebs-verstärkers
125	V	QUICKSTOP_CFG	R	LREAL	Bremszeit bei einem Achsfehler
126	V	KP_CFG	R	LREAL	Verstärkungsfaktor des P-Reglers
127	V	KF_CFG	R	LREAL	Bewertungsfaktor der Geschwindigkeits- vorsteuerung
128	V	KB_CFG	R	LREAL	Bewertungsfaktor der Beschleunigungs- vorsteuerung
129	V	TV_CFG	R	LREAL	Vorhaltezeit des PID-Reglers in [sec]
130	V	TN_CFG	R	LREAL	Nachstellzeit des PID-Reglers in [sec]
131	V	FILTER_TIME_CONST_CFG	R	LREAL	Filterzeitkonstante für Feininterpolator in [sec]
132	V	HOMING_TYPE_CFG	R	LREAL	Art und Weise der Referenzpunktfahrt
133	V	HOMING_MAX_VEL_TO_CAM_CFG	R	LREAL	Geschwindigkeit bei der Suche des Referenz-nockens
134	V	HOMING_MAX_ACC_TO_CAM_CFG	R	LREAL	Beschleunigung bei der Suche des Referenz-nockens
135	V	HOMING_MAX_VEL_TO_ZERO_CFG	R	LREAL	Geschwindigkeit bei der Suche des Nullimpul-ses
136	V	HOMING_MAX_ACC_TO_ZERO_CFG	R	LREAL	Beschleunigung bei der Suche des Nullimpul-ses
137	V	TORQUE_TECH_UNITS_DE-NOM_CFG	R	LREAL	Nenner für Umrechnungsfaktor des Drehmo-mentes
138	V	TORQUE_TECH_UNITS_NUM_CFG	R	LREAL	Zähler für Umrechnungsfaktor des Drehmo-mentes
139	V	VELOCITY_TECH_UNITS_DE-NOM_CFG	R	LREAL	Nenner für Umrechnungsfaktor der Drehzahl
140	V	VELOCITY_TECH_UNITS_NUM_CFG	R	LREAL	Zähler für Umrechnungsfaktor der Drehzahl

¹⁾ B = Base, E = Extended, V = Vendor (Bezeichnungen nach PLCopen)

²⁾ R = Parameter kann mit MC_ReadParameter bzw. MC_ReadBoolParameter gelesen werden.

W = Parameter kann mit MC_WriteParameter bzw. MC_WriteBoolParameter geschrieben werden.

5 Parameterliste

Anwenderspezifische Parameter

5.2 Anwenderspezifische Parameter

Nr.	Name	R/W ¹⁾	TYP	Bedeutung
Override				
1000	VEL_OVERRIDE	R	LREAL	Geschwindigkeitsoverride [Wert:0-1.2]
1001	ACC_OVERRIDE	R	LREAL	Beschleunigungs-/Verzögerungsoverride
1002	JERK_OVERRIDE	R	LREAL	Ruck-Override
Achsunabhängige Zustände vom Interpolator				
1101	INIT_DRIVE_NOT	R	BOOL	Interpolator-Zustand: Drivehandler nicht initialisiert
1102	INIT_DRIVE_RUN	R	BOOL	Interpolator-Zustand: Drivehandlerinitialisierung läuft
1103	INIT_DRIVE_ERROR	R	BOOL	Interpolator-Zustand: Fehler während der Drivehandlerinitialisierung
1104	INIT_DRIVE_OK	R	BOOL	Interpolator-Zustand: Drivehandlerinitialisierung beendet
Zustände von Achsen (siehe Abbildung 4. Zustandsdiagramm einer Achse beim Interpolator)				
1200	INT_STATE_STAND_STILL	R	BOOL	Interpolator-Zustand Standstill
1201	INT_STATE_HOMING	R	BOOL	Interpolator-Zustand Homing
1202	INT_STATE_ERROR_STOP	R	BOOL	Interpolator-Zustand ErrorStop
1203	INT_STATE_STOPPING	R	BOOL	Interpolator-Zustand Stopping
1204	INT_STATE_DISCRETE_MOTION	R	BOOL	Interpolator-Zustand DiscreteMotion
1205	INT_STATE_CONTINUOUS_MOTION	R	BOOL	Interpolator-Zustand ContinuousMotion
1206	INT_STATE_SYNCHRONIZED_MOTION	R	BOOL	Interpolator-Zustand SynchronizedMotion
1300	INT_STATE_STAND_STILL_IDLE	R	BOOL	Zustand Idle im Interpolator-Zustand Standstill : Interpolator ist bereit jede Motion-Aufgabe auszuführen
1301	INT_STATE_STAND_STILL_POWER_RUN	R	BOOL	Zustand PowerRun im Interpolator-Zustand Standstill : On-/Off der Reglerfreigabe läuft
Laufende Aufgabe beim Interpolator				
1400	CMD_NO_TASK	R	BOOL	Akt. Aufgabe: keine
1401	CMD_POWER	R	BOOL	Akt. Aufgabe: MC_Power
1402	CMD_STOP	R	BOOL	Akt. Aufgabe: MC_Stop
1403	CMD_RESET	R	BOOL	Akt. Aufgabe: MC_Reset
1404	CMD_HOME	R	BOOL	Akt. Aufgabe: MC_Home
1405	CMD_MOVE_ABSOLUTE	R	BOOL	Akt. Aufgabe: MC_MoveAbsolute
1406	CMD_MOVE_RELATIVE	R	BOOL	Akt. Aufgabe: MC_MoveRelative
1407	CMD_MOVE_ADDITIVE	R	BOOL	Akt. Aufgabe: MC_MoveAdditive
1408	CMD_POSITION_PROFILE	R	BOOL	Akt. Aufgabe: MC_PositionProfile
1409	CMD_MOVE_VELOCITY	R	BOOL	Akt. Aufgabe: MC_MoveVelocity
1410	CMD_VELOCITY_PROFILE	R	BOOL	Akt. Aufgabe: MC_VelocityProfile
1411	CMD_ACCELERATION_PROFILE	R	BOOL	Akt. Aufgabe: MC_AccelerationProfile
1412	CMD_CAM_IN	R	BOOL	Akt. Aufgabe: MC_CamIn
1413	CMD_CAM_OUT	R	BOOL	Akt. Aufgabe: MC_CamOut
1414	CMD_GEAR_IN	R	BOOL	Akt. Aufgabe: MC_GearIn
1415	CMD_GEAR_OUT	R	BOOL	Akt. Aufgabe: MC_GearOut
1416	CMD_MOVE_SUPERIMPOSED	R	BOOL	Aufgabe MC_MoveSuperimposed aktiv
1417	CMD_PHASING	R	BOOL	Aufgabe MC_Phasing aktiv
1418	CMD_GEAR_IN_POS	R	BOOL	Aufgabe MC_GearInPos
1419	CMD_TORQUE_CONTROL	R	BOOL	Aufgabe MC_TorqueControl
Referenzpunkt				
1500	HOME	R	BOOL	Referenzpunkt bekannt
1501	HOME_OFFSET	R	LREAL	Referenzpunktlage (Grundoffset) in [U]

Nr.	Name	R/W ¹⁾	TYP	Bedeutung
Reglerfreigabe				
1502	POWER_STAGE	R	BOOL	Achse betriebsbereit (Reglerkreis geschlossen)
Sollwertvorgabe aus dem Interpolator				
1600	INTERPOL_CMD_POS	R	LREAL	Aktuelle Sollposition ohne Überlagerung (SuperImposed) in [U]
1601	INTERPOL_CMD_POS_ABS	R	LREAL	Aktuelle an die Achse übergebene Sollposition aus allen aktiven Bewegungen in [U]
1602	INTERPOL_CMD_V_ABS	R	LREAL	Aktuelle Sollgeschwindigkeit aus allen aktiven Bewegungen in [U/s]
1603	INTERPOL_CMD_A_ABS	R	LREAL	Aktuelle Beschleunigung aus allen aktiven Bewegungen in [U/s ²]
1604	INTERPOL_CMD_R_ABS	R	LREAL	Aktueller Ruck aus allen aktiven Bewegungen in [U/s ³]
1605	INTERPOL_CMD_TORQUE	R	LREAL	Aktueller Solldrehmoment
Koordinatenverschiebung				
1610	OFFSET	R	LREAL	Offset des Koordinatensystems in Bezug auf Grundoffset
Istposition vom Antrieb und davon abgeleitete Ableitungen				
1700	ACTUAL_POSITION	R	LREAL	Aktuelle vom Antrieb gemeldete Position
1701	ACTUAL_VELOCITY	R	LREAL	1. Ableitung von ARlstPos in [U/s]
1702	ACTUAL_ACCELERATION	R	LREAL	2. Ableitung von ARlstPos in [U/s ²]
1703	ACTUAL_JERK	R	LREAL	3. Ableitung von ARlstPos in [U/s ³]
1704	ACTUAL_TORQUE	R	LREAL	Vom Antrieb gemeldeter Drehmoment
Bewegungszustand von Achse				
1800	MOVE_STATE_IDLE	R	BOOL	Achsbewegungszustand: Stillstand
1801	MOVE_STATE_ACCELERATION	R	BOOL	Achsbewegungszustand: Beschleunigung
1802	MOVE_STATE_DECELERATION	R	BOOL	Achsbewegungszustand: Verzögerung
1803	MOVE_STATE_CONST_VELOCITY	R	BOOL	Achsbewegungszustand: konstante Geschwindigkeit
Werte von Eingangsparameter von aktiven MC_MoveAbsolute, MC_MoveRelative, MC_MoveAdditive, MC_MoveVelocity				
1900	MOVE_DEST_POS	R	LREAL	Eingangsparameter: Position oder Distance in [U]
1901	MOVE_DEST_VELOCITY	R	LREAL	Eingangsparameter: Velocity in [U/s]
1902	MOVE_DEST_ACCELERATION	R	LREAL	Eingangsparameter: Acceleration in [U/s ²]
1903	MOVE_DEST_DECELERATION	R	LREAL	Eingangsparameter: Deceleration in [U/s ²]
1904	MOVE_DEST_JERK	R	LREAL	Eingangsparameter: Jerk in [U/s ³]
1906	MOVE_DIRECTION_POSITIV	R	BOOL	Eingangsparameter: Direction=MD_POSITIV_E
1907	MOVE_DIRECTION_NEGATIV	R	BOOL	Eingangsparameter: Direction=MD_NEGATIV_E
1908	MOVE_DIRECTION_SHORTEST_WAY	R	BOOL	Eingangsparameter: Direction=MD_SHORTEST_WAY_E
1909	MOVE_BM_ABORTING	R	BOOL	Eingangsparameter: BufferMode=BM_ABORTING_E
1910	MOVE_BM_BUFFERED	R	BOOL	Eingangsparameter: BufferMode=BM_BUFFERED_E
1911	MOVE_BM_BLENDING_LOW	R	BOOL	Eingangsparameter: BufferMode=BM_BLENDING_LOW_E
1912	MOVE_BM_BLENDING_PREVIOUS	R	BOOL	Eingangsparameter: BufferMode=BM_BLENDING_PREVIOUS_E
1913	MOVE_BM_BLENDING_NEXT	R	BOOL	Eingangsparameter: BufferMode=BM_BLENDING_NEXT_E
1914	MOVE_BM_HIGH	R	BOOL	Eingangsparameter: BufferMode=BM_BLENDING_HIGH_E
Sollwerte vom Interpolator aus aktiven MC_MoveAbsolute, MC_MoveRelative, MC_MoveAdditive, MC_MoveVelocity				
2000	MOVE_CMD_POS	R	LREAL	Sollposition in [U]
2001	MOVE_CMD_VELOCITY	R	LREAL	Sollgeschwindigkeit in [U/s]
2002	MOVE_CMD_ACCELERATION	R	LREAL	Sollbeschleunigung in [U/s ²]
2003	MOVE_CMD_DECELERATION	R	LREAL	Sollverzögerung in [U/s ²]
2004	MOVE_CMD_JERK	R	LREAL	Sollruck in [U/s ³]
2010	MOVE_PHASE	R	LREAL	Bewegungsphase: 1-7

Nr.	Name	R/W ¹⁾	TYP	Bedeutung
Werte von Eingangsparameter von aktivem MC_Stop				
2100	STOP_DEST_DECELERATION	R	LREAL	Eingangsparameter: Deceleration in [U/s ²]
2101	STOP_DEST_JERK	R	LREAL	Eingangsparameter: Jerk in [U/s ³]
Sollwerte vom Interpolator aus aktivem MC_Stop				
2102	STOP_CMD_VELOCITY	R	LREAL	Sollgeschwindigkeit in [U/s]
2103	STOP_CMD_DECELERATION	R	LREAL	Sollverzögerung in [U/s ²]
2104	STOP_CMD_JERK	R	LREAL	Sollruck in [U/s ³]
Werte von Eingangsparameter von aktivem MC_MoveSuperImposed				
2200	SUP_IMP_DEST_POS	R	LREAL	Eingangsparameter: Distance in [U]
2201	SUP_IMP_DEST_VELOCITY	R	LREAL	Eingangsparameter: VelocityDiff in [U/s]
2202	SUP_IMP_DEST_ACCELERATION	R	LREAL	Eingangsparameter: Acceleration in [U/s ²]
2203	SUP_IMP_DEST_DECELERATION	R	LREAL	Eingangsparameter: Deceleration in [U/s ²]
2204	SUP_IMP_DEST_JERK	R	LREAL	Eingangsparameter: Jerk in [U/s ³]
2205	SUP_BM_ABORTING	R	BOOL	Eingangsparameter: BufferMode=BM_ABORTING_E
2206	SUP_BM_BUFFERED	R	BOOL	Eingangsparameter: BufferMode=BM_BUFFERED_E
2207	SUP_BM_BLENDING_LOW	R	BOOL	Eingangsparameter: BufferMode=BM_BLENDING_LOW_E
2208	SUP_BM_BLENDING_PREVIOUS	R	BOOL	Eingangsparameter: BufferMode=BM_BLENDING_PREVIOUS_E
2209	SUP_BM_BLENDING_NEXT	R	BOOL	Eingangsparameter: BufferMode=BM_BLENDING_NEXT_E
2210	SUP_BM_HIGH	R	BOOL	Eingangsparameter: BufferMode=BM_BLENDING_HIGH_E
Sollwerte vom Interpolator aus aktivem MC_MoveSuperImposed				
2215	SUP_IMP_CMD_POS	R	LREAL	Sollposition in [U]
2216	SUP_IMP_CMD_VELOCITY	R	LREAL	Sollgeschwindigkeit in [U/s]
2217	SUP_IMP_CMD_ACCELERATION	R	LREAL	Sollbeschleunigung in [U/s ²]
2218	SUP_IMP_CMD_DECELERATION	R	LREAL	Sollverzögerung in [U/s ²]
2219	SUP_IMP_CMD_JERK	R	LREAL	Sollruck in [U/s ³]
2220	SUP_IMP_STATE_IDLE	R	BOOL	Achsbewegungszustand: Stillstand
2221	SUP_IMP_STATE_ACCELERATION	R	BOOL	Achsbewegungszustand: Beschleunigung
2222	SUP_IMP_STATE_DECELERATION	R	BOOL	Achsbewegungszustand: Verzögerung
2223	SUP_IMP_STATE_CONST_VELOCITY	R	BOOL	Achsbewegungszustand: konstante Geschwindigkeit
2230	SUP_IMP_PHASE	R	LREAL	Bewegungsphase: 1-7
Werte von Eingangsparameter von aktivem MC_Phasing				
2300	PHASING_DEST_POS	R	LREAL	Eingangsparameter: PhaseShift in [U]
2301	PHASING_DEST_VELOCITY	R	LREAL	Eingangsparameter: Velocity in [U/s]
2302	PHASING_DEST_ACCELERATION	R	LREAL	Eingangsparameter: Acceleration in [U/s ²]
2303	PHASING_DEST_DECELERATION	R	LREAL	Eingangsparameter: Deceleration in [U/s ²]
2304	PHASING_DEST_JERK	R	LREAL	Eingangsparameter: Jerk in [U/s ³]
2305	PHASING_BM_ABORTING	R	BOOL	Eingangsparameter: BufferMode=BM_ABORTING_E
2306	PHASING_BM_BUFFERED	R	BOOL	Eingangsparameter: BufferMode=BM_BUFFERED_E
2307	PHASING_BM_BLENDING_LOW	R	BOOL	Eingangsparameter: BufferMode=BM_BLENDING_LOW_E
2308	PHASING_BM_BLENDING_PREVIOUS	R	BOOL	Eingangsparameter: BufferMode=BM_BLENDING_PREVIOUS_E
2309	PHASING_BM_BLENDING_NEXT	R	BOOL	Eingangsparameter: BufferMode=BM_BLENDING_NEXT_E
2310	PHASING_BM_HIGH	R	BOOL	Eingangsparameter: BufferMode=BM_BLENDING_HIGH_E
Sollwerte vom Interpolator aus aktivem MC_Phasing				
2315	PHASING_CMD_POS	R	LREAL	Sollposition in [U]
2316	PHASING_CMD_VELOCITY	R	LREAL	Sollgeschwindigkeit in [U/s]
2317	PHASING_CMD_ACCELERATION	R	LREAL	Sollbeschleunigung in [U/s ²]

Nr.	Name	R/W ¹⁾	TYP	Bedeutung
2318	PHASING_CMD_DECELERATION	R	LREAL	Sollverzögerung in [U/s ²]
2319	PHASING_CMD_JERK	R	LREAL	Sollruck in [U/s ³]
2320	PHASING_STATE_IDLE	R	BOOL	Achsbewegungszustand: Stillstand
2321	PHASING_STATE_ACCELERATION	R	BOOL	Achsbewegungszustand: Beschleunigung
2322	PHASING_STATE_DECELERATION	R	BOOL	Achsbewegungszustand: Verzögerung
2323	PHASING_STATE_CONST_VELOCITY	R	BOOL	Achsbewegungszustand: konstante Geschwindigkeit
2330	PHASING_PHASE	R	LREAL	Bewegungsphase: 1-7
Sollwerte vom Interpolator aus aktivem MC_PositonProfile				
2400	POS_PROFILE_CMD_POS	R	LREAL	Sollposition in [U]
2401	POS_PROFILE_CMD_VELOCITY	R	LREAL	Sollgeschwindigkeit in [U/s]
2402	POS_PROFILE_CMD_ACCELERATION	R	LREAL	Sollbeschleunigung in [U/s ²]
2403	POS_PROFILE_CMD_DECELERATION	R	LREAL	Sollverzögerung in [U/s ²]
2404	POS_PROFILE_CMD_JERK	R	LREAL	Sollruck in [U/s ³]
Sollwerte vom Interpolator aus aktivem MC_VelocityProfile				
2500	VELOCITY_PROFILE_CMD_POSITION	R	LREAL	Sollposition in [U]
2501	VELOCITY_PROFILE_CMD_VELOCITY	R	LREAL	Sollgeschwindigkeit in [U/s]
2502	VELOCITY_PROFILE_CMD_ACCELERATION	R	LREAL	Sollbeschleunigung in [U/s ²]
2503	VELOCITY_PROFILE_CMD_DECELERATION	R	LREAL	Sollverzögerung in [U/s ²]
2504	VELOCITY_PROFILE_CMD_JERK	R	LREAL	Sollruck in [U/s ³]
Sollwerte vom Interpolator aus aktivem MC_AccelerationProfile				
2600	ACCPROFILE_CMD_POSITION	R	LREAL	Sollposition in [U]
2601	ACCPROFILE_CMD_VELOCITY	R	LREAL	Sollgeschwindigkeit in [U/s]
2602	ACCPROFILE_CMD_ACCELERATION	R	LREAL	Sollbeschleunigung in [U/s ²]
2603	ACCPROFILE_CMD_DECELERATION	R	LREAL	Sollverzögerung in [U/s ²]
2604	ACCPROFILE_CMD_JERK	R	LREAL	Sollruck in [U/s ³]
Sollwerte vom Interpolator aus aktivem MC_CamIn				
2700	CAM_CMD_POSITION	R	LREAL	Sollposition in [U]
2701	CAM_CMD_VELOCITY	R	LREAL	Sollgeschwindigkeit in [U/s]
2702	CAM_CMD_ACCELERATION	R	LREAL	Sollbeschleunigung in [U/s ²]
2703	CAM_CMD_DECELERATION	R	LREAL	Sollverzögerung in [U/s ²]
2704	CAM_CMD_JERK	R	LREAL	Sollruck in [U/s ³]
2710	CAM_MASTER_TAB_POS	R	LREAL	Masterposition für Interpolation der Slaveposition aus CAM-Tabelle in [U]
Sollwerte vom Interpolator aus aktivem MC_GearIn				
2800	GEAR_CMD_POSITION	R	LREAL	Sollposition in [U]
2801	GEAR_CMD_VELOCITY	R	LREAL	Sollgeschwindigkeit in [U/s]
2802	GEAR_CMD_ACCELERATION	R	LREAL	Sollbeschleunigung in [U/s ²]
2803	GEAR_CMD_DECELERATION	R	LREAL	Sollverzögerung in [U/s ²]
2804	GEAR_CMD_JERK	R	LREAL	Sollruck in [U/s ³]
Sollwerte vom Interpolator aus aktivem MC_GearInPos				
2820	GEAR_IN_POS_CMD_POSITION	R	LREAL	Sollposition in [U]
2821	GEAR_IN_POS_CMD_VELOCITY	R	LREAL	Sollgeschwindigkeit in [U/s]
2822	GEAR_IN_POS_CMD_ACCELERATION	R	LREAL	Sollbeschleunigung in [U/s ²]
2823	GEAR_IN_POS_CMD_DECELERATION	R	LREAL	Sollverzögerung in [U/s ²]
2824	GEAR_IN_POS_CMD_JERK	R	LREAL	Sollruck in [U/s ³]
Mit MC_DefineAxis konfigurierten und auf Interpolationstakt umgerechneten Achsparameter				
2900	IMPULSE_PRO_EINHEIT_RUN	R	LREAL	Umrechnungsfaktor: Anzahl von Inkrementen pro Einheit [U]

Nr.	Name	R/W ¹⁾	TYP	Bedeutung
2901	HOME_OFFSET_CFG_RUN	R	LREAL	Grundoffset in Inkrementen
2903	MAX_VELOCITY_CFG_RUN	R	LREAL	maximale Geschwindigkeit in [U/GIT]
2904	MAX_ACCELERATION_CFG_RUN	R	LREAL	maximale Beschleunigung in [U/GIT ²]
2905	MAX_DECELERATION_CFG_RUN	R	LREAL	maximale Verzögerung in [U/GIT ²]
2906	MAX_JERK_CFG_RUN	R	LREAL	maximaler Ruck in [U/GIT ³]
2907	TORQUE_IN_TECH_UNIT_RUN	R	LREAL	Umrechnungsfaktor für Drehmoment: Buseinheit in technische Einheit
2908	VELOCITY_IN_TECH_UNIT_RUN	R	LREAL	Umrechnungsfaktor für Geschwindigkeit (Drehzahl)t: Buseinheit in technische Einheit
Phasenverschiebung				
3000-3127	PHASING_PHASE_SHIFT	R	LREAL	Phasenverschiebung der Master-Achse 0 ... 127 in [U]

- ¹⁾ R = Parameter kann mit MC_ReadParameter bzw. MC_ReadBoolParameter gelesen werden.
W = Parameter kann mit MC_WriteParameter bzw. MC_WriteBoolParameter geschrieben werden.

6 SPS-Programmierung

6.1 ETC-SPS-Programmierung mit CoDeSys

CoDeSys ist eine komplette Entwicklungsumgebung zum Erstellen und Testen von SPS-Programmen für die ETC.

CoDeSys bietet Möglichkeiten zum Debuggen von Programmen, ähnlich den heutigen Hochsprachenentwicklungssystemen (Setzen von Haltepunkten, Monitoring von Variablen, Aufzeichnung eines Traces (Oszilloskopfunktion) usw.).

Eine Einführung in die SPS-Programmierung nach IEC 61131-3 mit CoDeSys enthält das CoDeSys-Handbuch (siehe "Docu"-Verzeichnis im CoDeSys-Installationspfad).

Dieses Benutzerhandbuch ist als Ergänzung zum CoDeSys-Handbuch zu verstehen. Es enthält Beschreibungen zu Funktionen bzw. Anpassungen im Zusammenhang mit CoDeSys und der ETC.

6.2 CoDeSys installieren**6.2.1 Systemvoraussetzungen für CoDeSys V2.xx**

- ▶ Pentium-Prozessor (Pentium II, ab 350 MHz empfohlen)
- ▶ 32 MB RAM (64 MB empfohlen)
- ▶ Windows 2000 oder XP
- ▶ MS Internet Explorer ab Version 4.0

6.2.2 Software installieren

1. Legen Sie die CoDeSys-Setup-CD in Ihr CD-ROM-Laufwerk.
Wenn das Autostart-Feature des CD-ROM-Laufwerks in Windows aktiviert ist, startet das Setup automatisch.
2. Sollte das Setup nicht automatisch starten:
 - Wählen Sie im Startmenü den Menüpunkt "Ausführen".
 - Geben Sie in der Befehlszeile den Laufwerksbuchstaben Ihres CD-ROM-Laufwerks gefolgt von "setup.exe" ein (z. B. "d:\setup.exe") und bestätigen Sie mit "OK".
3. Folgen Sie den Anweisungen des Setup-Programms.
CoDeSys wird im Verzeichnis "c:\programme\lenze\CoDeSys2 V2.3" installiert.

**Stop!**

Die Datei "codesys.ini" im Installationsverzeichnis kann nach der Installation von CD schreibgeschützt sein. Der Schreibschutz muss aufgehoben werden, da sonst die CoDeSys-Einstellungen nicht gespeichert werden.

6.3 ETC und PC verbinden

Die Verbindung zur ETC kann über eine der folgenden Schnittstellen hergestellt werden:

- Serielle Schnittstelle (V.24)
- Ethernet-Schnittstelle

6.3.1 V.24-Schnittstelle

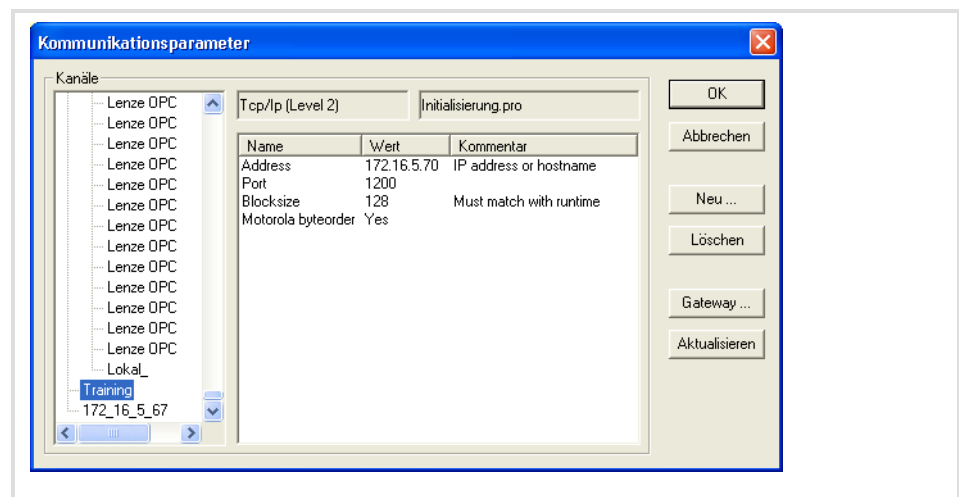
Für die Kommunikation über die serielle Schnittstelle muss der RS232- oder CAN2-Anschluss der ETC über ein Nullmodemkabel mit einem freien COM-Port des PCs verbunden werden. In den Kommunikationsparametern von CoDeSys muss der Treiber "3S Serial RS232 driver" ausgewählt werden. In den Einstellungen zu diesem Treiber müssen folgende Werte für die entsprechende Schnittstelle der Steuerung ausgewählt werden:

Anschluss ETC	RS232	CAN2
Baudrate	115200	38400
Parity	No	No
Stop bits	1	1
Motorola byteorder	Yes	Yes

6.3.2 Ethernet-Schnittstelle

Für die Kommunikation über Ethernet muss der ETC-Ethernet-Anschluss über ein Netzkabel mit dem PC verbunden werden.

In den Kommunikationsparametern von CoDeSys muss der Treiber "3S Tcp/Ip level2 driver" ausgewählt werden. In den Einstellungen zum Treiber müssen folgende Werte eingetragen werden:



6.4 Projektierung

6.4.1 Zielsystem einstellen

Beim Anlegen eines neuen Projekts in CoDeSys öffnet sich automatisch ein Dialog, der nach der verwendeten Hardware (=Zielsystem) fragt. Nach dem Anlegen ist der Konfigurationsdialog über den Menüpunkt "Zielsystemeinstellungen" im Registerblatt "Ressourcen" aufrufbar.



Hinweis!

Das gewählte Zielsystem wird beim Einloggen in die Steuerung überprüft, so dass ein SPS-Programm durch CoDeSys nicht zu einem falschen Zielsystem übertragen werden kann.

Bei einem Bootprojekt (📖 231) ist darauf zu achten, dass das ausgewählte Zielsystem zu der verwendeten Firmware der Steuerung passt, da beim Laden eines Bootprojekts, zum Beispiel durch ein ETC-MMI, dieses nicht überprüft wird.

► Steuerungsvariante Motion: Zielsystemvorauswahl = ETCxM

Die Parameter unter "Zielsystemeinstellungen" werden anhand des gewählten Zielsystems voreingestellt und brauchen in der Regel nicht verändert werden.

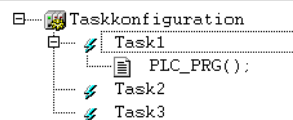
Eine Beschreibung der Zielsystemeinstellungen finden Sie im CoDeSys-Handbuch.

6.4.2 SPS-Tasks der ETCxM konfigurieren

SPS-Tasks einbinden

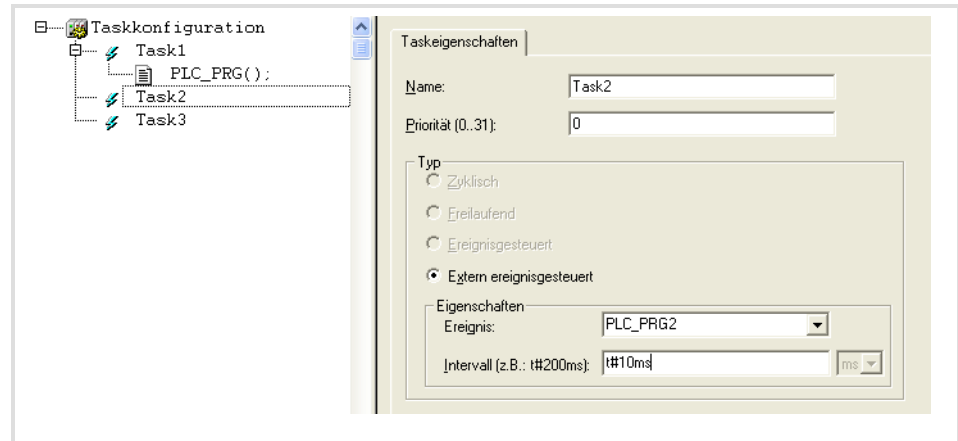
Mit der ETCxM ist es möglich, vier voneinander unabhängige SPS-Tasks zu nutzen. Jede Task kann mit einem Aufruf eines IEC-Programmbausteins (PRG) verknüpft sein. Die mit den Tasks verknüpften Programmbausteine werden von dem Multitaskingbetriebssystem der ETCxM quasiparallel bearbeitet.

Die Abarbeitungsreihenfolge ergibt sich aus der Intervallzeit (zeitgesteuert) und der Priorität einer Task. Eine Task läuft also solange, bis ein SPS-Zyklus abgearbeitet wurde oder eine andere Task mit höherer Priorität gestartet wird. Damit ein SPS-Programm durch die ETCxM abgearbeitet wird, muss in der Taskkonfiguration von CoDeSys mindestens eine Task angelegt sein und mit einem IEC-Programmbaustein verknüpft werden.



Eigenschaften einer Task

Die Eigenschaften der einzelnen Task konfigurieren Sie in der "Taskkonfiguration" von CoDeSys.



Folgende Punkte sind dabei zu beachten:

- ▶ Je größer der Wert bei Priorität, desto größer ist die Task-Priorität.
- ▶ Eine Task mit Intervall von "t#0s" darf höchstens eine Priorität von 10 erhalten, sonst setzt die Steuerung die Priorität aller Tasks auf 0 und meldet einen Fehler (Modul 13, Fehlernummer 202).
- ▶ Zudem muss eine Task mit Intervall von "t#0s" die niedrigste Priorität von allen Tasks haben, sonst setzt die Steuerung die Priorität aller Tasks auf 0 und meldet einen Fehler (Modul 13, Fehlernummer 201). Die Tasks mit niedrigerer Priorität als die vorgenannte Tasks würden sonst nicht zur Ausführung kommen.
- ▶ Gibt es mehrere Tasks mit Intervall "t#0s", müssen diese Tasks alle die gleiche Priorität haben.
- ▶ Die Intervallzeit ist zwischen "t#0ms" (Dauerläufer) und "t#1000ms" zu wählen.
- ▶ Bei dem Eintrag "Ereignis" in den Taskeigenschaften von CoDeSys ist jede Task mit einer Ereignisvariablen zu verknüpfen.
- ▶ Das Betriebssystem der ETCxM ordnet damit den 4 Tasks den entsprechenden Aufruf des mit der Task verknüpften CoDeSys-Programmbausteins zu. Jeder Task ist eine Ereignisvariable zugeordnet.

Tasknummer	Ereignisvariable
1	PLC_PRG1
2	PLC_PRG2
3	PLC_PRG3
4	PLC_PRG4

**Stop!**

Beim Zugriff auf Daten bzw. beim Aufruf von Bausteinen, die in mehreren Tasks verwendet werden, muss beachtet werden, dass keine Synchronisation zwischen den Tasks besteht.

Der SPS-Programmierer muss dafür Sorge tragen, dass sich dadurch keine Probleme ergeben

Prozessabbild

Bei der Zuordnung der CoDeSys-Programmbausteine zu den SPS-Tasks ist zu beachten, dass jeder Task nur ein Teil des Prozessabbildes zugeordnet ist. Das Prozessabbild ist eine Kopie der aktuellen Zustände der Ein-/Ausgangsbaugruppen (Hardware), auf die das SPS-Programm zugreift.

Vor jedem SPS-Zyklus (kompletter Programmdurchlauf) werden die aktuellen Zustände der Eingangsmodule in das Prozessabbild kopiert und nach jedem SPS-Zyklus werden die aktuellen Zustände der Ausgänge im Prozessabbild zu den Ausgangsmodulen übertragen.

Da jeder Task ein Teil des Prozessabbildes zugeordnet ist, wird vor und nach einem Durchlauf des Programmbausteins, der einer Task zugeordnet ist, nur der entsprechende Teil des Prozessabbildes aktualisiert.

6.4.3 E/A-Module konfigurieren

CoDeSys beinhaltet einen CAN-Bus-Konfigurator. Der Konfigurator stützt sich auf sogenannte EDS-Dateien (EDS = Electronic Data Sheet), die im Unterverzeichnis "ETC" des Targets-Verzeichnisses abgelegt sein müssen.

Eine allgemeine Beschreibung zur Konfiguration eines CAN-Moduls finden Sie in der CoDeSys-Dokumentation.

6.4.3.1 CAN-Master konfigurieren (globale CAN-Einstellungen)**Basisparameter**

Adresse automatisch: Der Schalter sollte ausgewählt werden, da CoDeSys sonst die Adressen automatisch vergibt.

CAN-Parameter

Baudrate: Die Baudrate für den CAN-Bus muss entsprechend den Einstellungen der angeschlossenen CAN-Module ausgewählt werden.



Hinweis!

Die Eingabe der Baudrate ist nur für die ETCxM aktiv.
Für die ETCxC hat dieser Parameter keine Funktion. Es gilt nur die Einstellung in den Maschinenkonstanten.

Com. Cycle Period: Wird von der Steuerung nicht ausgewertet.

Sync. Window Length: Wird von der Steuerung nicht ausgewertet.

Sync COB-ID: Wird von der Steuerung nicht ausgewertet.

Node-ID: Geben Sie hier eine "0" ein.

Automatisch Starten: Wird von der Steuerung nicht ausgewertet.

DSP301 ...: Wird ausgewertet und schaltet beim modularen IP20-I-/O-System die CAN-Modulauswahl aktiv.

Heartbeat Master: Wird von der Steuerung ausgewertet.

6.4.3.2 CAN-Slave konfigurieren

Basisparameter

Basisparameter	CAN Parameter	PDO-Mapping Empfangen	PDO-Mapping Senden	Service Data Objects
Modul-Id: 8 Knotennummer: 0 Eingabeadresse: %IW64 Ausgabeadresse: %QW64 Diagnoseadresse: %MB12				

Modul-ID: Idennummer des Moduls.

Knotennummer: Die Node-ID wird unter der Registerkarte CAN-Parameter eingegeben.

Eingabe-/Ausgabeadresse: Die Basisadressen sind als Wortadressen im Bereich von %QW64 / %IW64 bis %QW127 / %IW127 zu vergeben.

Weitere Informationen finden Sie im Kapitel "Adressierung" (📖 224).

Diagnoseadresse: Dieses Feld wird von der Steuerung nicht ausgewertet.

CAN-Parameter

The screenshot shows the 'CAN Parameter' configuration window. The 'Allgemein' section has 'Node ID' set to 80. The 'Nodeguard' section has 'Nodeguarding' checked, 'Guard COB-ID' set to 1792+NodeID, 'Guard Time (ms)' set to 500, and 'Life Time Factor' set to 3. The 'Heartbeat Einstellungen' section has 'Heartbeat Erzeugung aktivieren' checked, 'Heartbeat Producer Time' set to 0 ms, and 'Heartbeat Verbrauch aktivieren' unchecked. The 'Emergency Telegram' section has 'Emergency' checked and 'COB-ID' set to 128+NodeID. An 'Info...' button is present next to the Nodeguard section.

Allgemein: Bei Node-ID ist die an dem Modul eingestellte CAN-ID (Knotennummer) einzutragen.

Nodeguarding: Das Nodeguarding dient zum Erkennen, ob ein CANopen-Modul am Bus angeschlossen ist. Dazu sendet die Steuerung im Intervall von "Guard time" eine Nachricht zum Modul und erwartet eine Antwort.

Empfängt die Steuerung nach "Life-Time Factor"-Anforderungen keine Antwort, wird eine Fehlermeldung erzeugt.

Ist keine "Guard Time" und kein "Life Time Factor" angegeben, wird das CAN-Modul nicht überwacht; das heißt, ein Ausfall wird nicht bemerkt.

Heartbeat Einstellungen: Wird von der Steuerung nicht ausgewertet.

Emergency: Ist dieser Eintrag abgewählt, ignoriert die Steuerung "Emergency-Telegramme" der Steuerung.

PDO-Mapping

Per PDO (Process Data Object) werden bei CANopen die Prozessdaten übertragen; das heißt die Zustände der digitalen und analogen Ein-/Ausgänge.

Jeder PDO ist eine eindeutige COB-ID (385-1407) zugewiesen, wobei die COB-IDs bei den CANopen-IO-Modulen mit der Annahme von maximal 127 Modulen folgendermaßen vergeben sind:

1. Tx-PDO: $384 + \text{NodeID}$
1. Rx-PDO: $512 + \text{NodeID}$
2. Tx-PDO: $640 + \text{NodeID}$
2. Rx-PDO: $768 + \text{NodeID}$
3. Tx-PDO: $896 + \text{NodeID}$
3. Rx-PDO: $1024 + \text{NodeID}$
4. Tx-PDO: $1152 + \text{NodeID}$
4. Rx-PDO: $1280 + \text{NodeID}$

(Tx = Transmit, Rx = Receive ist aus Sicht des CANopen-Moduls zu sehen)

Es können also ohne Einschränkung der maximalen Anzahl von Modulen 4 Receive- und Transmit-PDOs pro Modul vergeben werden. Sollen abweichend vom Standard mehr PDOs pro Modul definiert werden, ist darauf zu achten, dass die COB-ID's nicht doppelt vergeben werden.

Die erste Tx/Rx-PDO enthält die digitalen Daten und die zweite Tx/Rx-PDO die analogen Daten. Für die zweite PDO sollte bei "Transmission Type" immer "asynchron-nur RTR" eingetragen sein, da dies die Standardeinstellung der meisten Module für die Übertragung von analogen Werten ist.

Service Data Objects

Diese Einträge werden von der Steuerung nicht ausgewertet.

6.4.3.3 Überprüfung der Anzahl von Eingängen und Ausgängen

Die Steuerung führt eine implizite Überprüfung der Anzahl der digitalen/ analogen Ein- und Ausgänge durch.

Dabei wird von der Annahme ausgegangen, dass die digitalen Daten in der ersten Tx/Rx-PDO und die analogen Werte in der zweiten Tx/Rx-PDO liegen. Diese Annahme und die daraus folgende Überprüfung wird nur durchgeführt, wenn in der EDS-Datei zu dem CANopen-Modul für den Objekteintrag 1000_{hex} (Device Profil Number) in den zusätzlichen Informationen (höchstwertigen 16-bit) die IO-Unterstützung für digitale und analoge Ein-/Ausgänge markiert ist (s. CiA DS401: Device Profil for Generic I/O Modules).

**Hinweis!**

Die Steuerung erzeugt eine Fehlermeldung, wenn die Anzahl der konfigurierten digitalen/analogen Ein-/Ausgänge einer PDO die tatsächliche Anzahl übersteigt.

Auszug aus einer EDS-Datei

```
[1000]
ParameterName=DeviceType
ObjectType=0x7
DataType=0x7
AccessType=ro
PDOMapping=0
DefaultValue=0x30191
```

**Beschreibung Objektindex
1000hex**

Zusätzliche Information				Allgemeine Information	
Spez. Funktionalität		I/O Funktionalität		Device Profile Number	
31	24	23	16	15	0

I/O-Funktionalität

Bit 16: Digitale Eingänge
 Bit 17: Digitale Ausgänge
 Bit 18: Analoge Eingänge
 Bit 19: Analoge Ausgänge

Überprüfung ausschalten

Um die Überprüfung der Anzahl der IOs auszuschalten, muss DefaultValue = 0x191 gesetzt werden. Falls ein Modul auch mit der 1. Tx/Rx-PDO analoge Daten überträgt, muss DefaultValue = 0xC0191 gesetzt werden, also die Unterstützung für digitale E/A's ausgeschaltet werden.

6.4.3.4 Suchen von konfigurierten CANopen-Modulen

Die Steuerung sucht nach allen in der Steuerungskonfiguration konfigurierten Modulen bevor die eigentliche Kommunikation (per PDO) mit dem Modul gestartet wird.

Das Suchen der Module erfolgt, per NodeGuarding, falls konfiguriert, ansonsten per SDO-Transfer (Auslesen von Objektindex 1000_{hex}).

Suchen ausschalten

Für CANopen-Module, die kein NodeGuarding oder SDO-Transfer unterstützen, kann das Suchen der Module ausgeschaltet werden. Dazu muss in der EDS-Datei zu dem CANopen-Modul für den Objekteintrag 1000_{hex} (Device Profil Number) eine 0 eingetragen werden.

6.4.4 Adressierung

Die ETC arbeitet nach dem Big-Endian-Datenmodell, dabei liegen alle Datentypen mit dem hochwertigsten Byte beginnend, also dem "dicken Ende", zuerst im Speicher. Dies muss vor allem bei der Kommunikation mit einem PC-HMI (z. B. ETC-MMI) über den DB2 (ETCxC) berücksichtigt werden, da PC's nach dem Little-Endian-Datenmodell arbeiten. Die Steuerung stellt hierfür einige Funktionen zur Verfügung, um dies zu erleichtern (📖 261).

Die direkte Darstellung einer einzelnen Speicherzelle erfolgt mittels spezieller Zeichenreihen. Diese entsteht aus dem Prozentzeichen "%", einem Bereichspräfix, einem Präfix für die Datenbreite und zwei oder drei natürlichen Zahlen, die durch Punkte voneinander getrennt sind.

Bereichspräfix	Bedeutung
I	Eingang
Q	Ausgang
M	Datenbaustein (DBO ... DB15) der ETC

Datenbreiten-Präfix	Bedeutung
X	1 Bit
B	Byte (8 Bits)
W	Word (16 Bits)
D	Doppelwort (32 Bits)

6.4.4.1 E/A-Module adressieren

Die E/A-Adressierung ist grundsätzlich wortweise organisiert. Die Adressen der I/O's ergeben sich aus der internen Organisation des Prozessabbildes der Steuerung.

Zugriffsart	Syntax	Bemerkung
Bitweise	%IX x.y / %QX x.y	x: Nummer des Datenwortes (0 ... 127) y: Bit in dem Wort (0 ... 15)
Byteweise	%IB x / %QB x	x: Nummer des Bytes (0 ... 254)
Wortweise	%IW x / %QW x	x: Nummer des Datenwortes
Doppelwortweise	%ID x / %QD x	x: Nummer des Doppelwortes

**Stop!**

Wortweiser Zugriff: Wegen der internen Datenorganisation ist bei einem byteweisen Zugriff darauf zu achten, dass im Speicher zuerst das High-Byte und dann das Low-Byte steht (siehe Beispiel).

Doppelwortweiser Zugriff: Wegen der internen Datenorganisation macht ein doppelwortweiser Zugriff normalerweise keinen Sinn, da High- und Low-Word vertauscht wären (siehe Beispiel).

6.4.4.2 Datenbausteine adressieren

Die Datenbausteine (DB0 - DB15) werden über den Bereichspräfix "M" angesprochen.

Zugriffsart	Syntax	Bemerkung
Bitweise	%MX x.y.z	x: Nummer des Datenbausteins y: Wort in dem Datenbaustein (0 ... 255) z: Bit in dem Wort (0 ... 15)
Byteweise	%MB x.y.z	x: Nummer des Datenbausteins y: Wort in dem Datenbaustein (0 ... 255) z: linkes (höherwertiges) oder rechtes (niederwertiges) Byte des Wortes (links = 1, rechts = 0)
Wortweise	%MW x.y	x: Nummer des Datenbausteins y: Wort in dem Datenbaustein (0 ... 255)
Doppelwortweise	%MD x.y	x: Nummer des Datenbausteins y: Doppelwort in dem Datenbaustein (0 ... 127)



Stop!

Doppelwortweiser Zugriff: Wegen der internen Daten-Organisation macht ein doppelwortweiser Zugriff keinen Sinn, da High- und Low-Word vertauscht wären.

6.4.5 Remanente Variablen

Mit dem Schlüsselwort "RETAIN" können bei der Variablendeklaration in CoDeSys remanente Daten festgelegt werden (siehe CoDeSys-Dokumentation). Es stehen insgesamt 12 kB für remanente Daten zur Verfügung.

Die ETC unterstützt nur die Variablen-Typen VAR und VAR RETAIN.

6.4.6 Objektverzeichnis (Parametermanager)

In CoDeSys ist es möglich, ein sogenanntes CANopen-Objektverzeichnis anzulegen, das durch ein CANopen-Gerät (z. B. CANopen-Bedienteil) gelesen und beschrieben werden kann.

Um die Übertragung eines Objektverzeichnisses über den CAN-Bus zu aktivieren, muss der Steuerung über die Funktion "InitServerSdo()" (siehe Bibliothek "ServerSdo.lib") das Objektverzeichnis bekannt gegeben werden.

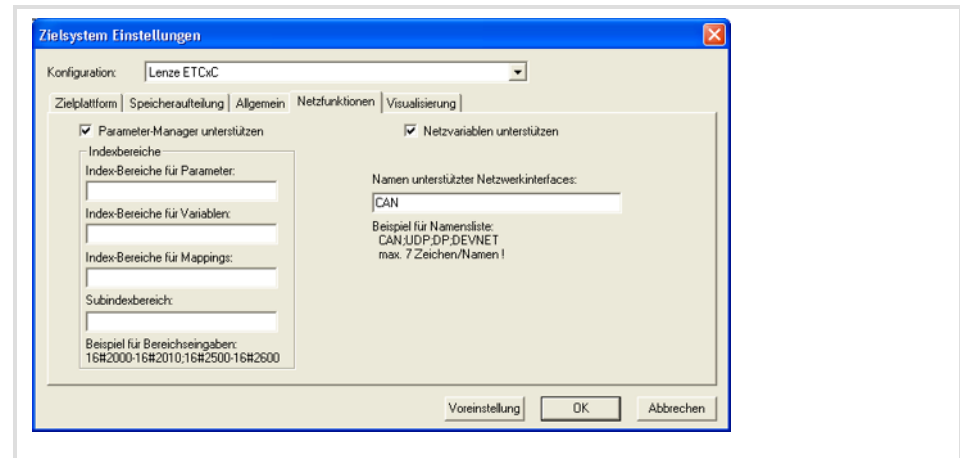


Hinweis!

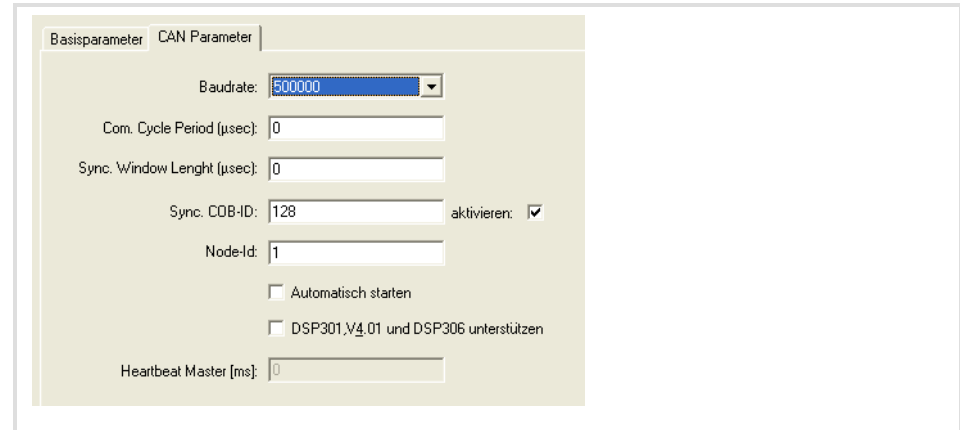
Beim Einfügen der Bibliothek "ServerSdo.lib" in der Bibliotheksverwaltung von CoDeSys werden automatisch weitere Bibliotheken (SysExtensions.lib, SysLibCallback.lib, 3S_CANopenNetVar.lib, 3S_CANopenManager.lib und 3S_CanDrv.lib) eingefügt.

Ein Objektverzeichnis kann über den Objektverzeichnis-Editor von CoDeSys erstellt werden. Um den Objektverzeichnis-Editor benutzen zu können, muss im CoDeSys unter **ZielsystemEinstellungen** -> **Netzfunktionen** (siehe folgendes Bild) die Unterstützung von Objektverzeichnissen und Netzvariablen aktiviert sein.

Definieren Sie Index- bzw. Subindex-Bereiche für SDO-Parameter und Variablen und geben Sie "CAN" für den Namen unterstützter Netzwerkinterfaces ein.



Die CANopen-Adresse der Steuerung ist unter **Steuerungskonfiguration** -> **CAN-Master** -> **CAN Parameter** -> **Node-Id** festzulegen.

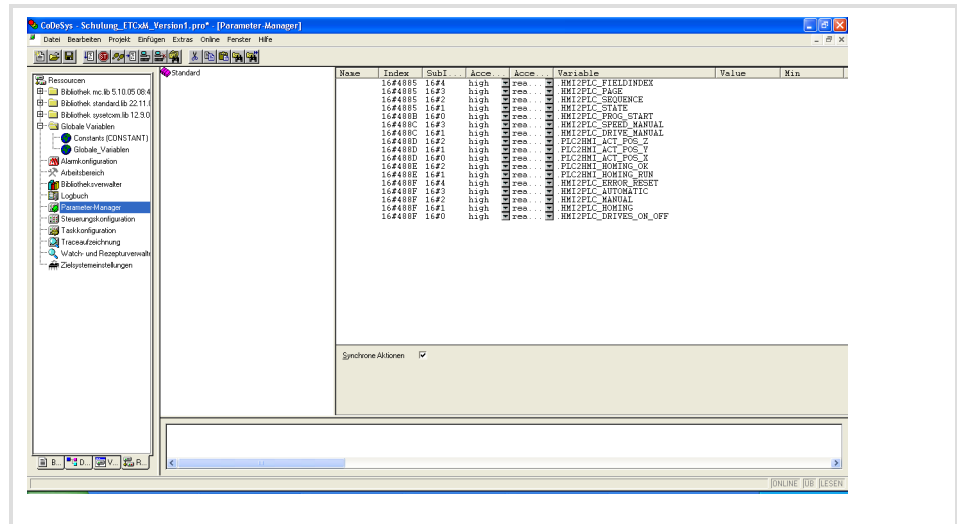


Für die Festlegung der Node-Id ist Folgendes zu beachten:

Wenn die Node-Id im Bereich 1 ... 63 liegt, werden zwei Server-SDO-Kanäle mit den Adressen "Node-Id" und "Node-Id + 64" erzeugt, d. h. die Steuerung erhält 2 CANopen-Adressen. Da die Verwendung von 2 SDO-Kanälen im Allgemeinen nicht notwendig ist, sollte eine Node-Id im Bereich 64 ... 127 vergeben werden, da dann nur ein SDO-Kanal angelegt wird.

Im Objektverzeichnis-Editor von CoDeSys dürfen nur Einträge in der Registerkarte "Variable" eingetragen werden.

Legen Sie ein neues Objektverzeichnis an. Hier definieren Sie die Variablen, die mit anderen Steuerungen ausgetauscht werden sollen.



Die Registerkarte Variable muss angewählt sein. Nun können die Indexe mit ihren Variablennamen eingegeben werden.

Index: In hexadezimaler Schreibweise

Subindex: Der Bereich der Subindexe ist in den Zieleinstellungen festgelegt.

Zugriff: Zugriffshäufigkeit

Attribut: Eigenschaft der Variablen (Schreiben, Lesen)

Variable: Name der Variablen (Achtung: vor dem Text muss ein Punkt stehen.)

6.5 Netzwerkvariablen

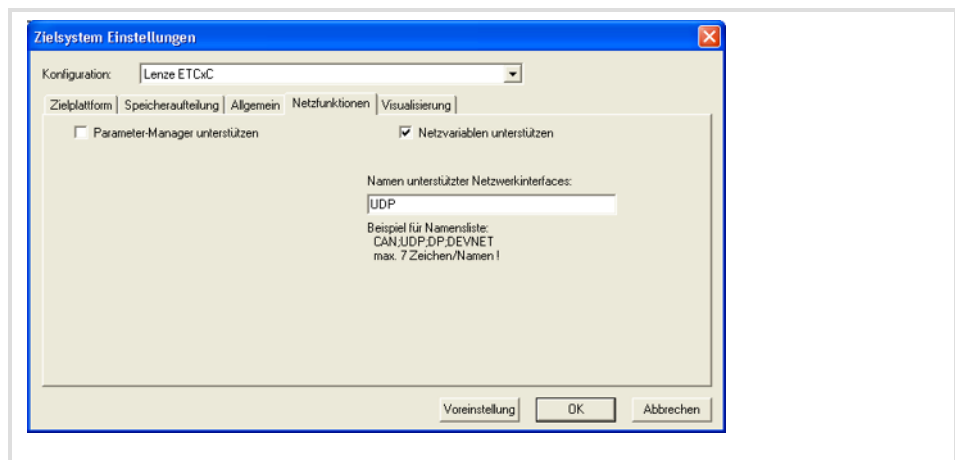
Netzwerkvariablen sind eine Möglichkeit, Daten zwischen zwei oder mehreren Steuerungen auszutauschen. Derzeit sind Netzwerkvariablen auf Basis von UDP implementiert. Die Variablenwerte werden dabei auf der Basis von Broadcast-Nachrichten automatisch ausgetauscht. Diese Dienste sind vom Protokoll her nichtbestätigte Dienste, d. h. es gibt keine Kontrolle, ob die Nachricht auch beim Empfänger ankommt. Der Netzwerkvariablenaus-tausch entspricht einer 1 (Sender) zu n (Empfänger)-Verbindung.

Voraussetzung

Die Bibliotheken "NetVarUdp_LIB.lib", "SysLibCallback.lib" und "SysLibSocket.lib" sind in der Bibliotheksverwaltung von CoDeSys eingefügt.

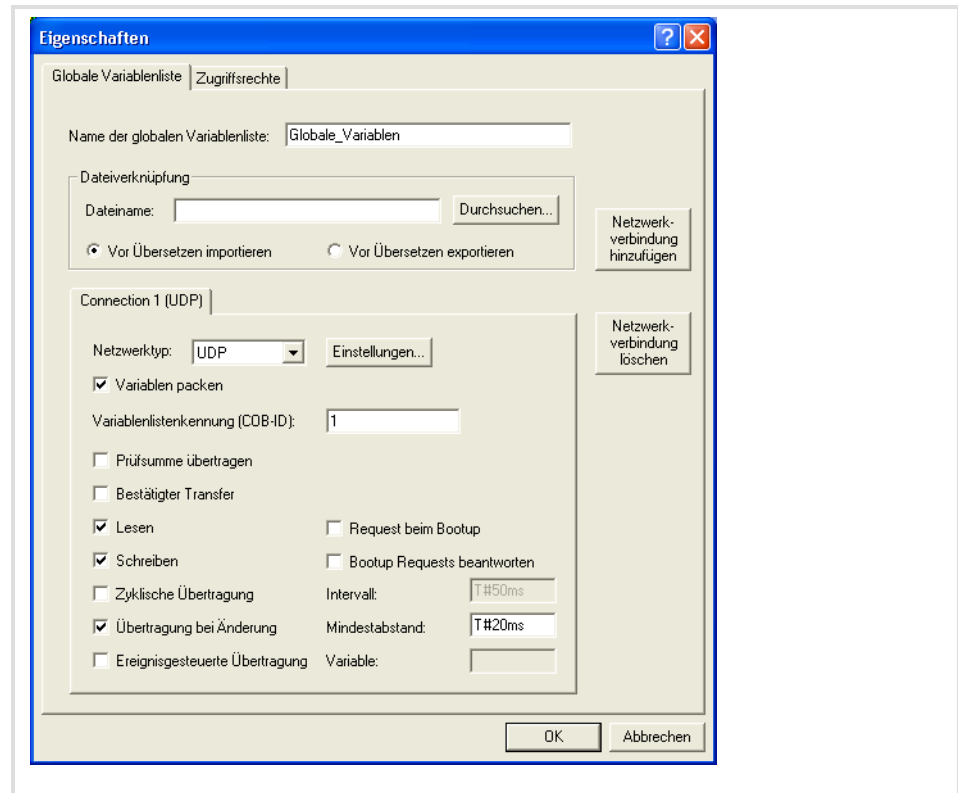
6.5.1 Einstellungen in den Zielsystemeinstellungen

Aktivieren Sie das Kontrollkästchen "Netzvariablen unterstützen" im Reiter "Netzfunktionen" der Dialogbox "Zielsystemeinstellungen". Bei "Namen unterstützter Netzwerkinterfaces" geben Sie den "UDP" an.



6.5.2 Einstellungen in den globalen Variablenlisten

Legen Sie eine neue globale Variablenliste an. Hier definieren Sie die Variablen, die mit anderen Steuerungen ausgetauscht werden sollen. Über den Eigenschaftsdialog der Variablenliste lassen sich die Übertragungseigenschaften festlegen.



Wenn Sie die Schaltfläche **Netzwerkverbindung hinzufügen** drücken, können Sie die Netzwerkeigenschaften dieser Variablenliste definieren. Die Optionen haben dabei folgende Bedeutung:

Netzwerktyp: Als **Netzwerktyp** muss UDP ausgewählt werden.

Lesen: Lesen bedeutet, dass die Variablenwerte von einer (oder mehreren) anderen Steuerungen gelesen werden.

Schreiben: Schreiben bedeutet, dass die Variablen dieser Variablenliste zu anderen Steuerungen gesendet werden. Es wird empfohlen, für eine Variablenliste nur eine der Möglichkeiten zu wählen, also entweder nur Lesen oder nur Schreiben. Sollen verschiedene Variablen eines Projekts gelesen und geschrieben werden, so verwenden Sie mehrere Variablenlisten: eine zum Lesen, eine zum Schreiben. Außerdem wird empfohlen, dass in einem Netzwerk nur eine Steuerung die gleiche Variablenliste sendet.

Zyklische Übertragung: Nur gültig, wenn "Schreiben" aktiviert ist. Die Werte werden in der angegebenen Intervallzeit gesendet, unabhängig davon, ob sie sich geändert haben.

Übertragung bei Änderung: Die Werte werden nur gesendet, wenn sich der Wert geändert hat, oder wenn die bei Mindestintervall angegebene Zeit abgelaufen ist.

Variablen packen: Wenn diese Option aktiviert ist, werden die Variablen nach Möglichkeit in einer Übertragungseinheit zusammengefasst. Bei UDP ist eine Übertragungseinheit 256 Bytes groß. Passen nicht alle Variablen der Liste in eine Übertragungseinheit, werden für diese Liste mehrere Übertragungseinheiten gebildet. Ist die Option nicht aktiviert, kommt jede Variable in eine eigene Übertragungseinheit.

Wenn "Übertragung bei Änderung" konfiguriert ist, wird für jede Übertragungseinheit getrennt geprüft, ob sie geändert ist und gesendet werden muss.

Basisidentifizier: Der Basisidentifizier wird als eindeutige Kennung benutzt, um Variablenlisten verschiedener Projekte auszutauschen. Variablenlisten mit gleichem Basisidentifizier werden ausgetauscht.

Es ist darauf zu achten, dass die Definitionen der Variablenlisten mit gleichem Basisidentifizier in den verschiedenen Projekten übereinstimmen.

Hierzu kann das Feature "Dateiverknüpfung" verwendet werden. Aus einem Projekt wird die Variablenliste exportiert. Die anderen Projekte importieren sie, wodurch der Inhalt nicht neu eingegeben werden muss.

Damit die Daten korrekt zwischen zwei Steuerungen ausgetauscht werden, müssen die globalen Variablenlisten in den beiden Projekten übereinstimmen. Ein Projekt kann die Datei vor dem Übersetzen exportieren, die anderen Projekte sollten die vor dem Übersetzen importieren.

Neben einfachen Datentypen kann eine Variablenliste auch Strukturen und Arrays enthalten. Die Elemente dieser zusammengesetzten Datentypen werden einzeln versendet. Wenn eine Variablenliste größer als eine Übertragungseinheit ist, werden die Daten auf mehrere Übertragungseinheiten aufgeteilt. Es kann darum nicht zugesichert werden, dass alle Daten der Variablenliste in einem Zyklus empfangen werden.

Teile der Variablenliste können in verschiedenen Zyklen empfangen werden. Dies ist auch für Variablen mit Struktur- und Arraytypen möglich.

6.6 Programm generieren

Nach der Erstellung eines Projektes in CoDeSys in den Programmiersprachen nach IEC61131-3, muss das Projekt in ein für die Steuerung ausführbares Programm übersetzt werden.

Über den Menüpunkt **Online -> Einloggen** wird automatisch aus dem aktuell bearbeiteten Projekt ein ausführbares Programm erzeugt und ins RAM der Steuerung übertragen.

Besteht eine Online-Verbindung zur Steuerung, ist es über den Menüpunkt **Online -> Bootprojekt erzeugen** möglich, das aktuelle Projekt ins Flash-PROM der Steuerung zu übertragen. Zusätzlich wird auf der Festplatte im Projektverzeichnis eine Datei "Projektname.prg" erzeugt, wenn unter **Projekt -> Optionen -> Symbolkonfiguration** das Kontrollfeld "Binärfile der Applikation erzeugen" angewählt ist.

Diese Datei kann dann auch über die Monitorschnittstelle der Steuerung per Z-Modem-Übertragung in die Steuerung geladen werden, wobei der Dateiname des Bootprojektes in "default.prg" geändert werden muss. Für die ETCxC kann diese Datei auch über das ETC-MMI in die Steuerung übertragen werden.

6.7 Schnittstelle zur ETC

6.7.1 Datenbausteine

Der integrierten SPS steht ein interner RAM-Bereich zur Verfügung, der in 16 sogenannte Datenbausteine unterteilt ist. Jeder Datenbaustein (DB) umfasst 256 Datenworte (DW) zu 16 Bit.

6.7.2 Systemvariablen der ETCxM

Zur Zeit stehen bei der ETCxM die Datenbausteine der SPS als Datenspeicher frei zur Verfügung.

6.7.3 Maschinenkonstanten in der ETCxM verwenden

Die Maschinenkonstanten **MK_TECHNOLOGY_DATA_1** bis **MK_TECHNOLOGY_DATA_4** stehen für technologie-abhängige Daten zur Verfügung.

Diese Maschinenkonstanten enthalten jeweils 10 Einträge. Der Wertebereich jedes Eintrags entspricht einem 64-bit floating point-Wert (IEC61131-Datentyp LREAL). Die einzelnen Einträge können über die Funktion "GetUserParam()" von der SPS ausgelesen werden.

6.7.4 Fehlermeldungen bei der ETCxM auslesen

Bei der ETCxM können aktive Fehlermeldungen über Funktionsbausteine der Bibliothek "mc.lib" ausgelesen werden. Dabei werden achsspezifische Fehler und Betriebssystemfehler unterschieden. Es gibt jeweils eine Funktion zum Auslesen und zum Quittieren (Rücksetzen) eines anstehenden Fehlers.

Fehlerquelle	SPS-Bausteine
Achsen	MC_ReadAxisError MC_Reset
System	MC_ReadSysError MC_ResetSysError

6.8 Bibliothek

Die Funktionen und Funktionsblöcke der ETCxM stehen in der System-Bibliothek "SysETCxM.LIB".

Im Folgenden werden alle Funktionen dieser Bibliothek beschrieben.

6.8.1 Allgemeine Funktionen

6.8.1.1 CenterString

Deklaration

```
FUNCTION CenterString: INT
VAR_INPUT
    string_s      : STRING(255)      (* String, der umformatiert werden soll *)
    linelen_di    : DINT              (* Zeilenlänge für Zentrierung *)
END_VAR
```

Beschreibung

Die Funktion fügt Leerzeichen zur Zentrierung in den String ein. Der Returnwert entspricht der resultierenden Gesamtlänge des Strings.

Stellen Sie sicher, dass die angegebene Stringvariable groß genug ist, um den resultierenden String aufzunehmen!

Beispiel

```
test_s      : STRING(20) := 'Hallo';
CenterString(test_s, 20);
```

Durch den Aufruf von CenterString werden 7 Leerzeichen vor "Hallo" eingefügt.

6.8.1.2 FMOD

Deklaration

```
FUNCTION FMOD: LREAL
VAR_INPUT
    x: LREAL;
    y: LREAL;
END_VAR
```

Beschreibung

Die Funktion liefert den floating point Rest der Division x/y (mit Vorzeichen von x).

Beispiel

```
Ergebnis_Ir      : LREAL;
Ergebnis_Ir      := FMOD(10.5, 3.2); Ergebnis_Ir = 0.9
```

6.8.1.3 Format

Deklaration

```

FUNCTION FORMAT: INT
VAR_INPUT
    STRING_S      : STRING(255);      (* Zielstring *)
    FORMAT_S      : STRING(80);       (* Formatstring *)
    PARAMETER_P   : DINT;              (* Adresse einer Struktur oder Variablen *)
END_VAR

```

Beschreibung

Alle Zeichen im Formatstring, die nicht zu einer Formatangabe gehören, werden in STRING_S kopiert.

Formatstring: Das %-Zeichen in einem Formatstring eröffnet eine Formatangabe mit der allgemeinen Form:

% Flag Ausgabefeldlänge Genauigkeit Datentypangabe

Flag	
<nichts>	rechtsbündig, führende Leerzeichen / Nullen
-	linksbündig, nachfolgende Leerzeichen / Nullen
+	Vorzeichen immer ausgeben
Ausgabefeldlänge	
0n	min. n Stellen, mit Nullen auffüllen
n	min. n Stellen, mit Leerzeichen auffüllen
Genauigkeit	
<nichts>	6 Stellen
.0	keinen Dezimalpunkt ausgeben
.n	max. n Nachkommastellen ausgeben
Datentyp (CoDeSys)	Datentypangabe (Formatstring)
DWORD	u
DINT	d
LREAL	lf
STRING	s

Der Returnwert entspricht der Anzahl der Zeichen, die in den String STRING_S kopiert wurden.

Beispiel

```
TYPE TEST_R
STRUCT
    Dw          : DWORD;
    L           : LREAL;
END_STRUCT
END_TYPE

s_s           : STRING(80);
ret_i         : INT;
di            : DINT;
t_r           : TEST_R;
```

Ein Parameter:

```
di            : 345;
ret_i         := FORMAT(s_s, '%d', ADR(di)); ret_i = 4; s_s = '>-345'
```

Mehrere Parameter:

```
t_r.dw        := 123;
t_r.lr        := 4.321;
ret_i         := FORMAT(s_s, '1: %u, 2: %f', ADR(t_r)); ret_i = 19; s_s = '1: 123, 2: 4.321000'
```

6.8.1.4 GetFirmwareVersion

Deklaration

```
FUNCTION GetFirmwareVersion: STRING(80)
VAR_INPUT
    type_i      : INT;
END_VAR
```

Parameter

type_i	Versionsstring
1	NCR
2	Bootloader

Beschreibung

Die Funktion liefert Versionsstrings.

6.8.1.5 GetMacAddr (nur ETCxM)

Deklaration

```

FUNCTION GetMacAddr: BOOL
VAR_INPUT
    pMac      : DINT;      (* Adresse eines Arrays vom Typ ARRAY[0..5] OF BYTE *)

END_VAR

```

Beschreibung

Mit dieser Funktion kann die MAC-Adresse des Ethernet-Controllers der Steuerung ausgelesen werden. Der Funktion muss die Adresse eines 6 Byte großen Speicherbereichs übergeben werden (s. Beispiel). Der Returnwert der Funktion signalisiert, ob die Adresse gelesen werden konnte.

Beispiel

```

ret_bit      : BOOL;
mac_a        : ARRAY[0..5] OF BYTE;
ret_bit      := GetMacAddr(ADR(mac_ab));

```

6.8.1.6 GetUserParam (nur ETCxM)

Deklaration

```

FUNCTION GetUserParam: LREAL
VAR_INPUT
    index_dw   : DWORD;      (* Index von MK_TECHNOLOGIEDATENx *)

END_VAR

```

Beschreibung

Mit dieser Funktion können die Werte der Maschinenkonstanten MK_TECHNOLOGIEDATEN1 bis MK_TECHNOLOGIEDATEN4 gelesen werden. Der zu übergebende Wert *index_dw* entspricht der Position (beginnend mit 0) ab dem ersten Eintrag von MK_TECHNOLOGIEDATEN1 fortlaufend durchnummeriert bis zum letzten Eintrag von MK_TECHNOLOGIEDATEN4.

Beispiel

```

MK_TECHNOLOGIE-DATEN1  1.1, 0, 0, 0, 0, 0, 0, 0, 0, 0;
MK_TECHNOLOGIE-DATEN2  0, 4.5, 3, -5.7, 0, 0, 0, 0, 0, 0;

wert_d      : DINT;
wert_lr      : LREAL;
wert_lr      := GetUserParam(9); (* liefert den Wert -1.1 *)
wert_di      := LREAL_TO_DINT(GetUserParam(12)); (* liefert den Wert 3 *)

```

6.8.1.7 IO_SET

Deklaration

```

FUNCTION IO_SET: BOOL
VAR_INPUT
    byte_w      : WORD;
    bit_w       : WORD;
END_VAR

```

Beschreibung

Die Funktion invertiert den Zustand eines Ausgangs. In dem Ausgangsbyte *byte_w* (Wertebereich 0 ... 31) wird das Bit *bit_w* (Wertebereich 0 ... 7) invertiert. Der Returnwert der Funktion ist ohne Bedeutung.

Beispiel

```
IO_SET(1,3);
```

invertiert den Ausgang Q0.10

6.8.1.8 Load_Param

Deklaration

```

FUNCTION LOAD_PARAM : DINT
VAR_INPUT
    dateiname_s : STRING(15);      (* "device:Dateiname" *)
    pindex_di   : DINT;            (* P-Feldindex *)
    anzahl_di   : DINT;            (* Anzahl der P-Feldwerte *)
END_VAR

```

Beschreibung

LOAD_PARAM liest *anzahl_di* P-Feldwerte von der RAM-Disk (device: rd), dem FLASH-PROM (device: sd), einer SD-Card (SC) oder einem USB-Massenspeicher (MS), die vorher mit SAVE_PARAM in der Datei gespeichert wurden, und trägt die Werte ab dem Index *pindex_di* in das P-Feld ein.

Ein Returnwert ungleich 0 signalisiert einen Fehler (📖 257).

Beispiel

```

ret_di      : DINT;
ret_di      := LOAD_PARAM('rd:daten.dat', 1200, 30);

```

Es werden 30 P-Feldwerte aus der Datei "daten.dat" von der RAM-Disk gelesen und ab Index 1200 im P-Feld abgelegt.

6.8.1.9 SETINPUT_BIT

Deklaration

```

FUNCTION SETINPUT_BIT: BOOL
VAR_INPUT
    WORD_W      : WORD;          (* Nr. des Datenwortes *)
    BIT_W       : WORD;          (* Eingang der IO-Karte *)
END_VAR

```

Beschreibung

Die Funktion setzt ein Bit im Eingangsprozessabbild. Im *Eingangsprozessabbild des Datenwortes word_w* (Wertebereich 0 ... 63) wird Bit *bit_w* (Wertebereich 0 ... 31) gesetzt. Damit kann eine Testumgebung (ohne Hardware) geschaffen werden.

Der Returnwert FALSE signalisiert einen falschen Übergabeparameter.

6.8.1.10 SETINPUT_WORD

Deklaration

```

FUNCTION SETINPUT_WORD: BOOL
VAR_INPUT
    WORD_W      : WORD;          (* Nr. des Datenwortes *)
    VAL_W       : WORD;
END_VAR

```

Beschreibung

Die Funktion führt eine boolsche ODER-Verknüpfung mit dem Wert *val_w* und dem Datenwort *word_w* des Eingangsprozessabbildes durch und schreibt das Ergebnis in das selbe Datenwort.

Returnwert FALSE signalisiert einen falschen Übergabeparameter.

Die Funktion ermöglicht es einem Entwickler eine Testumgebung (ohne Hardware) zu schaffen.

6.8.1.11 SPSERROR

Deklaration

```

FUNCTION SPSERROR: BOOL
VAR_INPUT
    fehler_di      : DINT;      (* Fehlernummer *)
    klasse_dw      : DWORD;     (* Fehlerklasse *)
    formatstring   : STRING(80); (* Formatstring *)
    parameter_p    : DINT;      (* Adr. einer Struktur, die die Parameter beinhaltet *)
END_VAR

```

Beschreibung

Die Funktion setzt eine Fehlermeldung ab, die auf einem angeschlossenen Bedienteil zur Anzeige gebracht wird. Die Fehlernummer *fehler_di* dient gleichzeitig als Referenz auf einen Fehlertext. Die Fehlertexte können auf einem angeschlossenen PC (für beide ETC Varianten) oder auf dem ETC-MMI (nur ETCxC) angezeigt werden.

Die Fehlerklassen *klasse_dw* sind folgendermaßen unterteilt:

- 1: lokal leichter Fehler
- 2: lokal schwerer Fehler
- 3: global leichter Fehler
- 4: global schwerer Fehler

Die Fehlerklassen 3 und 4 führen zum Abbruch eines Programms bzw. einer Achsbewegung.

Zu dem über die Fehlernummer referenzierten Fehlertext kann eine zusätzliche Information ausgegeben werden. Über den Formatstring können weiterer Text bzw. numerische Werte übergeben werden.

Formatstring: Das %-Zeichen in einem Formatstring eröffnet eine Formatangabe mit der allgemeinen Form:

% Flag Ausgabefeldlänge Genauigkeit Datentypangabe

Flag	
<nichts>	rechtsbündig, führende Leerzeichen / Nullen
-	linksbündig, nachfolgende Leerzeichen / Nullen
+	Vorzeichen immer ausgeben
Ausgabefeldlänge	
0n	min. n Stellen, mit Nullen auffüllen
n	min. n Stellen, mit Leerzeichen auffüllen
Genauigkeit	
<nichts>	6 Stellen
.0	keinen Dezimalpunkt ausgeben
.n	max. n Nachkommastellen ausgeben
Datentyp (CoDeSys)	Datentypangabe (Formatstring)
DWORD	u
DINT	d
REAL	f e
LREAL	lf le
STRING	s

Der Returnwert der Funktion ist ohne Bedeutung.

Beispiel 1

```
SPSERROR(304, 4, 'Spindel läuft nicht an', 0);
```

Beispiel 2

```
TYPE PARAMETER_TR
STRUCT
    auftragsnr_dw      : DWORD;
    text_ps            : POINTER TO STRING;
    pos_lr             : LREAL;
END_STRUCT
END_TYPE
param_tr
string_s
param_tr.auftragsnr_dw
param_tr.text_ps
param_tr.pos_lr
SPSERROR(303, 1, 'Auftragsnr%u %s %9.3lf', ADR(param_tr));
```

6.8.1.12 SETLANGUAGE (nur ETCxM)

Deklaration

```
FUNCTION SETLANGUAGE: BOOL
VAR_INPUT
    Sprache          : INT;
END_VAR
```

Beschreibung

Mit dieser Funktion wird zwischen verschiedenen Sprachen für Ausgabeteixe umgeschaltet. Mit Sprache = 0 werden die Ausgabeteixdateien ncrspch0.txt/spsspch0.txt, mit Sprache = 1 die Dateien ncrspch1.txt/spsspch1.txt usw. angewählt.

Der Returnwert der Funktion ist ohne Bedeutung.

Beispiel

```
SETLANGUAGE(4);
```

Ausgabeteixdateien ncrspch4.txt und spsspch4.txt werden ausgewählt.

6.8.1.13 STRTOF

Deklaration

```
FUNCTION STRTOF: REAL
VAR_INPUT
    SString          : STRING(255);
    PIndex           : POINTER TO INT;
END_VAR
```

Beschreibung

Die Funktion konvertiert die Zeichen in *sString* ab der Position *Index* in einen REAL. Index enthält nach der Ausführung die Einleseendposition.

6.8.1.14 STRTOL10

Deklaration

```
FUNCTION STRTOL10: DINT
VAR_INPUT
    sString          : STRING(255);
    pIndex           : POINTER TO INT;
END_VAR
```

Beschreibung

Die Funktion konvertiert die Zeichen in *sString* ab der Position *Index* in einen DINT. Index enthält nach der Ausführung die Einleseendposition.

6.8.1.15 SYSERROR

Deklaration

```

FUNCTION SYSERROR: BOOL
VAR_INPUT
    fehler_di      : DINT;      (* Fehlernummer *)
    klasse_dw      : DWORD;     (* Fehlerklasse *)
    formatstring   : STRING(80); (* Formatstring *)
    parameter_p    : DINT;      (* Adr. einer Struktur, die die Parameter beinhaltet *)
END_VAR

```

Beschreibung

Die Funktion SYSERROR wird wie die Funktion SPSERROR verwendet.

Mit dieser Funktion können steuerungsinterne Fehler zur Anzeige gebracht werden.

6.8.1.16 TRACE

Deklaration

```

FUNCTION TRACE: BOOL
VAR_INPUT
    Idx            : DINT;      (* Traceindex *)
    Formatstring   : STRING(80); (* Formatstring *)
    parameter_p    : DINT;      (* Adresse einer Struktur *)
END_VAR

```

Beschreibung

Die Trace-Funktion kann sehr hilfreich beim Debuggen von dynamischen Vorgängen sein. Jeder Trace kann im Diagnose-Monitor gezielt aktiviert bzw. deaktiviert werden. Dazu wird beim Aufruf der Trace-Funktion eine Nummer angegeben, anhand derer der Trace identifiziert wird. Die Tracenummern *idx* 100 ... 200 sind für die SPS reserviert, alle anderen bleiben dem NC-Rechner vorbehalten.

Der Trace-Puffer im NC-Rechner fasst 512-Trace-Meldungen. Werden die Trace-Meldungen schneller in den Puffer eingetragen als sie über die Diagnose-Schnittstelle (RS232) gesendet werden können, kann der Puffer überlaufen und Meldungen verloren gehen.

Der Returnwert der Funktion ist ohne Bedeutung.

Beispiel

```

TYPE PARAMETER_TR
STRUCT
    param1_r      : REAL;
    param2_di     : DINT;
END_STRUCT
END_TYPE

param_tr:          PARAMETER_TR;
parameter_tr.param1_r := 30.2
parameter_tr.param2_di := 20;
TRACE(100, 'REAL:%f DINT:%d', ADR(param_tr));

```

6.8.1.17 WordWrap

Deklaration

```
FUNCTION WordWrap: INT
VAR_INPUT
    string_s      : STRING(255);      (* String, der umformatiert werden soll *)
    linelen_di    : DINT;              (* maximale Zeilenlänge *)
END_VAR
```

Beschreibung

Die Funktion fügt automatisch einen Zeilenumbruch (Linefeeds) in den String ein, wenn die Zeilenlänge den angegebenen Wert überschreitet. Der Returnwert ist die resultierende Gesamtlänge des Strings.

Stellen Sie sicher, dass die angegebene Stringvariable groß genug ist, um den resultierenden String aufzunehmen!

6.8.1.18 Returncodes der Funktionen LOAD, SAVE, LOAD_PARAM, SAVE_PARAM

Returncode	Beschreibung
-1	EEPROM defekt
-2	Checksummenfehler im EEPROM
-3	EEPROM voll
-4	Datei ist bereits geöffnet, Zugriff verweigert
-5	zu viele gleichzeitig geöffnete Dateien
-6	EEPROM Inhaltsverzeichnis ist voll
-7	Datei nicht vorhanden
-93	Checksummenfehler in der RAM-Disk
-94	nicht genug Speicher zum Anlegen der File-Buffer
-95	RAM-Disk Inhaltsverzeichnis ist voll
-96	RAM-Disk ist voll
-110	Unbekannter Devicename
-111	V24-Schnittstelle zum Diskettenlaufwerk ist belegt
-112	Diskettenlaufwerk ist nicht frei
-113	Timeout beim Zugriff auf Diskettenlaufwerk
-114	Fehler beim Zugriff auf Diskettenlaufwerk
-115	Diskettenlaufwerk meldet Übertragungsfehler
-116	Diskettenlaufwerk meldet V24-Overrun
-117	Diskette ist schreibgeschützt
-118	keine Diskette eingelegt
-119	Schreib-/Lesefehler auf der Diskette
-121	Dateiname enthält unzulässige Zeichen

Speziell für READ_PARAM, WRITE_PARAM

Returncode	Beschreibung
1	P-Feldindex außerhalb des erlaubten Bereichs

6.8.2 V24-Funktionen

6.8.2.1 ALLOCV24

Deklaration

```

FUNCTION AllocV24: DINT
VAR_INPUT
    unit_di      : DINT;          (* Schnittstelle: COM1 (X3), COM2 (X4) *)
    pri_di       : DINT;          (* Priorität (-128 ... +127) *)
END_VAR

```

Beschreibung

Eine V24-Schnittstelle muss vor Benutzung mit AllocV24() allociert werden. Danach kann sie mit InitV24() nach eigenen Bedürfnissen initialisiert werden.

Die Anforderung wird nur bedient, wenn die Schnittstelle frei ist oder die Priorität der Anforderung höher als die aktuelle Priorität ist. Eine allocierte V24-Schnittstelle kann mittels FreeV24() wieder freigegeben werden, damit wird der letzte Zustand vor der Allocation wieder hergestellt.

Mögliche Schnittstellen sind COM1 (*unit_di* = 0) und COM2 (*unit_di* = 1). Für die Priorität können Werte zwischen -128 (niedrigste) und +127 (höchste) verwendet werden, wobei eine Priorität von 127 bedeutet, dass die Schnittstelle nicht "gestohlen" werden kann.

Die Funktion liefert die Adresse der allocierten V24-Request Struktur zurück. Wird 0 zurück geliefert konnte die Schnittstelle nicht allociert werden.

Beispiel

```

request_p      : DINT;
request_p      := AllocV24(0, 110);

```

Die erste V24-Schnittstelle (X3-Stecker) wird mit Priorität 110 allociert.

6.8.2.2 INITV24

Deklaration

```

FUNCTION InitV24 : DINT
VAR_INPUT
    req_pr      : DINT;          (* Adresse der V24-Request Struktur *)
    mode_dw     : DWORD;
    flags_dw    : DWORD;
END_VAR

```

Beschreibung

Diese Funktion dient zur Einstellung der Schnittstellenparameter einer mit AllocV24 allocierten V24-Schnittstelle.

► *mode_dw* ergibt sich aus der Veroderung von Mode-Bits (📖 257).

► *flags_dw* entsteht durch Veroderung von Flag-Bits (📖 257).

Eine Returnwert ungleich 0 signalisiert einen Fehler (📖 257).

Beispiel

```

ret_di      : DINT;
requestV24_p : DINT;
requestV24_p := ALLOCV24(1, 127);
ret_di      := InitV24(requestV24_p,
mode_no_parity OR mode_8_bits_per_char OR
mode_no_rts_cts_control OR mode_stop_bit_len_2 OR
mode_rx_baudrate_9600 OR mode_tx_baudrate_9600 OR
mode_rx_buffersize_2k OR mode_tx_buffersize_2k, flags_no_protocol);

```

6.8.2.3 FREEV24**Deklaration**

```

FUNCTION FreeV2: BOOL
VAR_INPUT
    req_pr      : DINT;          (* Adresse der V24-Request Struktur *)
END_VAR

```

Beschreibung

Mit dieser Funktion werden die Zugriffsrechte auf die allocierte V24-Schnittstelle wieder abgegeben und die Schnittstelle wieder in den ursprünglichen Zustand versetzt.

Der Returnwert der Funktion ist ohne Bedeutung.

Beispiel

```

requestV24_p : DINT;
requestV24_p := ALLOCV24(1, 127);
REEV24(requestV24_p);

```

6.8.2.4 READBLOCKV24**Deklaration**

```

FUNCTION ReadBlockV24: DINT
VAR_INPUT
    pRequest      : DINT;          (* Adresse der V24-Request Struktur *)
    pBuffer       : DINT;          (* Adresse eines Datenpuffer *)
    BufSize       : INT;           (* Größe des Puffer in Byte *)
END_VAR

```

Beschreibung

Die Funktion liest maximal *BufSize* Zeichen in den angegebenen Datenpuffer *pBuffer* aus dem V24 Empfangspuffer.

Die Funktion liefert die Anzahl der aus dem Empfangspuffer gelesenen Zeichen.

6.8.2.5 READV24

Deklaration

```

FUNCTION ReadV24: DINT
VAR_INPUT
    req_pr      : DINT;          (* Adresse der V24-Request Struktur *)
END_VAR
  
```

Beschreibung

Ein Zeichen aus dem Empfangspuffer lesen. Der Empfangspuffer ist als FIFO implementiert, es wird immer zeichenweise gelesen und geschrieben.

Diese Funktion kehrt in jedem Fall sofort zurück. Returnwerte kleiner 0 zeigen einen Fehler an (📖 257). Wenn das FIFO leer ist, wird -1 (EOF) zurückgegeben.

Beispiel

```

requestV24_p    : DINT;
zeichen_di      : DINT;
zeichen_di      := READV24(requestV24_p);
  
```

6.8.2.6 WRITEBLOCKV24

Deklaration

```

FUNCTION WriteBlockV24: DINT
VAR_INPUT
    pRequest    : DINT;          (* Adresse der V24-Request Struktur *)
    pBuffer     : DINT;          (* Adresse eines Datenpuffer *)
    BufSize     : INT;           (* Anzahl der zu schreibenden Zeichen *)
END_VAR
  
```

Beschreibung

Die Funktion versucht *BufSize* Zeichen von der angegebenen Adresse *pBuffer* in den V24-Sendepuffer einzutragen. Sofern genug Platz im Sendepuffer ist, werden alle *BufSize*-Zeichen eingetragen.

Die Funktion liefert die Anzahl der in den Sendepuffer geschriebenen Zeichen zurück.

6.8.2.7 WRITEV24

Deklaration

```

FUNCTION WriteV24: DINT
VAR_INPUT
    req_pr      : DINT;          (* Adresse der V24-Request Struktur *)
    chr_di      : DINT;          (* zu schreibendes Zeichen *)
END_VAR
  
```

Beschreibung

Ein Zeichen in den Sendepuffer schreiben. Der Sendepuffer ist als FIFO implementiert, es wird immer zeichenweise gelesen und geschrieben.

Diese Funktion kehrt in jedem Fall sofort zurück, auch wenn das FIFO voll ist. In diesem Fall wird das Zeichen nicht geschrieben und -1 (EOF) zurückgegeben. Generell zeigt ein Returnwert kleiner 0 einen Fehler an (📖 257).

Beispiel

```
requestV24_p      : DINT;
zeichen_di        : DINT;
ret_di            : DINT;

zeichen_di        : = 65;
ret_di            : = WRITEV24(requestV24_p, zeichen_di);
```

6.8.2.8 CLRRXBUFFER

Deklaration

```
FUNCTION ClrRxBuffer: BOOL
VAR_INPUT
    req_pr          : DINT;          (* Adresse der V24-Request Struktur *)
END_VAR
```

Beschreibung

Mit dieser Funktion kann das Empfangs-FIFO komplett gelöscht werden.
Der Returnwert der Funktion ist ohne Bedeutung.

Beispiel

```
requestV24_p      : DINT;
requestV24_p      : = ALLOCV24(1, 127);
CLRRXBUFFER(requestV24_p);
```

6.8.2.9 CLRTXBUFFER

Deklaration

```
FUNCTION ClrTxBuffer: BOOL
VAR_INPUT
    req_pr          : DINT;          (* Adresse der V24-Request Struktur *)
END_VAR
```

Beschreibung

Mit dieser Funktion kann das Sender-FIFO gelöscht werden.
Der Returnwert der Funktion ist ohne Bedeutung.

Beispiel

```
requestV24_p      : DINT;
requestV24_p      : = ALLOCV24(1, 127);
CLRTXBUFFER(requestV24_p);
```

6.8.2.10 Globale Konstanten zu V24 Funktionen

(* Fehlercodes: *)	
V24_WRONG_UNIT_KDI	: DINT := -100; (* falsche V24-Schnittstellenr. *)
V24_OUTOFMEMORY_KDI	: DINT := -101; (* nicht genug RAM zum Anlegen der V24-FIFO-Puffer *)
V24_OVERRUN_ERROR_KDI	: DINT := -102; (* Overrun *)
V24_PARITY_ERROR_KDI	: DINT := -103; (* Parity-Fehler *)
V24_FRAMING_ERROR_KDI	: DINT := -104; (* Framing-Fehler *)
V24_RECEIVED_BREAK_KDI	: DINT := -105; (* Received break *)

(* Mode-Bits: *)	
MODE_EVEN_PARITY_KDW	: DWORD := 16#00000000;
MODE_ODD_PARITY_KDW	: DWORD := 16#00000004;
MODE_NO_PARITY_KDW	: DWORD := 16#00000010;
MODE_7_BITS_PER_CHAR_KDW	: DWORD := 16#00000000;
MODE_8_BITS_PER_CHAR_KDW	: DWORD := 16#00000001;
MODE_RTS_CTS_CONTROL_KDW	: DWORD := 16#00001080;
MODE_NO_RTS_CTS_CONTROL_KDW	: DWORD := 16#00000000;
MODE_STOP_BIT_LEN_1_KDW	: DWORD := 16#00000000;
MODE_STOP_BIT_LEN_2_KDW	: DWORD := 16#00000800;
MODE_BAUDRATE_2400_KDW	: DWORD := 16#00080000;
MODE_BAUDRATE_4800_KDW	: DWORD := 16#00090000;
MODE_BAUDRATE_9600_KDW	: DWORD := 16#000B0000;
MODE_BAUDRATE_19200_KDW	: DWORD := 16#000C0000;
MODE_BAUDRATE_38400_KDW	: DWORD := 16#000E0000;
MODE_BAUDRATE_57600_KDW	: DWORD := 16#000D0000;
MODE_BAUDRATE_115200_KDW	: DWORD := 16#000F0000;
MODE_RX_BUFFERSIZE_1K_KDW	: DWORD := 16#0A000000;
MODE_RX_BUFFERSIZE_2K_KDW	: DWORD := 16#0B000000;
MODE_RX_BUFFERSIZE_3K_KDW	: DWORD := 16#0C000000;
MODE_RX_BUFFERSIZE_4K_KDW	: DWORD := 16#0D000000;
MODE_TX_BUFFERSIZE_1K_KDW	: DWORD := 16#A0000000;
MODE_TX_BUFFERSIZE_2K_KDW	: DWORD := 16#B0000000;
MODE_TX_BUFFERSIZE_3K_KDW	: DWORD := 16#C0000000;
MODE_TX_BUFFERSIZE_4K_KDW	: DWORD := 16#D0000000;
(* Flag-Bits: *)	
FLAGS_NO_PROTOCOL_KDW	: DWORD := 16#00000000;
FLAGS_PLC_PROTOCOL_KDW	: DWORD := 16#00000001;
FLAGS_XON_XOFF_PROTOCOL_KDW	: DWORD := 16#00000002;
FLAGS_NAK_ON_RX_ERROR_KDW	: DWORD := 16#00000100;
FLAGS_MELD_RX_ERROR_KDW	: DWORD := 16#00000200;
FLAGS_USER_POLLING_KDW	: DWORD := 16#00001000;
FLAGS_CHANNEL_STOLEN_KDW	: DWORD := 16#80000000;

6.8.3 FILE IO-Funktionen

6.8.3.1 LOAD

Deklaration

```

FUNCTION LOAD: DINT
VAR_INPUT
    Name          : STRING(15);  (* 'device:Dateiname' *)
    daten_paten_p : DINT;        (* Adr. ab der die Daten gespeichert werden sollen *)
    len           : INT;         (* Länge der Daten in Bytes *)
END_VAR

```

Beschreibung

Mit dieser Funktion werden Daten von einem Gerät gelesen (📖 256).

Aus der Datei *name* werden von dem angegebenen Gerät *len* Bytes gelesen und ab der Speicheradresse *daten_p* abgelegt. Ohne Device-Angabe wird versucht, die Datei vom FLASHPROM zu lesen.

Ein Returnwert kleiner 0 signalisiert einen Fehler (📖 257).

Beispiel

```

werte_adi      : ARRAY[0..5] OF DINT;
anzahl_di      : DINT;
anzahl_di      := LOAD('RD:daten.dat', ADR(werte_adi), 24);

```

Sechs DINT-Werte aus der Datei "daten.dat" werden von der RAM-Disk gelesen und im Array *werte_di* abgelegt.

6.8.3.2 SAVE

Deklaration

```

FUNCTION SAVE: DINT
VAR_INPUT
    name          : STRING(15);  (* 'device:Dateiname' *)
    daten_p       : DINT;        (* Adr. ab der die Daten gespeichert werden sollen *)
    len           : INT;         (* Länge der Daten in Bytes *)
END_VAR

```

Beschreibung

Mit dieser Funktion können Daten auf einem Gerät (📖 256) gespeichert werden.

Unter *name* werden *len* Bytes ab der Speicheradresse *daten_p* auf dem angegebenen Gerät (device) gespeichert. Ohne device-Angabe wird die Datei ins FLASH-PROM geschrieben.

Ein Returnwert kleiner 0 zeigt einen Fehler beim Speichern an (📖 257).

Beispiel

```

werte_adi      : ARRAY[0..5] OF DINT;
anzahl_di      : DINT;
anzahl_di      := SAVE('RD:daten.dat', ADR(werte_adi), 24);

```

Sechs DINT-Werte aus dem Array *werte_di* werden in der Datei "daten.dat" auf der RAM-Disk gespeichert.

6.8.3.3 SetCurrentPath

Deklaration

```
FUNCTION SetCurrentPath: DINT
VAR_INPUT
    path_s      : STRING(40);      (* Geräte- und Pfad *)
END_VAR
```

Beschreibung

Legt das aktuelle Default-Laufwerk und den Verzeichnispfad für die File-IO Funktionen der Gerätetreiber (📖 256) fest. Mit dem Verzeichnispfad kann die IP-Adresse des CNC-Data Servers (Net-Disk) festgelegt werden.

Returnwert: Die Funktion liefert einen Fehlercode zurück (📖 257).

Beispiel

```
SetCurrentPath('ND:172.16.5.66');
```

6.8.3.4 SYSOPENFILE

Deklaration

```
FUNCTION SYSOPENFILE : DINT
VAR_INPUT
    filename_s      : STRING(15);
    accessmode_di    : DINT;
END_VAR
```

Beschreibung

Die Funktion dient zum Öffnen einer Datei auf dem angegebenen Gerät (📖 256). Sie liefert ein Filehandle, das für die Funktionen SysReadFile() und SysWriteFile() benötigt wird. Es können insgesamt 4 Dateien gleichzeitig geöffnet werden.

Parameter

filename_s:	Name der zu öffnenden Datei. Optional mit vorangehender Devicekennung und Doppelpunkt.
accessmode_di:	Zugriffsmodus O_RDONLY oder O_WRONLY (in globalen Konstanten definiert)

Returnwert: Filehandle (Größer oder gleich 0, wenn erfolgreich, kleiner 0 bei einem Fehler (📖 257)).

Beispiel

```
handle_di      : DINT;
handle_di      := SYSOPENFILE('RD:daten.dat', O_RDONLY);
```

6.8.3.5 SYSCLOSEFILE

Deklaration

```

FUNCTION SYSCLOSEFILE: DINT
VAR_INPUT
    handle_di      : DINT;
END_VAR

```

Beschreibung

Die Funktion dient zum Schließen einer zuvor mit SysOpenFile() geöffneten Datei. Das Inhaltsverzeichnis des FLASH-PROMs wird erst im FLASH-PROM upgedatet, wenn die letzte zum Schreiben geöffnete Datei geschlossen wird. Der Returnwert der Funktion ist kleiner 0 bei einem Fehler (alle Fehlercodes sind in den globalen Konstanten der Bibliothek definiert).

Parameter

Handle_di	Filehandle (von SysOpenFile() geliefert wurde)
-----------	--

6.8.3.6 SYSREADFILE

Deklaration

```

FUNCTION SYSREADFILE: DINT
VAR_INPUT
    handle_di      : DINT;    (* Filehandle, das von SysOpenFile geliefert wurde *)
    buffer_p       : DINT;    (* Adresse eines Puffer, in dem die Daten abgelegt werden *)
    maxlen_di      : DINT;    (* Länge des Puffers in Byte *)
END_VAR

```

Beschreibung

Die Funktion dient zum blockweisen Lesen (sequenziell) aus einer zuvor mit SysOpenFile() geöffneten Datei. Dabei werden maximal *maxlen_di* Zeichen aus der Datei in den mit *buffer_p* angegebenen Puffer gelesen. Weniger als *maxlen_di* Zeichen werden nur gelesen, wenn vorher das Dateiende erreicht wird oder ein Fehler beim Lesen auftritt.

Returnwert: Tatsächlich gelesene Anzahl von Zeichen; 0 bei Dateiende oder kleiner 0 bei einem Fehler (📖 257).

Beispiel

```

handle_di      : DINT;
anz_di         : DINT;
ret_di         : DINT;
buffer_ab      : ARRAY[0..100] OF BYTE;

handle_di      := SYSOPENFILE('test.dat', O_RDONLY);
anz_di         := SYSREADFILE(handle_di, ADR(buffer_ab), 20);
ret_di         := SYSCLOSEFILE(handle_di);

```

6.8.3.7 SYSREADLINE

Deklaration

 FUNCTION **SYSREADLINE**: DINT

VAR_INPUT

handle_di	: DINT;	(* Filehandle, das von SysOpenFile geliefert wurde *)
buffer_p	: DINT;	(* Adresse eines Puffer, in dem die Daten abgelegt werden *)
maxlen_di	: DINT;	(* Länge des Puffers in Byte *)

END_VAR

Beschreibung

Die Funktion dient zum zeilenweisen Lesen (sequenziell) aus einer zuvor mit SysOpenFile() geöffneten Datei. Dabei wird jeweils bis zum Zeilenende aber maximal *maxlen_di* Zeichen gelesen. Die Zeile wird nur nullterminiert, falls weniger als *maxlen_d* Zeichen gelesen wurden. Das Linefeed am Ende der Zeile wird mitgelesen.

Returnwert: Tatsächlich gelesene Anzahl von Zeichen, 0 bei Dateiende oder kleiner 0 bei einem Fehler (📖 257).

6.8.3.8 SYSWRITEFILE

Deklaration

 FUNCTION **SYSWRITEFILE**: DINT

VAR_INPUT

handle_di	: DINT;	(* Filehandle, das von SysOpenFile geliefert wurde *)
buffer_p	: DINT;	(* Adresse eines Puffer *)
len_di	: DINT;	(* Anzahl der zuschreibenden Zeichen *)

END_VAR

Beschreibung

Die Funktion dient zum blockweisen Schreiben (sequenziell) in eine zuvor mit SysOpenFile() geöffnete Datei. Dabei werden *len_di* Zeichen in die Datei geschrieben. Wenn beim Schreiben ein Fehler auftritt, wird die Datei gelöscht und ein Wert kleiner 0 zurückgegeben.

Returnwert: Anzahl der geschriebenen Daten, oder kleiner 0 bei einem Fehler (📖 257).

Beispiel

handle_di	: DINT;
anz_di	: DINT;
ret_di	: DINT;
buffer_s	: STRING(10) := 'Hallo';

handle_di	:= SYSOPENFILE('test.dat', O_WRONLY);
anz_di	:= SYSWRITEFILE(handle_di, ADR(buffer_s), 5);
ret_di	:= SYSCLOSEFILE(handle_di);

6.8.3.9 SYSREMOVEFILE**Deklaration**

```

FUNCTION SYSREMOVEFILE: DINT
VAR_INPUT
    filename_s      : STRING(15);      (* Dateiname *)
END_VAR

```

Beschreibung

Die Funktion dient zum Löschen einer Datei aus dem FLASH-PROM, der RAM-Disk oder der FLOPPY.

Parameter

filename_pc:	Name der zu löschenden Datei. Optional mit vorangehender Device-Kennung und Doppelpunkt  256.
--------------	--

6.8.3.10 SYSDISKFORMAT**Deklaration**

```

FUNCTION SYSDISKFORMAT: DINT
VAR_INPUT
    device_s        : STRING(15);
END_VAR

```

Beschreibung

Mit der Funktion wird das FLASH-PROM, die RAM-Disk bzw. die FLOPPY formatiert. Dabei werden alle Daten gelöscht!

Parameter

device_pc:	Devicenamen  256. Leerstring (" ") für das Default-Device (FLASH-PROM).
------------	--

Returnwert: Kleiner 0 bei einem Fehler (alle Fehler sind in den globalen Konstanten der Bibliothek definiert).

6.8.3.11 SYSFIRSTFILE, SYSNEXTFILE**Deklaration**

```

FUNCTION SYSFIRSTFILE: DINT
VAR_INPUT
    info_p          : DINT;              (* Adr. einer Variablen vom Typ FILEINFO_TR *)
    pattern_s        : STRING(15);      (* Suchpattern *)
END_VAR

```

```

FUNCTION SYSNEXTFILE: DINT
VAR_INPUT
    info_p          : DINT;              (* Adr. einer Variablen vom Typ FILEINFO_TR *)
    pattern_s        : STRING(15);      (* Suchpattern *)
END_VAR

```

Beschreibung

Diese Funktionen werden benötigt um das Inhaltsverzeichnis des FLASH-PROMs, der RAM-Disk bzw. der FLOPPY auszulesen. Mit SysFirstFile() wird die angegebene Fileinfo-Struktur initialisiert und Informationen zur ersten Datei in der Struktur abgelegt. Alle folgenden Aufrufe von SysNextFile(), mit derselben Fileinfo-Struktur, legen die Informationen, der jeweils nächsten Datei in der Struktur ab.

Die Fileinfo-Struktur ist folgendermaßen aufgebaut:

```

TYPE FILEINFO_TR
STRUCT
    filename_s      : STRING(15); (* Dateiname *)
    filesize_dw     : DWORD; (* Dateigröße in Byte *)
    filenum_di      : DINT; (* Nr. des Directory-Eintrags *)
END_STRUCT
END_TYPE

```

Parameter

info_p:	Zeiger auf die Fileinfo-Struktur, in der Dateiname und Dateilänge gespeichert werden sollen.
pattern_s:	Suchpattern mit MS-DOS-Wildcards. Es wird jeweils die nächste Datei geliefert, deren Dateiname zum Pattern passt.

Returnwert: 0, wenn ok. DSK_FILE_NOT_EXIST_KDI, wenn keine weitere Datei mehr vorhanden. Alle anderen Werte signalisieren einen Fehler (alle Fehler sind in den globalen Konstanten der Bibliothek definiert).

Beispiel

```

info_st      : FILEINFO_TR;
err_di       DINT;

err_di = SYSFIRSTFILE(ADR(info_st), 'SD:*.DIN'); (* Info zum ersten File holen *)
err_di = SYSNEXTFILE(ADR(info_st), 'SD:*.DIN'); (* Info zum nächsten File holen *)

```

6.8.3.12 SYSDISKINFO

Deklaration

```

FUNCTION SYSDISKINFO: DINT
VAR_INPUT
    info_p      : DINT;      (* Adresse einer Variablen vom Typ DISKINFO_TR *)
    pattern_s    : STRING(15); (* Suchpattern *)
END_VAR

```

Beschreibung

Die Funktion ermittelt die Belegungsdaten eines Devices und legt sie in der angegebenen Diskinfo-Struktur ab.

Die Diskinfo-Struktur ist folgendermaßen aufgebaut:

```

TYPE DISKINFO_TR
STRUCT
    blocksize_w      : WORD; (* Größe eines logischen Blocks in Bytes *)
    blockanz_w       : WORD; (* Formatierte Gesamtkapazität in Blöcken *)
    blockused_w      : WORD; (* Belegte Kapazität in Blöcken *)
    blockfree_w      : WORD; (* Freie Kapazität in Blöcken *)
END_STRUCT
END_TYPE

```

Parameter

info_p:	Zeiger auf die Diskinfo-Struktur.
device_pc:	Zeiger auf den Devicenamen  256. Leerstring (" ") für das Default-Device.

6.8.3.13 Gerätetreiber

Die Steuerung unterstützt verschiedene Geräte, auf die schreibend (w) und je nach Gerät lesend (r) zugegriffen werden. Der Zugriff erfolgt über ein Dateisystem. Eine Datei muss vor dem Zugriff durch die Funktion SysOpenFile() geöffnet werden. Außer auf dem Gerät "Drucker" können auf einem Gerät mehrere Dateien gleichzeitig geöffnet werden. Insgesamt sind vier gleichzeitig geöffnete Dateien möglich. Die Dateinamen müssen der DOS 8.3-Konvention entsprechen, also maximal 8 Zeichen für den Dateinamen und 3 Zeichen für die Datei-Erweiterung enthalten (Sonderzeichen dürfen nicht verwendet werden).

Gerät	Kennung	Zugriff	Beschreibung
Silicon-Disk	SD	rw	Dieser Treiber ermöglicht den Zugriff auf das Flash-PROM der Steuerung.
RAM-Disk	RD	rw	Dieser Treiber ermöglicht den Zugriff auf die so genannte RAM-Disk (batteriegepuffertes RAM).
Net-Disk	ND	rw	Dieser Treiber ermöglicht den Zugriff auf die Festplatte eines PC's. Das Gerät kann nur genutzt werden, wenn die Steuerung über eine Ethernet-Schnittstelle verfügt. Auf dem PC muss der Lenze ETC Data Server gestartet sein. Die SPS kann auf das Verzeichnis zugreifen, das in der Konfigurationsdatei des Servers eingetragen wurde. Um die Verbindung zwischen Steuerung und PC herzustellen, wird der Steuerung über die Funktion SetCurrentPath() die IP-Adresse des PC's bekannt gegeben. Dazu ist es notwendig, dass in den Netzwerkeinstellungen des PC's eine feste IP-Adresse eingetragen wird.
PC-Disk	PC	rw	Dieser Treiber ermöglicht den Zugriff auf die Festplatte eines PC's. Das Gerät kann nur verwendet werden, wenn auf dem PC eine Applikation läuft, die das Lenze Gateway benutzt (z. B. ETC-MMI). Mit Hilfe der Gateway Konfigurationssoftware kann ein Verzeichnis festgelegt werden, auf das die SPS dann zugreifen kann.
Programmspeicher (nur ETCxC)	PS	w	Dieser Treiber ermöglicht es DIN-Programme durch die SPS in den internen DIN-Programmspeicher zu schreiben. Hierbei ist zu beachten, dass der Dateiname gleichzeitig der Programmnummer entspricht und damit im DIN-Programm keine Programmnummer (mit %-Anweisung) enthalten sein darf. Beispiel: SysOpenFile('PS:12.din', O_WRONLY); öffnet ein DIN-Prog. mit der Programmnummer %12.
Drucker	PR	rw	Dieser Treiber ermöglicht es, über eine serielle Schnittstelle Zeichen auszugeben. Die serielle Schnittstelle wird über die Maschinenkonstanten MK_DRUCKER und MK_DRUCKER_V24MODE konfiguriert.
SD-Karte		rw	Nur ETC 2. Gerätegeneration: Dieser Treiber ermöglicht den Zugriff auf die SD-Karte einer
USB-Massenspeicher		rw	Nur ETC 2. Gerätegeneration: Dieser Treiber ermöglicht den Zugriff auf einen USB-Massenspeicher

6.8.3.14 Globale Konstanten zu File IO-Funktionen

Konstante	Wert	Bedeutung
DSK_EEPROM_NOT_READY_KDI	-1	Flash-PROM defekt
DSK_EEPROM_CHECKSUM_KDI	-2	Checksummenfehler in der Flash-Disk
DSK_EEPROM_DISK_FULL_KDI	-3	Flash-Disk ist voll, Datei kann nicht übernommen werden
DSK_ACCESS_DENIED_KDI	-4	Datei ist bereits geöffnet, Zugriff verweigert
DSK_TOO_MANY_FILES_KDI	-5	Zu viele gleichzeitig geöffnete Dateien, Zugriff verweigert
DSK_EEPROM_DIR_FULL_KDI	-6	Flash-Disk-Inhaltsverzeichnis ist voll, zu viele Dateien
DSK_FILE_NOT_EXIST_KDI	-7	Datei nicht vorhanden
DSK_INVALID_ACCESSMODE_KDI	-8	Nicht erlaubter Zugriffsmodus bei SysOpenFile
DSK_UNKNOWN_EEPROM_TYPE_KDI	-9	Unbekannten Flash-Typ gefunden
DSK_FILE_NOT_OPEN_KDI	-10	Datei nicht zum Lesen oder Schreiben geöffnet
DSK_EEPROM_OUT_OF_MEMORY_KDI	-11	Flash-Disk: Nicht genug Speicher zum Anlegen der Flash-Disk
DSK_LONGNAME_NOT_SUPPORTED_KDI	-12	Lange Dateinamen werden vom angesprochenen Gerät nicht unterstützt

Programmspeicher (1)

Konstante	Wert	Bedeutung
DSK_PS_BLOCK_TOO_LONG_KDI	-60	DIN-Satz länger als erlaubt
DSK_PS_INVALID_PRG_KDI	-61	Fehler bei der Programmanlage
DSK_PS_DIN_ERR_KDI	-62	Fehler im DIN-Programm
DSK_PS_NO_DATA_KDI	-63	Leeres DIN-Programm, keine Satzdaten
DSK_PS_CANNOT_OPEN_KDI	-64	Fehler beim Öffnen des Programms
DSK_PS_PRGNUM_NON_FOUND_KDI	-65	Keine Programmnummer gefunden
DSK_PS_NO_EOL_KDI	-66	Kein Abschlusszeichen am Satzende
DSK_PS_TRANSFER_ERR_KDI	-67	Block zu lang bei Online-Programmübertragung

RAM-Disk

Konstante	Wert	Bedeutung
XT3_TIMEOUT_WHILE_BOOTING_KDI	-90	Fehler beim Booten der XT3-Karte
DSK_SRAM_TIMEOUT_KDI	-91	XT3-Erweiterungskarte meldet sich nicht
DSK_SRAM_COM_ERROR_KDI	-92	Kommunikationsfehler mit XT3-Erweiterungskarte
DSK_SRAM_CHECKSUM_KDI	-93	Checksummenfehler in der RAM-Disk
DSK_SRAM_OUTOFMEMORY_KDI	-94	Nicht genug Heap-Speicher zum Anlegen der File-Buffer
DSK_SRAM_DIR_FULL_KDI	-95	Kein Platz mehr im RAM-Disk-Directory, zu viele Dateien
DSK_SRAM_DISK_FULL_KDI	-96	RAM-Disk ist voll
DSK_SRAM_VALIDATE_ERR_KDI	-97	Validieren der RAM-Disk: Datei war nicht zu retten und wurde gelöscht
DSK_SRAM_BATT_EMPTY_KDI	-98	Batterie ist leer, bitte wechseln
DSK_SRAM_RAM_ERROR_KDI	-99	RAM-Fehler im CMOS-RAM

Floppy-Disk

Konstante	Wert	Bedeutung
DSK_UNKNOWN_DEVICE_KDI	-110	Unbekannter Devicename
DSK_FLOPPY_V24_INUSE_KDI	-111	V24-Schnittstelle zum Diskettenlaufwerk ist bereits anderweitig belegt
DSK_FLOPPY_INUSE_KDI	-112	Diskettenlaufwerk ist nicht frei, Zugriff verweigert
DSK_FLOPPY_TIMEOUT_KDI	-113	Timeout beim Zugriff auf das Diskettenlaufwerk
DSK_FLOPPY_PROG_ERROR_KDI	-114	Fehler beim Zugriff auf das Diskettenlaufwerk
DSK_FLOPPY_TRANSMIT_ERROR_KDI	-115	Diskettenlaufwerk meldet Übertragungsfehler
DSK_FLOPPY_OVERRUN_ERROR_KDI	-116	Diskettenlaufwerk meldet V24-Overrun
DSK_FLOPPY_WRITE_PROTECT_KDI	-117	Diskette ist schreibgeschützt
DSK_FLOPPY_NO_DISK_KDI	-118	Keine Diskette eingelegt
DSK_FLOPPY_NOT_FORMATTED_KDI	-119	Schreib-/Lesefehler auf der Diskette
DSK_FUNCTION_NOT_SUPPORTED_KDI	-120	Die auszuführende Funktion wird vom Device-Treiber nicht unterstützt
DSK_INVALID_FILENAME_KDI	-121	Dateiname enthält unzulässige Buchstaben
DSK_ILLEGAL_FILENAME_KDI	-125	Der angegebene Dateiname ist nicht zulässig

Drucker

Konstante	Wert	Bedeutung
DSK_PRINTER_v24_INUSE_KDI	-122	V24-Schnittstelle zum Drucker ist bereits anderweitig belegt
DSK_PRINTER_INUSE_KDI	-123	Drucker wird bereits benutzt, Zugriff verweigert
DSK_PRINTER_TIMEOUT_KDI	-124	Drucker nicht angeschlossen oder nicht bereit

Filesystem

Konstante	Wert	Bedeutung
DSK_BLOCK_OUT_OF_RANGE	-200	Datei-Blocknummer nicht im erlaubten Bereich
DSK_DUPLICATE_BLOCK_KI	-201	Datei-Block doppelt belegt
DSK_ERR_UNKNOWN_KDI	-202	Unbekannter Dateisystemfehler
DSK_ERR_INVALIDDRIVE_KDI	-203	Ungültiger Laufwerksbuchstabe
DSK_ERR_NOTFORMATTED_KDI	-204	Medium ist nicht formatiert
DSK_ERR_INVALIDNAME_KDI	-206	Ungültiger Datei oder Verzeichnisname
DSK_ERR_DUPLICATED_KDI	-207	Datei oder Verzeichnis existiert bereits
DSK_ERR_NOMOREENTRY_KDI	-208	Kein freier Verzeichniseintrag mehr verfügbar
DSK_ERR_NOTOPEN_KDI	-209	Datei nicht geöffnet
DSK_ERR_EOF_KDI	-210	Unerwartetes Dateiende
DSK_ERR_NOTUSEABLE_KDI	-211	Falsches Argument bei Seek-Funktion
DSK_ERR_LOCKED_KDI	-212	Datei ist in Benutzung, Zugriff verweigert
DSK_ERR_ACCESSDENIED_KDI	-213	Datei ist schreibgeschützt, Zugriff verweigert
DSK_ERR_NOTEMPTY_KDI	-214	Verzeichnis ist nicht leer und kann nicht gelöscht werden
DSK_ERR_INITFUNC_KDI	-215	Initialisierungsfehler
DSK_ERR_CARDREMOVED_KDI	-216	Medium wurde entfernt
DSK_ERR_INVALIDSECTOR_KDI	-218	Dateisystemfehler, unzulässige Sektornummer
DSK_ERR_READ_KDI	-219	Sektor konnte nicht gelesen werden
DSK_ERR_WRITE_KDI	-220	Sektor konnte nicht geschrieben werden
DSK_ERR_INVALIDMEDIA_KDI	-221	Nicht unterstütztes Mediumformat
DSK_ERR_NOMORETASK_KDI	-222	Kein freier Eintrag mehr in Task-Tabelle
DSK_ERR_WRITEPROTECT_KDI	-223	Medium ist schreibgeschützt
DSK_ERR_INVFATTYPE_KDI	-224	Nicht unterstütztes FAT-Format
DSK_ERR_MEDIATOOSMALL_KDI	-225	Medium zu klein für File-Allocation-Table
DSK_ERR_MEDIATOOLARGE_KDI	-226	Medium zu groß für File-Allocation-Table
DSK_ERR_NOTSUPPSECTORSIZE_KDI	-227	Nicht unterstütztes Sektorformat
DSK_ERR_DRVALREADYMNT_KDI	-228	Laufwerk schon angemeldet
DSK_ERR_TOOLONGNAME_KDI	-229	Zu langer Dateiname
DSK_ERR_ALLOCATION_KDI	-230	Speichermangel
DSK_ERR_CHAINERROR_KDI	-241	File allocation chain error
DSK_ERR_PATHNOTFOUND_KDI	-242	Pfad nicht gefunden
DSK_ERR_DRIVEERROR_KDI	-243	Physikalischer Laufwerksfehler
DSK_ERR_NOPERMISSION_KDI	-244	Zugriff verweigert
DSK_ERR_INVALIDMODE_KDI	-245	Ungültiger Zugriffsmodus
DSK_ERR_NOFREEHANDLE_KDI	-246	Kein freies Filehandle verfügbar
DSK_ERR_DISKFULL_KDI	-247	Disk ist voll

Programmspeicher (2)

Konstante	Wert	Bedeutung
DSK_PCDISK_IOERR_KDI	-301	Unbekannter IO-Fehler
DSK_DLL_TIMEOUT_KDI	-302	Timeout bei IO-Operation
DSK_DLL_OUT_OF_MEMORY_KDI	-303	PSpeichermangel bei IO-Operation

Net-Disk

Konstante	Wert	Bedeutung
DSK_NET_NOT_CONFIGURED_KDI	-331	Laufwerk nicht initialisiert
DSK_NET_COM_ERR_KDI	-332	Kommunikationsfehler, falscher Befehlscode
DSK_NET_ANSWER_ERR_KDI	-333	Kommunikationsfehler, falsche Quittung
DSK_NET_TIMEOUT_ERR_KDI	-334	Timeout beim Zugriff auf das Laufwerk
DSK_NET_CANNOT_OPFEN_FILE_KDI	-335	Server meldet Fehler bei FileOpen
DSK_NET_CANNOT_CLOSE_FILE_KDI	-336	Server meldet Fehler bei FileClose
DSK_NET_HANDLE_ERROR_KDI_KDI	-337	Server meldet Handle-Fehler
DSK_NET_READ_ERROR_KDI_KDI	-338	Server meldet Fehler bei FileRead
DSK_NET_WRITE_ERROR_KDI_KDI	-339	Server meldet Fehler bei FileWrite
DSK_NET_FILE_NOT_EXISTS_KDI	-340	Datei nicht vorhanden
DSK_NET_DIR_NOT_CREATED_KDI	-341	Server meldet Fehler bei CreateDir
DSK_NET_DIR_NOT_DELETED_KDI	-342	Server meldet Fehler bei DeleteDir
DSK_NET_FILE_NOT_CREATED_KDI	-343	Server meldet Fehler bei CreateFile
DSK_NET_FILE_NOT_DELETED_KDI	-344	Server meldet Fehler bei DeleteFile
DSK_NET_BUSY_KDI	-345	Laufwerk ist nicht frei, Zugriff verweigert

CMOS-Daten

Konstante	Wert	Bedeutung
DSK_CMS_INIT_KDI	-401	CMOS-RAM ungültig: Remanente Daten wurden gelöscht
DSK_CMS_BATT_EMPTY_KDI	-402	CMOS-RAM: Batterie ist leer, bitte wechseln
DSK_CMS_OLD_BOOTLOADER_KDI	-403	CMOS-RAM: alte Version des Bootloaders, bitte updaten
O_RDONLY	0	
O_WRONLY	1	

6.8.4 Speicherzugriffsfunktionen

6.8.4.1 DEFDATATYPES

Deklaration

```
FUNCTION DEFDATATYPES: BYTE
VAR_INPUT
    iBereich      : INT;
    sDescriptor    : STRING(255);
END_VAR
```

Beschreibung

Die Funktion definiert die Datentypen im DB2 und damit das notwendige Byte-Swapping beim Datenaustausch zwischen NCR und MMI.

Zurzeit werden zwei Bereiche unterschieden:

Bereich 1 von Wort 0 ... 127 enthält Daten von SPS an MMI, Bereich 2 von Wort 128 ... 191 enthält Daten von MMI an SPS. Zur Definition der Datentypen wird ein Descriptor-String verwendet. Der Descriptorstring hat folgendes Format:

<Anzahl><Typ><Anzahl><Typ>...

Anzahl ist eine Dezimalzahl und gibt die Menge der Daten vom nachfolgenden Typ an.

Typ ist ein Kleinbuchstabe, der die Größe des zugehörigen Datentyps wie folgt beschreibt:

Typ	Codesys-Datentyp	Größe
b	BYTE	8 Bit
w	BOOL, WORD, INT	16 Bit
d	DWORD, DINT, REAL	32 Bit
l	LREAL	64 Bit

Die Funktion liefert den Wert 1, wenn die Funktion erfolgreich ausgeführt wurde, ansonsten den Wert 0.

Beispiel

Am Anfang vom DB2 im Bereich 1 von %MW2.0 bis %MW2.16 liegen 10 BYTE, 2 WORD, 3 DINT und 1 LREAL. Im Bereich 2 von %MW2.128 bis %MW2.158 liegen 5 WORD, 8 DWORD und 5 WORD in dieser Reihenfolge. Die Definition erfolgt folgendermaßen:

```
DEFDATATYPES(1,'10b2w3d1l')
DEFDATATYPES(2,'5w8d10w')
```

Wenn eine Nachrichtenschnittstelle im DB2 zur Kommunikation zwischen SPS und HMI verwendet wird und in den Nachrichten andere Datentypen als WORD verwendet werden, sind weitergehende Anpassungen notwendig. Die ETC kennt zu diesem Zweck typengerechte Funktionen:

GET_BYTE	PUT_BYTE
GET_WORD	PUT_WORD
GET_INT	PUT_INT
GET_DWORD	PUT_DWORD
GET_DINT	PUT_DINT
GET_REAL	PUT_REAL
GET_LREAL	PUT_LREAL

Die Funktionen GET_... lesen den entsprechenden Datentyp von der angegebenen Adresse und führt das notwendige Byte-Swapping für das HMI durch. Die Funktionen PUT_... schreiben den entsprechenden Datentyp gewapped an die angegebene Adresse.

Voraussetzung dafür, dass die Daten dann richtig beim HMI ankommen, wenn die obigen Funktionen verwendet werden, ist die Definition der Nutzdaten der Nachricht als Byte mit DEFDATATYPES. Für das Standard-HMI sieht das dann wie folgt aus:

```
DEFDATATYPES(1,'96w4w52b2w'); (* 96 WORD, 4 WORD, 52 BYTE, 2 WORD *)
DEFDATATYPES(2,'32w4w52b2w'); (* 32 WORD, 4 WORD, 52 BYTE, 2 WORD *)
```

Die Definition ergibt sich aus der Festlegung der Daten im DB2, die zwischen ETC-MMI und SPS ausgetauscht werden:

Datenwort	Name	Richtung
000.00 – 015.15	Zustände (256 Bits)	PLC → HMI
016.00 – 079.15	Anzeigen (64 Datenwort)	PLC → HMI
080.00 – 095.15	Hinweise/statische Fehler (256 Bits)	PLC → HMI
096.00 – 096.15	Nachrichtenpuffer: Quittungszähler	PLC → HMI
097.00 – 097.15	Nachrichtenpuffer: Zähler 1	PLC → HMI
098.00 – 125.15	Nachrichtenpuffer: Nutzdaten (28 Datenworte)	PLC → HMI
126.00 – 126.15	Nachrichtenpuffer: Zähler 2	PLC → HMI
127.00 – 127.15	Überwachung SPS läuft	PLC → HMI
128.00 – 143.15	Zustände (256 Bits)	PLC ← HMI
144.00 – 159.15	Tasten (256 Bits)	PLC ← HMI
160.00 – 160.15	Nachrichtenpuffer: Quittungszähler	PLC ← HMI
161.00 – 161.15	Nachrichtenpuffer: Zähler 1	PLC ← HMI
162.00 – 189.15	Nachrichtenpuffer: Nutzdaten (28 Datenwort)	PLC ← HMI
190.00 – 190.15	Nachrichtenpuffer: Zähler 2	PLC ← HMI
191.00 – 191.15	Überwachung ETC-MMI läuft	PLC ← HMI

6.8.4.2 GET_BYTE, GET_WORD, GET_DWORD, GET_INT, GET_DINT, GET_REAL, GET_LREAL

Deklaration

```
FUNCTION GET_TYPE : BYTE
VAR_INPUT
    pAddress      : DINT;          (* Speicheradresse *)
END_VAR
```

Beschreibung

Die Funktionen GET_TYPE lesen den entsprechenden Datentyp TYPE von der angegebenen Adresse und führen das notwendige Byte-Swapping durch (siehe auch DEFDATATYPES).

6.8.4.3 MEMCOMP

Deklaration

```
FUNCTION MEMCOMP: INT
VAR_INPUT
    pMem1      : DINT;          (* Speicheradresse *)
    pMem2      : DINT;          (* Speicheradresse *)
    diSize     : DINT;          (* Anzahl Bytes *)
END_VAR
```

Beschreibung

Es werden die Speicherbereiche Mem1 und Mem2 byteweise verglichen.

Returnwert

0: wenn der Inhalt gleich ist

1: wenn der Inhalt von Mem1 > Mem2 ist

-1: wenn der Inhalt von Mem1 < Mem2 ist

6.8.4.4 MEMCOPY

Deklaration

```
FUNCTION MEMCOPY: BOOL
VAR_INPUT
    ziel_p      : DINT;          (* Speicheradresse *)
    quelle_p    : DINT;          (* Speicheradresse *)
    anzahl_di   : DINT;          (* Anzahl der Bytes *)
END_VAR
```

Beschreibung

Es werden *anzahl_di* Bytes ab Speicheradresse *quelle_p* gelesen und ab Speicheradresse *ziel_p* eingetragen.

Der Returnwert der Funktion hat keine Bedeutung.

Beispiel

```
ret_bit      : BOOL;
feld1_ab     : ARRAY[0..10] OF BYTE;
feld2_ab     : ARRAY[0..5] OF BYTE;

MEMCOPY(ADR(feld1_ab[4]), ADR(feld2_ab[2]), 4);
MEMCOPY(ADR(feld2_ab[0]), ADR(feld1_ab[2]), SIZEOF(feld2_ab));
```

6.8.4.5 MEMSET

Deklaration

```

FUNCTION MEMSET: BOOL
VAR_INPUT
    pMem      : DINT;          (* Speicheradresse *)
    bValue     : DINT;          (* Wert *)
    dwSize     : DINT;          (* Anzahl der Bytes *)
END_VAR

```

Beschreibung

Es werden *dwSize* Bytes ab Speicheradresse *pMem* mit Wert *bValue* geschrieben.

Beispiel

```

feld_ab      : ARRAY[0..10] OF BYTE;
MEMSET( ADR(feld_ab[0]), 0, SIZEOF(feld_ab) );

```

6.8.4.6 OVESWAPPED

Deklaration

```

FUNCTION MOVESWAPPED: DINT
VAR_INPUT
    pDestination : DINT;          (* Speicheradresse *)
    pSource       : DINT;          (* Speicheradresse *)
    sDescstring   : STRING(255);  (* Datenbeschreibungsstring *)
END_VAR

```

Beschreibung

Die Funktionen kopiert Daten von der Speicheradresse *pDestination* zu Speicheradresse *pSource*. Die Daten werden entsprechend des angegebenen Descriptor-Strings gewappt kopiert.

Der Descriptorstring hat folgendes Format:

<Anzahl><Typ><Anzahl><Typ>...

Anzahl ist eine Dezimalzahl und gibt die Menge der Daten vom nachfolgenden Typ an.

Typ ist ein Kleinbuchstabe, der die Größe des zugehörigen Datentyps wie folgt beschreibt:

Typ	Codesys-Datentyp	Größe
b	BYTE	8 Bit
w	BOOL, WORD, INT	16 Bit
d	DWORD, DINT, REAL	32 Bit
l	LREAL	64 Bit

Die Funktion liefert als Returnwert die Anzahl der kopierten Bytes.

Beispiel

```

TYPE DATA_TR :
STRUCT
    var1          : DINT;
    var2          : DINT;
END_STRUCT
END_TYPE

byte_di          : DINT;
descstring_s     : STRING(10) := '2d';
data_st          : DATA_TR := (var1:=16#FF, var2:=16#123456);
buf_ab           : ARRAY[0..20] OF BYTE;

byte_di := MOVESWAPPED(ADR(buf_ab), ADR(data_st), descstring_s);

```

Ergebnis:

```
byte_di = 8, buf_ab[0] = 16#FF, buf_ab[4] = 16#56, buf_ab[5] = 34, buf_ab[6] := 16#12
```

Alle anderen Elemente des Arrays sind gleich 0.

6.8.4.7 PUT_BYTE, PUT_WORD, PUT_DWORD, PUT_INT, PUT_DINT, PUT_REAL, PUT_LREAL

Deklaration

```

FUNCTION PUT_TYPE: BYTE
VAR_INPUT
    pAddress      : DINT;          * Speicheradresse *)
    Value         : TYPE;         (* zu schreibender Wert *)
END_VAR

```

Beschreibung

Die Funktionen PUT_TYPE schreiben den entsprechenden Datentyp TYPE gewappt an die angegebene Adresse (siehe auch DEFDATATYPES).

6.8.5 CANopen Funktionen

**Hinweis!**

Bei den CANopen Funktion sind die Identifier (Identifier = Basis-Identifier + Knotenadresse) nach der CANopen Spezifikation DS301 einzuhalten.

Für Prozessdatenobjekte gelten die im Abschnitt "PDO-Mapping" (222) beschriebenen Identifier, für Servicedatenobjekte TX-SDO = 1408 + NodeId sowie für RX-SDO = 1536 + NodeId.

Identifier außerhalb dieser Bereiche werden von der Steuerung ignoriert.

6.8.5.1 CopDefineDS403

Deklaration

FUNCTION **CopDefineDS403**: BOOL

VAR_INPUT (* Def. die Global-Database für ein CanOpen-Bedienteil nach DS403 *)

 NodeId : BYTE;

 DataBase : DINT;

 DataSize : WORD;

 Datatypes : STRING(255);

 ReadDataObject : WORD;

 WriteDataObject : WORD;

END_VAR

Parameter

NodeId	Knotennummer des Bedienteils
DataBase	Adresse der Global-Database mit den Ein-/Ausgabedaten des Bedienteils
DataSize	Größe der Global-Database in Byte, max. 1020 Byte
Datatypes	Descriptor-String zur Definition der Datentypen in der Global-Database
ReadDataObject	Objektnummer zum Lesen der "Local-Database"
WriteDataObject	Objektnummer zum Beschreiben der "Local-Database"

Beschreibung

Diese Funktion gibt dem Laufzeitsystem einen Datenbereich der SPS zum Datenaustausch mit einem CANopen-Bedienteil bekannt. Dieser Datenbereich wird in der SPS am einfachsten als Datenstruktur oder als Array angelegt und mit den anzuzeigenden Daten initialisiert. Nach Aufruf der Funktion wird diese Global-Database einmal komplett in die Local-Database des entsprechenden Knotens übertragen. Anschließend werden nur noch die Änderungen der Anzeigedaten von der Steuerung übertragen.

ReadDataObject und WriteDataObject sind die Objektnummern des Objektverzeichnisses des CANopen-Displays, die vom Laufzeitsystem zum Lesen und Schreiben der Local-Database verwendet werden müssen.

Der Descriptorstring hat folgendes Format:

<Anzahl><Typ><Anzahl><Typ>...

Anzahl ist eine Dezimalzahl und gibt die Menge der Daten vom nachfolgenden Typ an.

Typ ist ein Kleinbuchstabe, der die Größe des zugehörigen Datentyps wie folgt beschreibt:

Typ	Codesys-Datentyp	Größe
b	BYTE	8 Bit
w	BOOL, WORD, INT	16 Bit
d	DWORD, DINT, REAL	32 Bit

Der Returnwert ist TRUE, wenn die Funktion erfolgreich ausgeführt werden konnte. Andernfalls konnte die Funktion, z. B. wegen Speichermangel, nicht ausgeführt werden.



Hinweis!

Die Database sollte nur so groß sein, wie die tatsächlich verwendeten Daten es erfordern.

Die SPS sollte immer nur die Daten in der Global-Database aktualisieren, die auch aktuell im Display zur Anzeige kommen, um die Buslast auf dem CAN-Bus nicht unnötig zu erhöhen. Zyklische Änderungen in jedem SPS-Takt sollten generell vermieden werden.

Beispiel

```

TYPE DATABASE_TR :
STRUCT
    dummy_dw      : DWORD;
    pad1_dw       : DWORD;
    pad2_b        : BYTE;
    pad3_b        : BYTE;
    pad4_w        : WORD;
END_STRUCT
END_TYPE
data_st          : DATABASE_TR;
COPDEFINEDS403( 127, ADR(data_st), SIZEOF(data_st), '2d2b1w', 16#6700, 16#8100 );

```

Die Funktion definiert für ein CANopen-Bedienteil mit der Knotennummer 127, das entsprechend in der Steuerungskonfiguration von CoDeSys eingefügt wurde, eine Global-Database mit den durch den Descriptor-String definierten Elementen.

6.8.5.2 CopXDefineDS403 (nur ETCxM)

Deklaration

```

FUNCTION CopXDefineDS403: BOOL
VAR_INPUT
    CanNum_b      : BYTE
    NodeID_b      : BYTE
    DataBase_p    : DINT;
    Len_w         : WORD;
    Datatypes_s   : STRING(255);
    ReadDataObject : WORD;
    WriteDataObject : WORD;
END_VAR
  
```

(* Definiert die Global-Database für ein CANopen-Bedienteil nach DS403 für 1/2 CAN-Bus *)

Parameter

CanNum_b:	Nummer des CAN-Bus (1 oder 2)
NodeID_b:	Knotennummer des Bedienteils
DataBase_p:	Adresse der Global-Database mit den Ein-/Ausgabedaten des Bedienteils
Len_w:	Größe der Global-Database in Byte, max. 1020 Byte
Datatypes_s:	Descriptor-String zur Definition der Datentypen in der Global-Database
ReadDataObject:	Objektnummer zum Lesen der "Local-Database"
WriteDataObject:	Objektnummer zum Beschreiben der "Local-Database"

Beschreibung

Die Funktionalität dieser Funktion entspricht der Funktion CopDefineDS403(). Sie enthält einen zusätzlichen Parameter *CanNum_b*, der die Nummer des betroffenen CAN-Bus bestimmt.

6.8.5.3 CopGetNodeState (nur ETCxM)

Deklaration

```

FUNCTION CopGetNodeState: BOOL
VAR_INPUT
    NodeID_b      : BYTE;
END_VAR
  
```

* CANopen-ID des Moduls *)

Parameter

NodeID_b:	Knotennummer des CANopen-Knotens
-----------	----------------------------------

Beschreibung

Der Returnwert TRUE signalisiert, dass das entsprechende CANopen-Modul am 1. CAN-Bus vorhanden und im Zustand "Operational" ist. Ansonsten wird FALSE zurückgeliefert. Diese Erkennung funktioniert nur, wenn das No-guarding für das entsprechende CANopen-Gerät aktiviert ist.

6.8.5.4 CopReadObject

Deklaration

```
FUNCTION CopReadObject: BOOL
(* Auslesen von Objekten des Objektverzeichnisses von CANopen-Geräten an CAN1 *)
VAR_INPUT
    NodeID      : BYTE;
    ObjectNum    : WORD;
    SubIndex     : BYTE;
    DataType     : WORD;
    Buffer        : DINT;
    BufSize      : WORD;
    Status       : POINTER TO BYTE;
END_VAR
```

Parameter

NodeID	Knotennummer des CANopen-Knotens
ObjectNum	Nummer des Objekts, welches gelesen werden soll
SubIndex	Index des Unterobjekts, welches gelesen werden soll
DataType	Datentyp nach CiA DS301
Buffer	Adresse des Puffers, in dem die gelesenen Daten abgelegt werden
BufSize	Größe des Puffers in Byte
Status	Adresse einer Variablen zur Ablage des Transferstatus

Beschreibung

Diese Funktion ermöglicht es der SPS, einzelne Objekte von CANopen-Geräten auszulesen. Der Lesevorgang erfolgt dabei im Hintergrund während die SPS weiter läuft. Die Transferqueue kann bis zu 15 Schreib-/Leseaufträge puffern.

Durch Abfrage des Transferstatus kann die SPS überprüfen, ob die Übertragung noch läuft, beendet ist, oder ein Fehler bei der Übertragung aufgetreten ist. Sollen mehrere Transferaufträge parallel ausgeführt werden, so muss für jeden Auftrag eine eigene Statusvariable definiert werden.

Der Returnwert ist TRUE, wenn der Auftrag zum Auslesen in die Transferqueue übertragen werden konnte, andernfalls ist die Transferqueue bereits voll.

Status	Bedeutung
0	Inaktiv
1	Auftrag in der Transferqueue
2	Transfer aktiv
3	Transfer erfolgreich beendet
4	Transfer abgebrochen
5	Timeout

Beispiel

```
status_b
devicetype_dw
```

```
CopReadObject(1, 16#1000, 0, COP_UNSIGNED32_KW, ADR(devicetype_dw),
SIZEOF(devicetype_dw), ADR(status_b));
```

6.8.5.5 CopXReadObject (ETCxM an CAN2)

Deklaration

FUNCTION **CopXReadObject**: BOOL (* Auslesen von Objekten des Objektverzeichnis-
ses von CANopen-Geräten an CAN2 *)

VAR_INPUT

CanNum_b : BYTE;
 NodeID_b : BYTE;
 ObjectNum_w : WORD;
 SubIndex_b : BYTE;
 DataType_w : WORD;
 Buffer_p : DINT;
 BufSize_w : WORD;
 Status_pb : POINTER TO BYTE;

END_VAR

Parameter

CanNum_b:	Nummer des CAN-Bus (1 oder 2)
NodeID_b:	Knotennummer des CANopen-Knotens
ObjectNum_w:	Nummer des Objekts, das gelesen werden soll
SubIndex_b:	Index des Unterobjekts, das gelesen werden soll
DataType_w:	Datentyp nach CiA DS301
Buffer_p:	Adresse des Puffers, in dem die gelesenen Daten abgelegt werden
BufSize_w:	Größe des Puffers in Byte
Status_pb:	Adresse einer Variablen zur Ablage des Transferstatus

Beschreibung

Die Funktionalität dieser Funktion entspricht voll der Funktion CopReadObject(). Sie enthält einen zusätzlichen Parameter *CanNum_b*, der die Nummer des betroffenen CAN-Bus bestimmt.

6.8.5.6 CopWriteObject

Deklaration

FUNCTION **CopWriteObject**: BOOL
 (* Beschreiben von Objekten im Objektverzeichnis von CANopen-Geräten an CAN1 *)

VAR_INPUT

NodeID : BYTE;
 ObjectNum : WORD;
 SubIndex : BYTE;
 DataType : WORD;
 Buffer : DINT;
 BufSize : WORD;
 Status : POINTER TO BYTE;

END_VAR

Parameter

NodeID	Knotennummer des CANopen-Gerätes
ObjectNum	Nummer des Objekts, welches geschrieben werden soll
SubIndex	Index des Unterobjekts, welches geschrieben werden soll
DataType	Datentyp nach CiA DS301
Buffer	Adresse des Puffers, in dem die zu schreibenden Daten abgelegt sind
BufSize	Anzahl zu schreibender Bytes
Status	Adresse einer Variablen zur Ablage des Transferstatus

Beschreibung

Diese Funktion ermöglicht es der SPS, einzelne Objekte von CANopen-Geräten zu beschreiben. Der Schreibvorgang erfolgt dabei im Hintergrund während die SPS weiter läuft. Die Transferqueue kann bis zu 15 Schreib-/Leseaufträge puffern.

Durch Abfrage des Transferstatus kann die SPS überprüfen, ob die Übertragung noch läuft, beendet ist oder ein Fehler bei der Übertragung aufgetreten ist. Sollen mehrere Transferaufträge parallel ausgeführt werden, so muss für jeden Auftrag eine eigene Statusvariable definiert werden.

Die Funktion liefert TRUE, wenn der Auftrag zum Schreiben in die Transferqueue übertragen wurde, andernfalls ist die Transferqueue bereits voll.

Status	Bedeutung
0	Inaktiv
1	Auftrag in der Transferqueue
2	Transfer aktiv
3	Transfer erfolgreich beendet
4	Transfer abgebrochen
5	Timeout

Beispiel

```
status_b      : BYTE;
wert_di       : DINT := 1000;
```

```
CopWriteObject(5, 16#607A, 0, COP_INTEGER32_KW, ADR(wert_di), sizeof(wert_di),
ADR(status_b));
```

Eintragen der Zielposition für einen DS402-Antrieb.

6.8.5.7 CopXWriteObject (ETCxM auf CAN2)

Deklaration

FUNCTION CopXWriteObject: BOOL

(* Beschreiben von Objekten im Objektverzeichnis von CANopen-Geräten an CAN2 *)

VAR_INPUT

CanNum_b : BYTE;
 NodeID_b : BYTE;
 ObjectNum_w : WORD;
 SubIndex_b : BYTE;
 DataType_w : WORD;
 Buffer_p : DINT;
 BufSize_w : WORD;
 Status_pb : POINTER TO BYTE;

END_VAR

Parameter

CanNum_b:	Nummer des CAN-Bus (1 oder 2)
NodeID_b:	Knotennummer des CANopen-Gerätes
ObjectNum_w:	Nummer des Objekts, das geschrieben werden soll
SubIndex_b:	Index des Unterobjekts, das geschrieben werden soll
DataType_w:	Datentyp nach CiA DS301
Buffer_p:	Adresse des Puffers, in dem die zu schreibenden Daten abgelegt sind
BufSize_w:	Anzahl zu schreibender Bytes
Status_pb:	Adresse einer Variablen zur Ablage des Transferstatus

Beschreibung

Die Funktionalität dieser Funktion entspricht der Funktion CopWriteObject. Sie enthält einen zusätzlichen Parameter *CanNum_b*, der die Nummer des betroffenen CAN-Bus bestimmt.

6.8.5.8 CopXSetOperational (nur ETCxM)

Deklaration

FUNCTION CopXSetOperational: BOOL

(* Setzen des CANopen-Teilnehmers in Zustand Operational *)

VAR_INPUT

CanNum_b : BYTE;
 NodeID_b : BYTE;
 Status_pb : POINTER TO BYTE;

END_VAR

Parameter

CanNum_b:	Nummer des CAN-Bus (1 oder 2)
NodeID_b:	Knotennummer des CANopen-Gerätes
Status_pb:	Adresse einer Variablen zur Ablage des Transferstatus

Beschreibung

Die Funktion ermöglicht es der SPS, einzelne CANopen-Teilnehmer in den Zustand Operational zu setzen. Voraussetzung dafür ist, dass der Teilnehmer in der CoDeSys-Steuerungskonfiguration eingetragen wurde. Das Versenden der Telegramme erfolgt dabei im Hintergrund, während die SPS weiterläuft.

Durch Abfrage des Transferstatus kann die SPS überprüfen, ob die Nachricht gesendet wurde oder ein Fehler aufgetreten ist. Sollen mehrere Transferaufträge parallel ausgeführt werden, so muss für jeden Auftrag eine eigene Statusvariable definiert werden.

Die Funktion liefert TRUE, wenn der Auftrag zum Schreiben in die Transferqueue übertragen wurde. Anderenfalls ist die Transferqueue bereits voll.

Status	Bedeutung
0	Inaktiv
1	Auftrag in der Transferqueue
2	Transfer aktiv
3	Transfer erfolgreich beendet
4	Transfer abgebrochen

Beispiel

```
status_b      BYTE;
wert_di       : DINT := 1000;
```

```
CopXSetOperational(1,5, ADR(status_b));
```

Der Knoten mit der Adresse 5 am 1. CAN-Bus wird in den Zustand Operational gesetzt.

6.8.5.9 CopXSetPreoperational (nur ETCxM)**Deklaration**

```
FUNCTION CopXSetPreoperational: BOOL (* Setzen des CANopen-Teilnehmers in Zustand Preoperational *)
```

```
VAR_INPUT
```

```
CanNum_b      : BYTE;
NodeID_b      : BYTE;
Status_pb     : POINTER TO BYTE;
```

```
END_VAR
```

Parameter

```
CanNum_b:      Nummer des CAN-Bus (1 oder 2)
NodeID_b:      Knotennummer des CANopen-Gerätes
Status_pb:     Adresse einer Variablen zur Ablage des Transferstatus
```

Beschreibung

Diese Funktion ermöglicht es der SPS, einzelne CANopen-Teilnehmer in den Zustand Preoperational zu setzen. Voraussetzung hierfür ist, dass der Teilnehmer in der CoDeSys-Steuerungskonfiguration eingetragen wurde. Das Versenden des Telegramms erfolgt dabei im Hintergrund, während die SPS weiterläuft.

Durch Abfrage des Transferstatus kann die SPS überprüfen, ob die Nachricht gesendet wurde oder ein Fehler aufgetreten ist. Sollen mehrere Transferaufträge parallel ausgeführt werden, so muss für jeden Auftrag eine eigene Statusvariable definiert werden.

Die Funktion liefert TRUE, wenn der Auftrag zum Schreiben in die Transferqueue übertragen wurde. Anderenfalls ist die Transferqueue bereits voll.

Status	Bedeutung
0	Inaktiv
1	Auftrag in der Transferqueue
2	Transfer aktiv
3	Transfer erfolgreich beendet
4	Transfer abgebrochen

Beispiel

```
status_b      : BYTE;
wert_di       : DINT := 1000;
```

```
CopXSetPreoperational(1,5, ADR(status_b));
```

Der Knoten mit der Adresse 5 am 1. CAN-Bus wird in den Zustand Preoperational gesetzt.

6.8.5.10 CopXEnableSync (nur ETCxM)

Deklaration

```
FUNCTION CopXEnableSync: DINT          (* Freigabe der Sync-Nachricht *)
VAR_INPUT
    CanNum_uc      : BYTE;
END_VAR
```

Parameter

```
CanNum_uc      Nummer des CAN-Bus (1 oder 2)
```

Beschreibung

Mit Aufruf dieser Funktion wird das Versenden der CANopen-Sync-Nachricht durch die SPS aktiviert. Die Sync-Nachrichten werden mit der Zykluszeit der aufrufenden PLC-Task versendet.

Die Funktion liefert folgende Werte zurück:

Returnwert	Bedeutung
0	Funktion erfolgreich ausgeführt
-1	CAN-Task ist nicht bereit
-3	Fehlerhafte CAN-Nummer
-4	Die Funktion ist schon frei gegeben
-5	Sendepuffer des CAN-Controllers ist schon belegt

6.8.5.11 CopXDisableSync (nur ETCxM)

Deklaration

```
FUNCTION CopXDisableSync : DINT      (* Sperren des Versenden der Sync-Nachrichten *)
VAR_INPUT
    CanNum_uc      : BYTE;
END_VAR
```

Parameter

CanNum_uc	Nummer des CAN-Bus (1 oder 2)
-----------	-------------------------------

Beschreibung

Mit Aufruf dieser Funktion wird das Versenden der CANopen-Sync-Nachricht durch die SPS deaktiviert. Das Sperren der Sync-Nachrichten kann nur von der PLC-Task erfolgen, die das Versenden aktiviert hat.

Die Funktion liefert folgende Werte zurück:

Returnwert	Bedeutung
0	Funktion erfolgreich ausgeführt
-1	CAN-Task ist nicht bereit
-3	Fehlerhafte CAN-Nummer
-4	Die Funktion ist nicht aktiv
-5	Das Versenden der Sync-Nachrichten wurde von einer anderen Task frei gegeben.
-6	Der Sendepuffer des CAN-Controllers ist mit einer anderen COB-ID belegt.

6.8.5.12 Globale Konstanten zu CANopen Funktionen

Diese Konstanten sind als Übergabeparameter "DataType" für die Funktionen CopReadObject() und CopWriteObject() gedacht. Die Werte der Konstanten entsprechen CiA DS301 "Application Layer and Communication Profile".

Konstante	Wert	Bedeutung
COP_BOOLEAN_KW	16#1	
COP_INTEGER8_KW	16#2	
COP_INTEGER16_KW	16#3	
COP_INTEGER32_KW	16#4	
COP_UNSIGNED8_KW	16#5	
COP_UNSIGNED16_KW	16#6	
COP_UNSIGNED32_KW	16#7	
COP_REAL32_KW	16#8	
COP_VISIBLE_STRING_KW	16#9	
COP_OCTET_STRING_KW	16#A	
COP_UNICODE_STRING_KW	16#B	
COP_BIT_STRING_KW	16#E	
COP_REAL64_KW	16#11	

Definitionen für Systemvariable db0_can_errorstatus_adw

CAN_TXOVERRUN_KDW	16#20000	Sendequeue ist voll
CAN_OVERRUN_KDW	16#10000	Overrun beim Empfang
CAN_BITERR_KDW	16#C000	Transmit-Bit-Error
CAN_ACKERR_KDW	16#2000	Acknowledge-Error
CAN_CRCERR_KDW	16#1000	CRC-Error
CAN_FORMERR_KDW	16#0800	Message-Format-Error
CAN_STUFERR_KDW	16#0200	Transmit-Error-Counter >= 96
CAN_RXWARN_KDW	16#0100	Recieve-Error-Counter >= 96
CAN_IDLE_KDW	16#0040	kein Fehler
CAN_FCS_KDW	16#0030	Fault-Confinement-State
CAN_BOFFINT_KDW	16#0004	Bus-Off
CAN_ERRINT_KDW	16#0002	Error-Interrupt
CAN_WAKEINT_KDW	16#0001	kein Fehler

6.8.6 CAN-Funktionen (nur ETCxM)

6.8.6.1 DefineCanMsg (nur ETCxM)

Deklaration

FUNCTION DefineCanMsg: DINT

VAR_INPUT

CanNum_w	: WORD;	(* Nummer der CAN-Schnittstelle - 1 oder 2 *)
WriteCobId_w	: WORD;	(* CobId: ETCxM → CAN-Teilnehmer *)
ReadCobId_w	: WORD;	(* CobId: CAN-Teilnehmer → ETCxM *)
QueueSize_w	: WORD;	(* Anzahl Nachrichten, die gepuffert werden *)

END_VAR

Beschreibung

Anlegen und Initialisieren einer Struktur für die Verwaltung von VAN-Transfers. Damit ist es möglich, beliebige CAN-Nachrichten zu empfangen oder zu versenden. Beim Aufruf der Funktion wird überprüft, ob die angegebenen CobId noch nicht in Benutzung sind. Dabei kann die CobId auch in der Steuerungskonfiguration von CoDeSys bei einem CANopen-Modul eingetragen sein. ReadCobId_w bzw. WriteCobId_w gleich 0 bedeutet, dass vorerst keine CobId benötigt wird.

Die Funktion liefert ein Handle auf die Verwaltungsstruktur zurück. Wird 0 zurückgeliefert, konnte keine Verwaltungsstruktur mit den angegebenen Parametern angelegt werden.

Beispiel

```
handle_pr      : DINT;

handle_pr = DefineCanMsg(1, 1014, 1114, 32);
```

6.8.6.2 AddCobIdCanMsg (nur ETCxM)

Deklaration

```
FUNCTION AddCobIdCanMsg: BOOL
VAR_INPUT
    handle_pr      : DINT;          (* Handle auf die Verwaltungsstruktur *)
    CobId_w        : WORD;          (* CobId *)
    Read_bit       : BOOL;          (* Richtung, TRUE für CAN-Teilnehmer ETCxM *)
END_VAR
```

Beschreibung

Mit dieser Funktion kann eine CobId in die Verwaltungsstruktur (s. Funktion DefineCanMsg()) für CAN-Transfers hinzugefügt werden. Die Funktion liefert TRUE, wenn die CobId erfolgreich in die Verwaltungsstruktur eingetragen wurde. Wenn die CobId schon in Benutzung ist oder die maximale Anzahl (32) der möglichen CobId erreicht ist, wird FALSE zurückgeliefert.

Beispiel

```
handle_pr      : DINT;
FctReturn_bit   : BOOL;

handle_pr = DefineCanMsg(1, 1014, 1114, 32);
FctReturn_bit = AddCobIdCanMsg(handle_pr, 1024, 1);
```

6.8.6.3 DelCobIdCanMsg (nur ETCxM)

Deklaration

```
FUNCTION DelCobIdCanMsg: BOOL
VAR_INPUT
    handle_pr      : DINT;          (* Handle auf die Verwaltungsstruktur *)
    CobId_w        : WORD;          (* CobId *)
END_VAR
```

Beschreibung

Mit dieser Funktion wird eine Cobld aus der Verwaltungsstruktur (s. Funktion DefineCanMsg()) für CAN-Transfers entfernt. Der Returnwert der Funktion ist ohne Bedeutung.

Beispiel

```
handle_pr      : DINT;

handle_pr = DefineCanMsg(1, 1014, 1114, 32);
DelCobldCanMsg(handle_pr, 1014);
```

6.8.6.4 CloseCanMsg (nur ETCxM)**Deklaration**

```
FUNCTION CloseCanMsg: BOOL
VAR_INPUT
    handle_pr      : DINT;          (* Handle auf die Verwaltungsstruktur *)
END_VAR
```

Beschreibung

Mit dieser Funktion wird die Verwaltungsstruktur für CAN-Transfers gelöscht (s. Funktion Define CanMsg()) und der Speicher wieder frei gegeben. Der Returnwert der Funktion ist ohne Bedeutung.

Beispiel

```
handle_pr

handle_pr = DefineCanMsg(1, 1014, 1114, 32);
CloseCanMsg(handle_pr);
```

6.8.6.5 ClearCanMsg (nur ETCxM)**Deklaration**

```
FUNCTION ClearCanMsg: BOOL
VAR_INPUT
    handle_pr      : DINT;          (* Handle auf die Verwaltungsstruktur *)
END_VAR
FUNCTION
```

Beschreibung

Mit dieser Funktion kann das Empfangs-FIFO gelöscht werden (s. Funktion DefineCanMsg()). Der Returnwert der Funktion ist ohne Bedeutung.

Beispiel

```
handle_pr      : DINT;

handle_pr = DefineCanMsg(1, 1014, 1114, 32);
ClearCanMsg(handle_pr);
```

6.8.6.6 ReadCanMsg (nur ETCxM)

Deklaration

```
FUNCTION ReadCanMsg: INT
VAR_INPUT
    handle_pr      : DINT;          (* Handle auf die Verwaltungsstruktur *)
    dest_pr        : DINT;          (* Adresse auf die Datenstruktur *)
END_VAR
```

Beschreibung

Mit dieser Funktion kann eine CAN-Nachricht aus dem Empfangspuffer gelesen werden. Der Empfangspuffer ist als FIFO implementiert. Die Struktur für Empfangsdaten hat folgenden Aufbau:

```
TYPE CAN_MSG_TR
STRUCT
    cobid_w      : WORD;
    size_b       : BYTE;
    rtr_bit      : BOOL;
    data_ab      : ARRAY [0..7] OF BYTE;
END_STRUCT
END_TYPE
```

Der Variablen dest_pr muss immer die Adresse einer Variablen des Typs CAN_MSG_TR übergeben werden (s. Beispiel). Diese Funktion kehrt in jedem Fall sofort zurück. Die Bedeutung der Returnwerte ist der folgenden Tabelle zu entnehmen.

Returnwert	Bedeutung
-2	Ungültiges Handle
0	Keine neue Nachricht im FIFO
1	Neue Nachricht vorhanden, keine weitere Nachricht im FIFO
2	Neue Nachricht vorhanden und weitere Nachrichten im FIFO

Beispiel

```
handle_pr      : DINT;
received_bit   : BOOL;
msg_st        : CAN_MSG_TR;

handle_pr = DefineCanMsg(1, 1014, 1114, 32);
handle_pr = DefineCanMsg(1, 1014, 1114, 32);
received_bit:=ReadCanMsg(handle_pr, ADR(msg_st));
```

6.8.6.7 WriteCanMsg (nur ETCxM)

Deklaration

```
FUNCTION WriteCanMsg: INT
VAR_INPUT
    handle_pr      : DINT;          (* Handle auf die Verwaltungsstruktur *)
    dest_pr        : DINT;          (* Adresse auf die Datenstruktur *)
END_VAR
```

Beschreibung

Mit dieser Funktion kann eine CAN-Nachricht in den Sendepuffer geschrieben werden. Der Sendepuffer ist als FIFO implementiert. Die Datenstruktur hat folgenden Aufbau:

```

TYPE CAN_MSG_TR
STRUCT
  cobid_w      : WORD;
  size_b       : BYTE;
  rtr_bit      : BOOL;
  data_ab      : ARRAY [0..7] OF BYTE;
END_STRUCT
END_TYPE

```

Der Variablen `source_pr` muss immer die Adresse einer Variablen des Typs `CAN_MSG_TR` übergeben werden (s. Beispiel). Diese Funktion kehrt in jedem Fall sofort zurück, auch wenn das FIFO voll ist. Die Bedeutung der Returnwerte ist der folgenden Tabelle zu entnehmen.

Returnwert	Bedeutung
-3	Nicht definierte Cobld in Datenstruktur
-2	Ungültiges Handle oder kein Speicherplatz für Nachricht vorhanden
-1	FIFO voll
0	Nachricht im FIFO eingetragen

Beispiel

```

handle_pr      : DINT;
ret_bit        : BOOL;
src_st         : CAN_MSG_TR;

```

```

handle_pr = DefineCanMsg(1, 1014, 1114, 32);
src_st.cobid_w := 1014;
src_st.size_b := 1;
src_st.rtr_bit := FALSE;
src_st.data_ab[0] := 27;
ret_bit = WriteCanMsg(handle_pr, ADR(src_st));

```

6.8.7 MMI-Kommunikationsfunktionen

6.8.7.1 GetApplicationMessage

Deklaration

```

FUNCTION_BLOCK GetApplicationMessage  (* holt eine Nachricht mit bis zu 512 Byte Nutz-
                                     daten vom MMI ab *)

VAR_INPUT
    data_pab      : POINTER TO ARRAY[0..511] OF BYTE;
    datatypes_s   : STRING(255);
END_VAR
VAR_OUTPUT
    sb1_b         : BYTE;
    sb2_b         : BYTE;
    index_b       : BYTE;
    handle_b      : BYTE;
    len_w         : WORD;
    ok            : BOOL;
END_VAR

```

Parameter

<code>data_pab</code>	Zeiger auf empfangene Nutzdaten
<code>datatypes_s</code>	optionaler Datendescrptor, siehe DefDataTypes
<code>sb1_b</code>	Steuerblock 1 der Nachricht vom HMI
<code>sb2_b</code>	Steuerblock 2, optional
<code>index_b</code>	Blockzähler, optional
<code>handle_b</code>	Kennung für Absender, reserviert
<code>len_w</code>	Länge der empfangenen Nutzdaten in Bytes
<code>ok</code>	TRUE, wenn eine Nachricht empfangen wurde

Beschreibung

Mit dieser Funktion kann die SPS eine Nachricht vom Typ SB0_SPSAUFRAG_KUC (SB0=14) mit bis zu 512 Byte Nutzdaten vom HMI abholen. Dabei wird der Nachrichtenpuffer zwischen NCR und MMI im Dual-Port-RAM verwendet. Die Funktion kann zusammen mit PutApplicationMessage verwendet werden, um eine schnelle Nachrichtenkommunikation zwischen SPS und HMI aufzubauen.

In *data_pab* kann ein Pointer auf einen Puffer übergeben werden, in dem die empfangene Nachricht abgelegt werden soll. Dabei ist sicherzustellen, dass dieser Puffer groß genug ist, um die empfangene Nachricht aufzunehmen. Beim Start oder Reset der SPS wird dieser Pointer automatisch auf den Anfang von DB7 (%MW7.0) initialisiert.

Wenn das Format der zu empfangenden Nachricht vorher bekannt ist, kann der Datendescrptor *datatypes_s* verwendet werden, um das eventuell notwendige Byte-Swapping der Nutzdaten automatisch vornehmen zu lassen. Andernfalls muss dies anschließend mit MOVESWAPPED durchgeführt oder die Funktionen GET_WORD, GET_DWORD, usw. verwendet werden, um die Daten aus der Nachricht zu holen. Sollte der angegebene Descriptorstring fehlerhaft sein, werden die Daten ohne Byte-Swapping übernommen.

Wenn beim Aufruf von `GetApplicationMessage` eine Nachricht vom HMI bereit steht, wird `ok` auf `TRUE` gesetzt und die Nachricht in den angegebenen Puffer kopiert. Die Funktion kehrt in jedem Fall sofort zurück, auch wenn keine Nachricht vom HMI bereit steht. In diesem Fall wird `ok` auf `FALSE` gesetzt.

Durch Auswertung von `sb1_b` kann die Nachricht von der SPS identifiziert werden. Die Festlegung der Steuerblöcke muss zwischen SPS- und MMI-Entwickler erfolgen. Die Länge der Nutzdaten wird in `len_w` in Bytes abgelegt.

Die Variablen `sb2_b` und `index_b` sind optional. Ihre Verwendung muss zwischen SPS- und HMI-Entwickler abgeklärt werden. `handle_b` ist für die gleichzeitige Kommunikation mit mehreren Applikationen reserviert.



Hinweis!

Die Übertragung dieser Nachrichten erfolgt grundsätzlich unsynchronisiert. In der Steuerung können bis zu 10 Nachrichten dieses Typs von MMI an SPS gepuffert werden, jede weitere geht verloren. In diesem Fall meldet die Steuerung eine entsprechende Fehlermeldung. Falls erforderlich müssen Synchronisationsmechanismen zwischen HMI und SPS vereinbart werden.

6.8.7.2 PutApplicationMessage

Deklaration

FUNCTION_BLOCK **PutApplicationMessage** (* Sendet eine Nachricht mit bis zu 512 Byte Nutzdaten ans MMI *)

VAR_INPUT

data_pab : POINTER TO ARRAY[0..511] OF BYTE;
 datatypes_s : STRING(255);
 sb1_b : BYTE;
 sb2_b : BYTE;
 index_b : BYTE;
 handle_b : BYTE;
 len_w : WORD;

END_VAR

VAR_INPUT

ok : BOOL;

END_VAR

Parameter

data_pab	Zeiger auf zu sendende Nutzdaten
datatypes_s	optionaler Datendescrptor, siehe DefDataTypes
sb1_b	Steuerblock 1 der Nachricht ans HMI
sb2_b	Steuerblock 2, optional
index_b	Blockzähler, optional
handle_b	Kennung für Empfänger, reserviert
len_w	Länge der empfangenen Nutzdaten in Bytes
ok	TRUE, wenn eine Nachricht gesendet wurde

Beschreibung

Mit dieser Funktion kann die SPS eine Nachricht vom Typ `SB0_SPSAUFTRA-G_KUC` (`SB0=14`) mit bis zu 512 Byte Nutzdaten ans HMI senden. Dabei wird der Nachrichtenpuffer zwischen NCR und MMI im Dual-Port-RAM verwendet. Die Funktion kann zusammen mit `GetApplicationMessage` verwendet werden, um eine schnelle Nachrichtenkommunikation zwischen SPS und MMI aufzubauen.

In `data_pab` kann ein Pointer auf einen Puffer mit den zu sendenden Nutzdaten übergeben werden. Beim Start oder Reset der SPS wird dieser Pointer automatisch auf den Anfang von `DB6 (%MW6.0)` initialisiert.

Der Datendescrptor `datatypes_s` kann verwendet werden, um das eventuell notwendige Byte-Swapping der Nutzdaten automatisch vornehmen zu lassen. Andernfalls muss dies vorher mit `MOVESWAPPED` durchgeführt oder die Funktionen `PUT_WORD`, `PUT_DWORD`, usw. verwendet werden, um die Daten in die Nachricht zu schreiben.

Wenn `datatypes_s` einen Leerstring enthält, muss die Anzahl der zu sendenden Nutzdatenbytes in `len_w` angegeben werden. Sonst ergibt sich die Länge aus dem Datendescrptor.

Mit `sb1_b` wird der Typ der Nachricht bestimmt. Für jedes verwendete Nutzdatenformat sollte ein eigener Steuerblock verwendet werden. Die Festlegung muss zwischen SPS- und MMI-Entwickler erfolgen.

Die Variablen `sb2_b` und `index_b` sind optional. Ihre Verwendung muss zwischen SPS- und HMI-Entwickler abgeklärt werden. `handle_b` ist für die gleichzeitige Kommunikation mit mehreren Applikationen reserviert.

Nach dem Eintragen der Nachricht in den Sendepuffer zum HMI, wird `ok` auf `TRUE` gesetzt. Bei einem fehlerhaften Descriptorstring `datatypes_s` oder einem Sendepufferüberlauf wird `ok` auf `FALSE` gesetzt. Die Funktion kehrt in jedem Fall sofort zurück, auch wenn die Nachricht nicht gesendet werden konnte. Im Falle eines Sendepufferüberlaufs kann der Aufruf natürlich später wiederholt werden.

**Hinweis!**

Die Übertragung dieser Nachrichten erfolgt grundsätzlich unsynchronisiert. In der Steuerung können insgesamt bis zu 30 Nachrichten von NCR an MMI gepuffert werden, jede weitere geht verloren. In diesem Fall meldet die Steuerung eine entsprechende Fehlermeldung. Falls erforderlich müssen Synchronisationsmechanismen zwischen HMI und SPS vereinbart werden.

6.8.8 Echtzeituhr (nur ETCxM)**6.8.8.1 RTC_GetTime_DT (nur ETCxM)****Deklaration**

```

FUNCTION RTC_GetTime_DT: BOOL
VAR_INPUT
    GetTime_pr      : DWORD;
END_VAR

```

Beschreibung

Mit dieser Funktion kann die Echtzeituhr der Steuerung ausgelesen werden. Der Funktion muss die Adresse einer Variablen vom Typ DT übergeben werden (siehe Beispiel). Der Returnwert TRUE der Funktion signalisiert, dass die Zeit ausgelesen werden konnte.

Beispiel

```

ret_bit      : BOOL;
time_dt      : DT;

ret_bit = RTC_GetTime_DT(ADR(time_dt));

```

6.8.8.2 RTC_SetTime_DT (nur ETCxM)**Deklaration**

```

FUNCTION RTC_SetTime_DT: BOOL
VAR_INPUT
    SetTime_dt      : DT;
END_VAR

```

Beschreibung

Mit dieser Funktion wird die Echtzeituhr der Steuerung auf den übergebenen Wert gesetzt. Der Returnwert TRUE der Funktion signalisiert, dass die Zeit in den Uhrenbaustein geschrieben wurde.

Beispiel

```

ret_bit      : BOOL;
time_dt      : DT := dt#2003-11-15-00:00:00;

ret_bit = RTC_SetTime_DT(time_dt);

```

6.9 Bibliothek ServerSDO.lib

6.9.1 InitServerSdo

Deklaration	<pre> FUNCTION InitServerSdo: BOOL VAR_INPUT Max_NetVarODIdx : UINT; pNetVarOD : POINTER TO NetVarOD_CAN; END_VAR </pre>				
Parameter	<table> <tr> <td>MAX_NetVarODIdx</td><td>maximal Index</td></tr> <tr> <td>pNetVarOD</td><td>Zeiger auf das Objektverzeichnis</td></tr> </table>	MAX_NetVarODIdx	maximal Index	pNetVarOD	Zeiger auf das Objektverzeichnis
MAX_NetVarODIdx	maximal Index				
pNetVarOD	Zeiger auf das Objektverzeichnis				
Beschreibung	Diese Funktion gibt dem Laufzeitsystem ein Objektverzeichnis zum Datenaustausch mit einem CANopen-Client bekannt.				
Beispiel	Um das CoDeSys-Objektverzeichnis zu aktivieren, muss folgender Funktionsaufruf einmalig im SPS-Programm durchgeführt werden: <pre> InitServerSdo(MAX_NetVarOD_CAN, ADR(pNetVarOD_CAN[0])); </pre>				

7 Stichwortverzeichnis

A

Achse, Rotations-, 72

Achse, virtuell, 72

Achsfehlermeldungen, 74

AddCobIdCanMsg, 277

Adresse CAN, 28

Adressierung

- Datenbaustein, 225

- E/A-Modul, 224

Adressierung, 224

ALLOCV24, 244

Antriebe, parametrieren, 19

Ausgangsparameter, Funktionsbaustein, 66

AXIS_REF, Datentyp, 67

B

Beispiel, SPS-Programm, 31

Bibliothek, 233

Bibliothek, ServerSDO.lib, 285

Bootloader, 235

Bootmonitor, Aufruf, 40

C

CAN

- Adresse, 28

- Master konfigurieren, 218

- Module suchen, 223

- Slave konfigurieren, 220

CAN-Bus, 51

CenterString, 233

ClearCanMsg, 278

CloseCanMsg, 278

CLRRXBUFFER, 247

CLRTXBUFFER, 247

CoDeSys, 214, 216, 234

- Adressierung, 224

- Datenbaustein, 225

- E/A-Modul, 224

- Bibliothek, 233

- Bibliothek, ServerSDO.lib, 285

- CAN-Master konfigurieren, 218

- CAN-Slave konfigurieren, 220

- CANopen-Module suchen, 223

- Datenbausteine, 232

- E/A-Anzahl überprüfen, 222

- E/A-Module konfigurieren, 218

- Fehlermeldungen, 232

- Funktionen

AddCobIdCanMsg, 277

allgemein, 233

ALLOCV24, 244

CANopen-Funktionen, 276

ClearCanMsg, 278

CloseCanMsg, 278

CLRRXBUFFER, 247

CLRTXBUFFER, 247

CopDefineDS403, 266

CopGetNodeState, 268

CopReadObject, 269

CopWriteObject, 270

CopXDefineDS403, 268

CopXDisableSync, 275

CopXEnableSync, 274

CopXReadObject, 270

CopXSetOperational, 272

CopXSetPreoperational, 273

CopXWriteObject, 272

DEFDATATYPES, 261

DefineCanMsg, 276

DelCobIdCanMsg, 277

FMOD, 233

Format, 234

FREEV24, 245

Gerätetreiber, 256

GET_BYTE, 263

GET_DINT, 263

GET_WORD, 263

GET_INT, 263

GET_LREAL, 263

GET_REAL, 263

GetApplicationMessage, 281

GetFirmwareVersion, 235

GetMacAddr, 236

GetUserParam, 236

Konstanten V24-Funktionen, 247 , 257

INITV24, 244

IO_SET, 237

LOAD, 243 , 249

LOAD_PARAM, 243

Load_Param, 237

MEMCOMP, 263

MEMCOPY, 263

MEMSET, 264

OVESWAPPED, 264

PUT_BYTE, 265

PUT_DWORD, 265

PUT_INT, 265

PUT_LREAL, 265

PUT_DINT, 265

PUT_WORD, 265

PutApplicationMessage, 282

READBLOCKV24, 245

ReadCanMsg, 279

READV24, 246

RTC_GetTime_DT, 284

RTC_SetTime_DT, 284

SAVE, 243 , 249

SAVE_PARAM, 243

CenterString, 233

SetCurrentPath, 250

SETINPUT_BIT, 238

SETINPUT_WORD, 238

SETLANGUAGE, 241

SPSERROR, 239

STRTOF, 241

STRTOL10, 241

SYSCLOSEFILE, 251

SYSDISKFORMAT, 253

SYSDISKINFO, 255

SYSEERROR, 242

SYSFIRSTFILE, 253

SYSNEXTFILE, 253

SYSOPENFILE, 250

SYSREADFILE, 251

SYSREADLINE, 252

SYSREMOVEFILE, 253

SYSWRITEFILE, 252

TRACE, 242

WordWrap, 243

WRITEBLOCKV24, 246

WriteCanMsg, 279

WRITEV24, 246

- installieren, 214

- Maschinenkonstanten, 232

- Netzwerkvariablen, 228

- Objektverzeichnis, 225

- Parametermanager, 225

- Variablenliste, global, 229 , 231

- programmieren mit, 23 , 213

- Projekt anlegen, 24

- Remanente Variablen, 225

- Schnittstelle zur ETC, 232

- SPS-Tasks konfigurieren, 216

- starten, 24

- Steuerung konfigurieren, 23

- Systemvariablen, 232

- Zielsystem einstellen, 216

- Zielsystemeinstellungen, 228

CopDefinedDS403, 266

CopGetNodeState, 268

CopReadObjekt, 269

CopWriteObject, 270

CopXDefineDS403, 268

CopXDisableSync, 275

CopXEnableSync, 274

CopXReadObject, 270

CopXSetOperational, 272

CopXSetPreoperational, 273

CopXWriteObject, 272

D

Datei

- EDS-, 26

- ETCxM.mk, 22

Datenbausteine, 232

Datentyp AXIS_REF, 67

DEFDATATYPES, 261

DefineCanMsg, 276

DelCobIdCanMsg, 277

E

E/A-Anzahl überprüfen, 222

E/A-Module konfigurieren, 218

EDS-Datei, 26

Eingangsparamter, Funktionsbaustein, 64

Erste Schritte, 10

ETC, Applikationen, 10

ETC und PC verbinden, 215

ETCHx

- IP-Adresse, 17

- mit PC verbinden, 12

- starten, 12

ETCxM.mk, 22

Ethernet-Schnittstelle, 215

F

Fehlergruppen, 73

Fehlerhandling, 73 , 202

Fehlerliste, 77

Fehlermeldungen, 232**Firmware, Update, 40****FMOD, 233****Format, 234****FREEV24, 245****Funktion**

- MC_Abort_Trigger, 123
- MC_AbortGearInPos, 176
- MC_AccelerationProfile, 116
- MC_CamExtTableSelect, 155
- MC_CamIn, 163
- MC_CamLargeTableSelect, 159
- MC_CamOut, 167
- MC_CamReadSlavePosition, 168
- MC_CamTableDelete, 162
- MC_CamTableSelect, 152
- MC_DefineAxis, 85
- MC_DefineAxisReset, 84
- MC_DigitalCamSwitch, 129
- MC_GearIn, 169
- MC_GearInPos, 172
- MC_GearOut, 177
- MC_Home, 104
- MC_MoveAbsolute, 89
- MC_MoveAdditive, 95
- MC_MoveRelative, 92
- MC_MoveSuperImposed, 98
- MC_MoveVelocity, 101
- MC_Phasing, 179
- MC_PositionProfile, 108
- MC_Power, 132
- MC_ReadActualPosition, 151
- MC_ReadActualTorque, 149
- MC_ReadActualVelocity, 148
- MC_ReadAxisError, 135
- MC_ReadBoolParameter, 139
- MC_ReadParameter, 138
- MC_ReadStatus, 133
- MC_Reset, 137
- MC_ResetPhaseShift, 182
- MC_SetOverride, 142
- MC_SetPosition, 145
- MC_Stop, 105
- MC_TorqueControl, 125
- MC_TouchProbe, 120
- MC_VelocityProfile, 112
- MC_WriteBoolParameter, 141
- MC_WriteMachineConstant, 82
- MC_WriteParameter, 140

Funktionen

- AddCobIdCanMsg, 277
- allgemein, 233
- ALLOCV24, 244
- CANopen-Funktionen, 276
- ClearCanMsg, 278
- CloseCanMsg, 278
- CLRRXBUFFER, 247
- CLRTXBUFFER, 247
- CopDefineDS403, 266
- CopGetNodeState, 268
- CopReadObjekt, 269
- CopWriteObject, 270
- CopXDefineDS403, 268
- CopXDisableSync, 275
- CopXEnableSync, 274
- CopXReadObject, 270
- CopXSetOperational, 272
- CopXSetPreoperational, 273
- CopXWriteObject, 272
- DEFDATATYPES, 261
- DefineCanMsg, 276
- DelCobIdCanMsg, 277
- FMOD, 233
- Format, 234
- FREEV24, 245
- Gerätetreiber, 256
- GET_BYTE, 263
- GET_DINT, 263
- GET_WORD, 263
- GET_INT, 263
- GET_LREAL, 263
- GET_REAL, 263
- GetApplicationMessage, 281
- GetFirmwareVersion, 235
- GetMacAddr, 236
- GetUserParam, 236
- I/O-Funktionen, 257
- INITV24, 244
- IO_SET, 237
- Konstanten V24-Funktionen, 247
- LOAD, 243, 249
- LOAD_PARAM, 243
- Load_Param, 237
- MEMCOMP, 263
- MEMCOPY, 263
- MEMSET, 264
- OVESWAPPED, 264
- PUT_BYTE, 265
- PUT_DINT, 265
- PUT_DWORD, 265

- PUT_INT, 265
- PUT_REAL, 265
- PUT_WORD, 265
- PutApplicationMessage, 282
- READBLOCKV24, 245
- ReadCanMsg, 279
- READV24, 246
- RTC_GetTime_DT, 284
- RTC_SetTime_DT, 284
- SAVE, 243, 249
- SAVE_PARAM, 243
- CenterString, 233
- SetCurrentPath, 250
- SETINPUT_BIT, 238
- SETINPUT_WORD, 238
- SETLANGUAGE, 241
- SPSERROR, 239
- STRTOF, 241
- STRTOL10, 241
- SYSCLOSEFILE, 251
- SYSDISKFORMAT, 253
- SYSDISKINFO, 255
- SYSERROR, 242
- SYSFIRSTFILE, 253
- SYSNEXTFILE, 253
- SYSOPENFILE, 250
- SYSREADFILE, 251
- SYSREADLINE, 252
- SYSREMOVEFILE, 253
- SYSWRITEFILE, 252
- TRACE, 242
- WordWrap, 243
- WRITEBLOCKV24, 246
- WriteCanMsg, 279
- WRITEV24, 246

Funktionsbaustein

- Ausgangsparameter, 66
- Eingangsparameter, 64
- gemeldete Fehler, 77
- MC_Abort_Trigger_CAN_LH, 200
- MC_AttachTaskToInt, 190
- MC_CalcTimeToMove, 193
- MC_CalcVelocityToMove, 195
- MC_DefineQuickOutputs, 185
- MC_DefineQuickInputs, 183
- MC_InitAxes, 88
- MC_ReadDigitalInput, 187
- MC_ReadDigitalOutput, 188
- MC_ReadEvent, 205
- MC_ReadSysError, 202
- MC_ResetSysError, 204
- MC_TouchProbe_CAN_LH, 197
- MC_WriteDigitalOutput, 189
- Position und Distanz, 66
- Schnittstelle, 64
- Technische Einheiten, 72
- Übersicht, 62
- Vorzeichenregel, 66

Funktionsbibliothek

- Datentyp AXIS_REF, 67
- MC.lib, 82
- Technische Einheiten, 72

G

Gerätetreiber, 256

GET_BYTE, 263

GET_DINT, 263

GET_DWORD, 263

GET_INT, 263

GET_LREAL, 263

GET_REAL, 263

GET_WORD, 263

GetApplicationMessage, 281

GetFirmwareVersion, 235

GetMacAddr, 236

GetUserParam, 236

Grundlagen Motion-Steuerung, 43

Guard Time, 28

H

HyperTerminal, 13

I

Inbetriebnahme, Inbetriebnahmeschritte, Übersicht, 11

INITV24, 244

Interpolator, 43

IO_SET, 237

IP-Adresse, ETCHx, 17

IPC, Applikationen, 10

K

Konfiguration, 46, 61

- Quickstop (QSP), 72
- Software-Endschalter, 71

Konstanten

- CANopen-Funktionen, 276
- I/O-Funktionen, 257
- V24-Funktionen, 247

L

Life Time Factor, 28

LOAD, 243, 249

LOAD_PARAM, 243

Load_Param, 237

M

MAC-Adresse, 18

Maschinenkonstante, 46, 61

- Funktion, 49
- MK_INTERPOL_CYCLE, 59

Maschinenkonstanten, 232

- ETCxM.mk, 22
- Übersicht, 19

MC.lib, 82

- Funktionsübersicht, 62
- Schnittstelle, 64

MC_Abort_Trigger, 123

MC_Abort_Trigger_CAN_LH, 200

MC_AbortGearInPos, 176

MC_AccelerationProfile, 116

MC_AttachTaskToInt, 190

MC_CalcTimeToMove, 193

MC_CalcVelocityToMove, 195

MC_CamExtTableSelect, 155

MC_CamIn, 163

MC_CamLargeTableSelect, 159

MC_CamOut, 167

MC_CamReadSlavePosition, 168

MC_CamTableDelete, 162

MC_CamTableSelect, 152

MC_DefineAxis, 85

MC_DefineAxisReset, 84

MC_DefineQuickInputs, 185

MC_DefineQuikInputs, 183

MC_DigitalCamSwitch, 129

MC_GearIn, 169

MC_GearInPos, 172

MC_GearOut, 177

MC_Home, 104

MC_InitAxes, 88

MC_MoveAbsolute, 89

MC_MoveAdditive, 95

MC_MoveRelative, 92

MC_MoveSuperImposed, 98

MC_MoveVelocity, 101

MC_Phasing, 179

MC_PositionProfile, 108

MC_Power, 132

MC_ReadActualPosition, 151

MC_ReadActualTorque, 149

MC_ReadActualVelocity, 148

MC_ReadAxisError, 135

MC_ReadBoolParameter, 139

MC_ReadDigitalInput, 187

MC_ReadDigitalOutput, 188

MC_ReadEvent, 205

MC_ReadParameter, 138

MC_ReadStatus, 133

MC_ReadSysError, 202

MC_Reset, 137

MC_ResetPhaseShift, 182

MC_ResetSysError, 204

MC_SetOverride, 142

MC_SetPosition, 145

MC_Stop, 105

MC_TorqueControl, 125

MC_TouchProbe, 120

MC_TouchProbe_CAN_LH, 197
 MC_VelocityProfile, 112
 MC_WriteBoolParameter, 141
 MC_WriteDigitalOutput, 189
 MC_WriteMachineConstant, 82
 MC_WriteParameter, 140
 MEMCOMP, 263
 MEMCOPY, 263
 MEMSET, 264
 MK_INTERPOL_CYCLE, 59
 Monitor-Schnittstelle
 - aktivieren, 13
 - Befehle, 15
 - Störungsbeseitigung, 16

N

Netzwerkvariablen, 228
 Nodeguarding, 28 , 221 , 223

O

Objektverzeichnis, 225
 OVESWAPPED, 264

P

Parameter
 - Anwender-spezifisch, 208
 - CANopen, 206
 Parameterliste, 206
 Parametermanager, 225
 Parametrieren, Antriebe, 19
 PC, Applikationen, 10
 PC und ETC verbinden, 215
 Position und Distanz, Funktionsbaustein, 66
 Produktkennung, 9
 Programm generieren, 231
 Programmierung mit CoDeSys, 23 , 213
 PUT_BYTE, 265
 PUT_DINT, 265
 PUT_DWORD, 265
 PUT_INT, 265
 PUT_LREAL, 265
 PUT_REAL, 265
 PUT_WORD, 265

PutApplicationMessage, 282

Q

Quickstop, 72

R

READBLOCKV24, 245
 ReadCanMsg, 279
 READV24, 246
 Remanente Variablen, 225
 Rotationsachse, 72
 RTC_GetTime_DT, 284
 RTC_SetTime_DT, 284

S

SAVE, 243 , 249
 SAVE_PARAM, 243
 Schnellhalt, 72
 Schnittstelle
 - Ethernet, 215
 - Monitor-, 13 , 15
 - V.24, 215
 Schnittstelle zur ETC, 232
 ServerSDO.lib, 285
 SetCurrentPath, 250
 SETINPUT_BIT, 238
 SETINPUT_WORD, 238
 SETLANGUAGE, 241
 Software-Endschalter, 71
 SPS-Programm
 - bedienen, 37
 - Beispiel, 31
 - Bibliotheken einfügen, 32
 - erstellen, 31
 - konfigurieren, 31
 - laden, 34 , 37
 - starten, 31 , 37
 - testen, 38
 SPS-Tasks konfigurieren, 216
 SPSError, 239
 Steuerung konfigurieren, 23 , 46 , 61
 Störungsbeseitigung, Monitor-Schnittstelle, 16
 STRTOF, 241
 STRTOL10, 241

SYSCLOSEFILE, 251
SYSDISKFORMAT, 253
SYSDISKINFO, 255
SYSERROR, 242
SYSFIRSTFILE, 253
SYSNEXTFILE, 253
SYSOPENFILE, 250
SYSREADFILE, 251
SYSREADLINE, 252
SYSREMOVEFILE, 253
Systemfehlermeldungen, 75
Systemvariablen, 232
SYSWRITEFILE, 252

T

Technische Einheiten, Funktionsbibliothek, 72
Terminalprogramm, 13
Testen, SPS-Programm, 38
TRACE, 242
Typenschild, 9
Typenschlüssel, 9

U

Übersicht, Maschinenkonstanten, 19
Update, Firmware, 40

V

V.24-Schnittstelle, 215
Variablenliste, global, 229
Virtuelle Achse, 72
Visualisierungsfenster, 38
Visualisierungsobjekt, 39
Vorzeichenregel, Funktionsbaustein, 66

W

WordWrap, 243
WRITEBLOCKV24, 246
WriteCanMsg, 279
WRITEV24, 246

Z

Zielsystem einstellen, 216
Zielsystemeinstellungen, 228
Zustände des Interpolators, 43



Lenze Drive Systems GmbH
Hans-Lenze-Straße 1
D-31855 Aerzen
Germany



+49 (0) 51 54 82-0



Service

00 80 00 24 4 68 77 (24 h helpline)



Service

+49 (0) 51 54 82-1112

E-Mail

Lenze@Lenze.de

Internet

www.Lenze.com

EDSTCXM2G
DE 1.0
© 06/2008
TD29