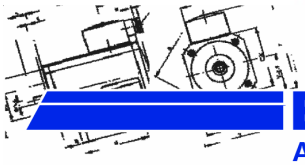
metronix
servo drives

CANopen



Servopositionierregler DIS-2

CANopen Handbuch

Metronix Meßgeräte und Elektronik GmbH
Kocherstraße 3
D-38120 Braunschweig
Germany

Telefon: +49-(0)531-8668-0
Telefax: +49-(0)531-8668-555
E-mail: vertrieb@metronix.de
http://www.metronix.de

metronix
servo drives

Urheberrechte

Die Informationen und Angaben in diesem Dokument sind nach bestem Wissen zusammengestellt worden. Trotzdem können abweichende Angaben zwischen dem Dokument und dem Produkt nicht mit letzter Sicherheit ausgeschlossen werden. Für die Geräte und zugehörige Programme in der dem Kunden überlassenen Fassung gewährleistet Metronix den vertragsgemäßen Gebrauch in Übereinstimmung mit der Nutzerdokumentation. Im Falle erheblicher Abweichungen von der Nutzerdokumentation ist Metronix zur Nachbesserung berechtigt und, soweit diese nicht mit unangemessenem Aufwand verbunden ist, auch verpflichtet. Eine eventuelle Gewährleistung erstreckt sich nicht auf Mängel, die durch Abweichen von den für das Gerät vorgesehenen und in der Nutzerdokumentation angegebenen Einsatzbedingungen verursacht werden.

Metronix übernimmt keine Gewähr dafür, dass die Produkte den Anforderungen und Zwecken des Erwerbers genügen oder mit anderen von ihm ausgewählten Produkten zusammenarbeiten. Metronix übernimmt keine Haftung für Folgeschäden, die im Zusammenwirken der Produkte mit anderen Produkten oder aufgrund unsachgemäßer Handhabung an Maschinen oder Anlagen entstehen.

Metronix behält sich das Recht vor, das Dokument oder das Produkt ohne vorherige Ankündigung zu ändern, zu ergänzen oder zu verbessern.

Dieses Dokument darf weder ganz noch teilweise ohne ausdrückliche Genehmigung des Urhebers in irgendeiner Form reproduziert oder in eine andere natürliche oder maschinenlesbare Sprache oder auf Datenträger übertragen werden, sei es elektronisch, mechanisch, optisch oder auf andere Weise.

Warenzeichen

Alle Produktnamen in diesem Dokument können eingetragene Warenzeichen sein. Alle Warenzeichen in diesem Dokument werden nur zur Identifikation des jeweiligen Produkts verwendet.
ServoCommander™ ist ein eingetragenes Warenzeichen der Metronix Meßgeräte und Elektronik GmbH.
Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Metronix Meßgeräte und Elektronik GmbH
Kocherstraße 3
D-38120 Braunschweig
Germany

Telefon: +49-(0)531-8668-0
Telefax: +49-(0)531-8668-555
E-mail: vertrieb@metronix.de
http://www.metronix.de

metronix
servo drives

Verzeichnis der Revisionen			
Autor:			
Handbuchname:	CANopen Handbuch		
Dateiname:	CAN-HB_DIS-2_2p0_DE.odt		
Lfd. Nr.	Beschreibung	Revisions-index	Datum der Änderung
001	Erstellung des Handbuches	1.0	22.09.2004
002	Ergänzungen und Änderngen lt. CANopen PDO's Behandlung, voltage_enabled	1.1	10.11.2004
003	Corporate Identity aktualisiert – keine technischen Änderungen	2.0	02.05.2011

Metronix Meßgeräte und Elektronik GmbH
Kocherstraße 3
D-38120 Braunschweig
Germany

Telefon: +49-(0)531-8668-0
Telefax: +49-(0)531-8668-555
E-mail: vertrieb@metronix.de
http://www.metronix.de

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Allgemeines

1.1 Dokumentation

Das vorliegende Handbuch beschreibt, wie der Servopositionierregler vom Typ DIS-2 in eine CANopen-Netzwerkumgebung einbezogen werden kann. Es wird die Einstellung der physikalischen Parameter, die Aktivierung des CANopen-Protokolls, die Einbindung in das CAN-Netzwerk und die Kommunikation mit dem Servopositionierregler beschrieben. Es richtet sich an Personen, die bereits mit dieser Servopositionierregler DIS-2 vertraut sind.

Es enthält Sicherheitshinweise, die beachtet werden müssen.

Weitergehende Informationen finden sich im Handbuch zum Servopositionierregler DIS-2.

1.2 CANopen

CANopen ist ein von der Vereinigung „CAN in Automation" erarbeiteter Standard. In diesem Verbund sind eine Vielzahl von Geräteherstellern organisiert. Dieser Standard hat die bisherigen herstellerspezifischen CAN-Protokolle weitgehend ersetzt. Somit steht dem Endanwender ein herstellerunabhängiges Kommunikations-Interface zur Verfügung.

Von diesem Verbund sind unter anderem folgende Handbücher beziehbar:

CiA Draft Standard 201-207: In diesen Werken werden die allgemeinen Grundlagen und die Einbettung von CANopen in das OSI-Schichtenmodell behandelt. Die relevanten Punkte dieses Buches werden im vorliegenden CANopen-Handbuch vorgestellt, so dass der Erwerb der DS201..207 im allgemeinen nicht notwendig ist.

CiA Draft Standard 301: In diesem Werk wird der grundsätzliche Aufbau des Objektverzeichnisses eines CANopen-Gerätes und der Zugriff auf dieses beschrieben. Außerdem werden die Aussagen der DS201..207 konkretisiert. Die für den Servopositionierregler DIS-2 benötigten Elemente des Objektverzeichnisses und die zugehörigen Zugriffsmethoden sind im vorliegendem Handbuch beschrieben. Der Erwerb der DS301 ist ratsam aber nicht unbedingt notwendig.

CiA Draft Standard 402: Dieses Buch befasst sich mit der konkreten Implementation von CANopen in Antriebsregler. Obwohl alle implementierten Objekte auch im

vorliegenden CANopen-Handbuch in kurzer Form dokumentiert und beschrieben sind, sollte der Anwender über dieses Werk verfügen.

Bezugsadresse:

CAN in Automation (CiA) International Headquarter
Am Weichselgarten 26
D-91058 Erlangen
Tel.: 09131-601091
Fax: 09131-601092
www.can-cia.de

2 Sicherheitshinweise für elektrische Antriebe und Steuerungen

2.1 Verwendete Symbole

Information Wichtige Informationen und Hinweise.

Vorsicht! Die Nichtbeachtung kann hohe Sachschäden zur Folge haben.

GEFAHR ! Die Nichtbeachtung kann **Sachschäden** und **Personenschäden** zur Folge haben.

Vorsicht! Lebensgefährliche Spannung. Der Sicherheitshinweis enthält einen Hinweis auf eine eventuell auftretende lebensgefährliche Spannung.

Die mit diesem Symbol gekennzeichneten Abschnitte stellen Beispiele dar, die das Verständnis und die Anwendung einzelner Objekte und Parameter erleichtern.

2.2 Allgemeine Hinweise

Bei Schäden infolge von Nichtbeachtung der Warnhinweise in dieser Betriebsanleitung übernimmt Metronix keine Haftung.

> Vor der Inbetriebnahme sind die Sicherheitshinweise für elektrische Antriebe und Steuerungen *ab Seite 12* durchzulesen.

Wenn die Dokumentation in der vorliegenden Sprache nicht einwandfrei verstanden wird, bitte beim Lieferant anfragen und diesen informieren.

Der einwandfreie und sichere Betrieb des Servopositionierreglers setzt den sachgemäßen und fachgerechten Transport, die Lagerung, die Montage, die Projektierung, unter der Beachtung der Risiken und Schutz- und Notfallmaßnahmen und die Installation sowie die sorgfältige Bedienung und die Instandhaltung voraus. Für den Umgang mit elektrischen Anlagen ist ausschließlich ausgebildetes und qualifiziertes Personal einzusetzen:

AUSGEBILDETES UND QUALIFIZIERTES PERSONAL

im Sinne dieses Produkthandbuches bzw. der Warnhinweise auf dem Produkt selbst sind Personen, die mit der Projektierung, der Aufstellung, der Montage, der Inbetriebsetzung und dem Betrieb des Produktes sowie mit allen Warnungen und Vorsichtsmaßnahmen gemäß dieser Betriebsanleitung in diesem Produkthandbuch ausreichend vertraut sind und die über die ihrer Tätigkeit entsprechenden Qualifikationen verfügen:

- Ausbildung und Unterweisung der Normen und Unfallverhütungsvorschriften, die in Zusammenhang mit der Anwendung stehen, bzw. Berechtigung, Geräte/Systeme gemäß den Standards der Sicherheitstechnik ein- und auszuschalten, zu erden und gemäß den Arbeitsanforderungen zweckmäßig zu kennzeichnen.
- Ausbildung oder Unterweisung gemäß den Standards der Sicherheitstechnik in Pflege und Gebrauch angemessener Sicherheitsausrüstung.
- Schulung in Erster Hilfe.

Die nachfolgenden Hinweise sind vor der ersten Inbetriebnahme der Anlage zur Vermeidung von Körperverletzungen und/oder Sachschäden zu lesen:

> Diese Sicherheitshinweise sind jederzeit einzuhalten.
>
> Vor der Installation und der Inbetriebnahme sind alle Sicherheitshinweise für elektrische Antriebe und Steuerungen in diesem Dokument sorgfältig zu lesen. Diese Sicherheitsinstruktionen und alle anderen Benutzerhinweise sind vor jeder Arbeit mit dem Servopositionierregler durchzulesen.
>
> Sollten keine Benutzerhinweise für den Servopositionierregler zur Verfügung stehen, wenden Sie sich an Ihren zuständigen Vertriebsrepräsentanten

Bei Verkauf, Verleih und/oder anderweitiger Weitergabe des Servopositionierreglers sind diese Sicherheitshinweise ebenfalls mitzugeben.

Ein Öffnen des Servopositionierreglers durch den Betreiber ist aus Sicherheits- und Gewährleistungsgründen nicht zulässig.

Die Voraussetzung für eine einwandfreie Funktion des Servopositionierreglers ist eine fachgerechte Projektierung!

GEFAHR!

Unsachgemäßer Umgang mit dem Servopositionierregler und Nichtbeachten der hier angegebenen Warnhinweise sowie unsachgemäße Eingriffe in die Sicherheitseinrichtung können zu Sachschaden, Körperverletzung, elektrischem Schlag oder im Extremfall zum Tod führen.

2.3 Gefahren durch falschen Gebrauch

GEFAHR!

Hohe elektrische Spannung und hoher Arbeitsstrom!

Lebensgefahr oder schwere Körperverletzung durch elektrischen Schlag!

GEFAHR!

Hohe elektrische Spannung durch falschen Anschluss!

Lebensgefahr oder Körperverletzung durch elektrischen Schlag!

GEFAHR!

Heiße Oberflächen auf Gerätegehäuse möglich!

Verletzungsgefahr! Verbrennungsgefahr!

GEFAHR!

Gefahrbringende Bewegungen!

Lebensgefahr, schwere Körperverletzung oder Sachschaden durch unbeabsichtigte Bewegungen der Motoren!

2.4 Sicherheitshinweise

2.4.1 Allgemeine Sicherheitshinweise

Der Servopositionierregler entspricht je nach Ausführung der Schutzklasse IP40..IP65, sowie der Verschmutzungsklasse 1. Es ist darauf zu achten, dass die Umgebung dieser Schutz- bzw. Verschmutzungsklasse entspricht.

Nur vom Hersteller zugelassene Zubehör- und Ersatzteile verwenden.

Die Spannungsversorgunen des Servopositionierreglers müssen entsprechend den EN-Normen und VDE-Vorschriften so an das Netz angeschlossen werden, dass sie mit geeigneten Freischaltmitteln (z.B. Hauptschalter, Schütz, Leistungsschalter) vom Netz getrennt werden können.

Es sind die Sicherheitsvorschriften und -bestimmungen des Landes, in dem das Gerät zur Anwendung kommt, zu beachten.

Die in der Betriebsanleitung angegebenen Umgebungsbedingungen müssen eingehalten werden. Sicherheitskritische Anwendungen sind nicht zugelassen, sofern sie nicht ausdrücklich vom Hersteller freigegeben werden.

Die Hinweise für eine EMV-gerechte Installation sind aus der Betriebsanleitung zu entnehmen. Die Einhaltung der durch die nationalen Vorschriften geforderten Grenzwerte liegt in der Verantwortung der Hersteller der Anlage oder Maschine.

Die technischen Daten, die Anschluss- und Installationsbedingungen für den Servopositionierregler sind aus diesem Produkthandbuch zu entnehmen und unbedingt einzuhalten.

GEFAHR!

Es sind die Allgemeinen Errichtungs- und Sicherheitsvorschriften für das Arbeiten an Starkstromanlagen (z.B. DIN, VDE, EN, IEC oder andere nationale und internationale Vorschriften) zu beachten.

Nichtbeachtung können Tod, Körperverletzung oder erheblichen Sachschaden zur Folge haben.

Ohne Anspruch auf Vollständigkeit gelten unter anderem folgende Vorschriften:

VDE 0100	Bestimmung für das Errichten von Starkstromanlagen bis 1000 Volt
EN 60204	Elektrische Ausrüstung von Maschinen
EN 50178	Ausrüstung von Starkstromanlagen mit elektronischen Betriebsmitteln

2.4.2 Sicherheitshinweise bei Montage und Wartung

Für die Montage und Wartung der Anlage gelten in jedem Fall die einschlägigen DIN, VDE, EN und IEC - Vorschriften, sowie alle staatlichen und örtlichen Sicherheits- und Unfallverhütungsvorschriften. Der Anlagenbauer bzw. der Betreiber hat für die Einhaltung dieser Vorschriften zu sorgen:

Die Bedienung, Wartung und/oder Instandsetzung des Servopositionierreglers darf nur durch für die Arbeit an oder mit elektrischen Geräten ausgebildetes und qualifiziertes Personal erfolgen.

Vermeidung von Unfällen, Körperverletzung und/oder Sachschaden:

Vertikale Achsen gegen Herabfallen oder Absinken nach Abschalten des Motors zusätzlich sichern, wie durch:

- mechanische Verriegelung der vertikalen Achse,
- externe Brems-/ Fang-/ Klemmeinrichtung oder
- ausreichenden Gewichtsausgleich der Achse.

Die serienmäßig gelieferte Motor-Haltebremse oder eine externe, vom Antriebsregelgerät angesteuerte Motor-Haltebremse alleine ist nicht für den Personenschutz geeignet!

Die elektrische Ausrüstung über den Hauptschalter spannungsfrei schalten und gegen Wiedereinschalten sichern, warten bis der Zwischenkreis entladen ist bei:

- Wartungsarbeiten und Instandsetzung
- Reinigungsarbeiten
- langen Betriebsunterbrechungen

Vor der Durchführung von Wartungsarbeiten ist sicherzustellen, dass die Stromversorgung abgeschaltet, verriegelt und der Zwischenkreis entladen ist.

Bei der Montage ist sorgfältig vorzugehen. Es ist sicherzustellen, dass sowohl bei Montage als auch während des späteren Betriebes des Antriebs keine Bohrspäne, Metallstaub oder Montageteile (Schrauben, Muttern, Leitungsabschnitte) in den Servopositionierregler fallen.

Ebenfalls ist sicherzustellen, dass die externe Spannungsversorgung des Servopositionierreglers (24V) abgeschaltet ist.

Ein Abschalten des Zwischenkreises muss immer vor dem Abschalten der 24V Reglerversorgung erfolgen.

Die Arbeiten im Maschinenbereich sind nur bei abgeschalteter und verriegelter Wechselstrom- bzw. Gleichstromversorgung durchzuführen. Abgeschaltete Endstufen oder abgeschaltete Reglerfreigabe sind keine geeigneten Verriegelungen. Hier kann es im Störungsfall zum unbeabsichtigten Verfahren des Antriebes kommen.

Die Inbetriebnahme mit leerlaufenden Motoren durchführen, um mechanische Beschädigungen, z.B. durch falsche Drehrichtung zu vermeiden.

Elektronische Geräte sind grundsätzlich nicht ausfallsicher. Der Anwender ist dafür verantwortlich, dass bei Ausfall des elektrischen Geräts seine Anlage in einen sicheren Zustand geführt wird.

Der Servopositionierregler und insbesondere der Bremswiderstand, extern oder intern, können hohe Temperaturen annehmen, die bei Berührung schwere körperliche Verbrennungen verursachen können.

2.4.3 Schutz gegen Berühren elektrischer Teile

Dieser Abschnitt betrifft nur Geräte und Antriebskomponenten mit Spannungen über 50 Volt. Werden Teile mit Spannungen größer 50 Volt berührt, können diese für Personen gefährlich werden und zu elektrischem Schlag führen. Beim Betrieb elektrischer Geräte stehen zwangsläufig bestimmte Teile dieser Geräte unter gefährlicher Spannung.

GEFAHR!

Hohe elektrische Spannung!

Lebensgefahr, Verletzungsgefahr durch elektrischen Schlag oder schwere Körperverletzung!

Für den Betrieb gelten in jedem Fall die einschlägigen DIN, VDE, EN und IEC - Vorschriften, sowie alle staatlichen und örtlichen Sicherheits- und Unfallverhütungsvorschriften. Der Anlagenbauer bzw. der Betreiber hat für die Einhaltung dieser Vorschriften zu sorgen:

Vor dem Einschalten die dafür vorgesehenen Abdeckungen und Schutzvorrichtungen für den Berührschutz an den Geräten anbringen. Für Einbaugeräte ist der Schutz gegen direktes Berühren elektrischer Teile durch ein äußeres Gehäuse, wie beispielsweise einen Schaltschrank, sicherzustellen. Die Vorschriften VBG 4 sind zu beachten!

Den Schutzleiter der elektrischen Ausrüstung und der Geräte stets fest an das Versorgungsnetz anschließen.

Nach der Norm EN60617 den vorgeschriebenen Mindest-Kupfer-Querschnitt für die Schutzleiterverbindung in seinem ganzen Verlauf beachten!

Vor Inbetriebnahme, auch für kurzzeitige Mess- und Prüfzwecke, stets den Schutzleiter an allen elektrischen Geräten entsprechend dem Anschlussplan anschließen oder mit Erdleiter verbinden. Auf dem Gehäuse können sonst hohe Spannungen auftreten, die elektrischen Schlag verursachen.

Elektrische Anschlussstellen der Komponenten im eingeschalteten Zustand nicht berühren.

Vor dem Zugriff zu elektrischen Teilen mit Spannungen größer 50 Volt das Gerät vom Netz oder von der Spannungsquelle trennen. Gegen Wiedereinschalten sichern.

Bei der Installation ist besonders in Bezug auf Isolation und Schutzmaßnahmen die Höhe der Zwischenkreisspannung zu berücksichtigen. Es muss für ordnungsgemäße Erdung, Leiterdimensionierung und entsprechenden Kurzschlussschutz gesorgt werden.

2.4.4 Schutz durch Schutzkleinspannung (PELV) gegen elektrischen Schlag

Alle Anschlüsse und Klemmen mit Spannungen von 5 bis 50 Volt an dem Servopositionierregler sind Schutzkleinspannungen, die entsprechend folgender Normen berührungssicher ausgeführt sind:

international: IEC 60364-4-41

Europäische Länder in der EU: EN 50178/1998, Abschnitt 5.2.8.1.

GEFAHR!

Hohe elektrische Spannung durch falschen Anschluss!

Lebensgefahr, Verletzungsgefahr durch elektrischen Schlag!

An alle Anschlüsse und Klemmen mit Spannungen von 0 bis 50 Volt dürfen nur Geräte, elektrische Komponenten und Leitungen angeschlossen werden, die eine Schutzkleinspannung (PELV = Protective Extra Low Voltage) aufweisen.

Nur Spannungen und Stromkreise, die sichere Trennung zu gefährlichen Spannungen haben, anschließen. Sichere Trennung wird beispielsweise durch Trenntransformatoren, sichere Optokoppler oder netzfreien Batteriebetrieb erreicht.

2.4.5 Schutz vor gefährlichen Bewegungen

Gefährliche Bewegungen können durch fehlerhafte Ansteuerung von angeschlossenen Motoren verursacht werden. Die Ursachen können verschiedenster Art sein:

- unsaubere oder fehlerhafte Verdrahtung oder Verkabelung
- Fehler bei der Bedienung der Komponenten
- Fehler in den Messwert- und Signalgebern
- defekte oder nicht EMV-gerechte Komponenten
- Fehler in der Software im übergeordneten Steuerungssystem

Diese Fehler können unmittelbar nach dem Einschalten oder nach einer unbestimmten Zeitdauer im Betrieb auftreten.

Die Überwachungen in den Antriebskomponenten schließen eine Fehlfunktion in den angeschlossenen Antrieben weitestgehend aus. Im Hinblick auf den Personenschutz, insbesondere der Gefahr der Körperverletzung und/oder Sachschaden, darf auf diesen Sachverhalt nicht allein vertraut werden. Bis zum Wirksamwerden der eingebauten Überwachungen ist auf jeden Fall mit einer fehlerhaften Antriebsbewegung zu rechnen, deren Maß von der Art der Steuerung und des Betriebszustandes abhängen.

GEFAHR!

Gefahrbringende Bewegungen!

Lebensgefahr, Verletzungsgefahr, schwere Körperverletzung oder Sachschaden!

Der Personenschutz ist aus den oben genannten Gründen durch Überwachungen oder Maßnahmen, die anlagenseitig übergeordnet sind, sicherzustellen. Diese werden nach den spezifischen Gegebenheiten der Anlage einer Gefahren- und Fehleranalyse vom Anlagenbauer vorgesehen. Die für die Anlage geltenden Sicherheitsbestimmungen werden hierbei mit einbezogen. Durch Ausschalten, Umgehen oder fehlendes Aktivieren von Sicherheitseinrichtungen können willkürliche Bewegungen der Maschine oder andere Fehlfunktionen auftreten.

2.4.6 Schutz gegen Berühren heißer Teile

GEFAHR!

Heiße Oberflächen auf Gerätegehäuse möglich!

Verletzungsgefahr! Verbrennungsgefahr!

Gehäuseoberfläche in der Nähe von heißen Wärmequellen nicht berühren! Verbrennungsgefahr!

Vor dem Zugriff Geräte nach dem Abschalten erst 10 Minuten abkühlen lassen.

Werden heiße Teile der Ausrüstung wie Gerätegehäuse, in denen sich Kühlkörper und Widerstände befinden, berührt, kann das zu Verbrennungen führen!

2.4.7 Schutz bei Handhabung und Montage

Die Handhabung und Montage bestimmter Teile und Komponenten in ungeeigneter Art und Weise kann unter ungünstigen Bedingungen zu Verletzungen führen.

GEFAHR!

Verletzungsgefahr durch unsachgemäße Handhabung!

Körperverletzung durch Quetschen, Scheren, Schneiden, Stoßen!

Hierfür gelten allgemeine Sicherhinweise:

Die allgemeinen Errichtungs- und Sicherheitsvorschriften zu Handhabung und Montage beachten.

Geeignete Montage- und Transporteinrichtungen verwenden.

Einklemmungen und Quetschungen durch geeignete Vorkehrungen vorbeugen.

Nur geeignetes Werkzeug verwenden. Sofern vorgeschrieben, Spezialwerkzeug benutzen.

Hebeeinrichtungen und Werkzeuge fachgerecht einsetzen.

Wenn erforderlich, geeignete Schutzausstattungen (zum Beispiel Schutzbrillen, Sicherheitsschuhe, Schutzhandschuhe) benutzen.

Nicht unter hängenden Lasten aufhalten.

Auslaufende Flüssigkeiten am Boden sofort wegen Rutschgefahr beseitigen.

3 Verkabelung und Steckerbelegung

3.1 Anschlussbelegungen

Das CAN-Interface ist bei den Geräten DIS-2 bereits im Servopositionierregler integriert und somit immer verfügbar.

Der CAN-Bus-Anschluss ist je nach Reglertyp auf unterschiedlichen Steckern zu finden. Hinzu kommt, dass je nach Ausführung eine Doppelbelegung mit digitalen oder analogen Eingängen vorhanden ist. Um den CAN-Bus benutzen zu können, müssen die doppelt belegten Pins entsprechend parametriert bzw. die Hardware(DIS-24/8) angepasst werden. Genaueres ist in dem Handbuch für den DIS-2 nachzulesen.

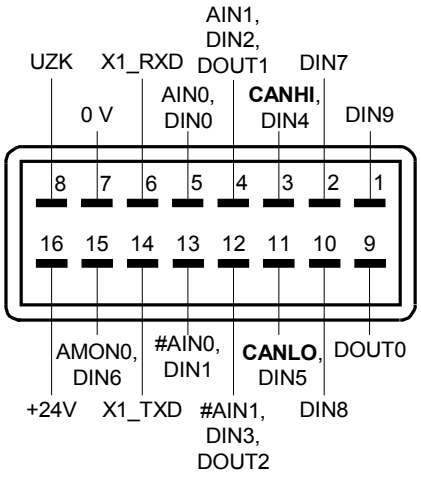


Abbildung 3.1: DIS-2 Steckverbinder.

CAN-Bus-Verkabelung
Bei der Verkabelung der Regler über den CAN-Bus sollten sie unbedingt die nachfolgenden Informationen und Hinweise beachten, um ein stabiles, störungsfreies System zu erhalten. Bei einer nicht sachgemäßen Verkabelung können während des Betriebs Störungen auf dem CAN-Bus auftreten, die dazu führen, dass der Regler aus Sicherheitsgründen mit einem Fehler abschaltet.

120Ω-Abschlusswiderstand

In den Geräten vom Typ DIS-2 ist kein Abschlusswiderstand integriert.

3.2 Verkabelungs-Hinweise

Der CAN-Bus bietet eine einfache und störungssichere Möglichkeit, alle Komponenten einer Anlage miteinander zu vernetzen. Voraussetzung dafür ist allerdings, dass alle nachfolgenden Hinweise für die Verkabelung beachtet werden.

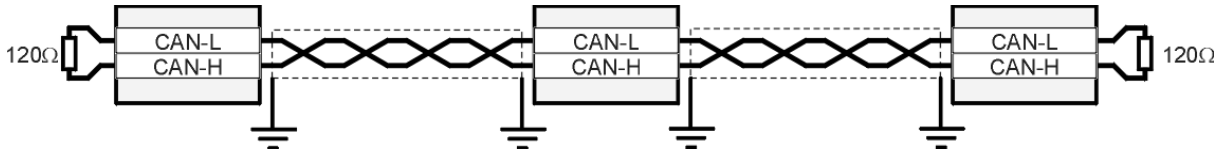


Abbildung 3.2: Verkabelungsbeispiel

- Die einzelnen Knoten des Netzwerkes werden grundsätzlich linienförmig miteinander verbunden, so dass das CAN-Kabel von Regler zu Regler durchgeschleift wird (siehe Abbildung 3.2).
- An beiden Enden des CAN-Kabels muss jeweils genau ein Abschlusswiderstand von 120Ω +/- 5% vorhanden sein. Häufig ist in CAN-Karten oder in einer SPS bereits ein solcher Abschlusswiderstand eingebaut, der entsprechend berücksichtigt werden muss.
- Von der Verwendung von Zwischensteckern bei der CAN-Bus-Verkabelung wird abgeraten. Sollte dies dennoch notwendig sein, ist zu beachten, dass metallische Steckergehäuse verwendet werden, um den Kabelschirm zu verbinden.
- Um die Störeinkopplung so gering wie möglich zu halten, sollten grundsätzlich
  - Motorkabel nicht parallel zu Signalleitungen verlegt werden.
  - Motorkabel gemäß der Spezifikation von Metronix ausgeführt sein.
  - Motorkabel ordnungsgemäß geschirmt und geerdet sein.
- Für weitere Informationen zum Aufbau einer störungsfreien CAN-Bus-Verkabelung verweisen wir auf die Controller Area Network protocol specification, Version 2.0 der Robert Bosch GmbH, 1991.
- Technische Daten CAN-Bus-Kabel:

1 Paar á 2 verdrillten Adern, d ≥ 0,22 mm^2	Schleifenwiderstand < 0,2 Ω/m
Geschirmt	Wellenwiderstand 100-120 Ω

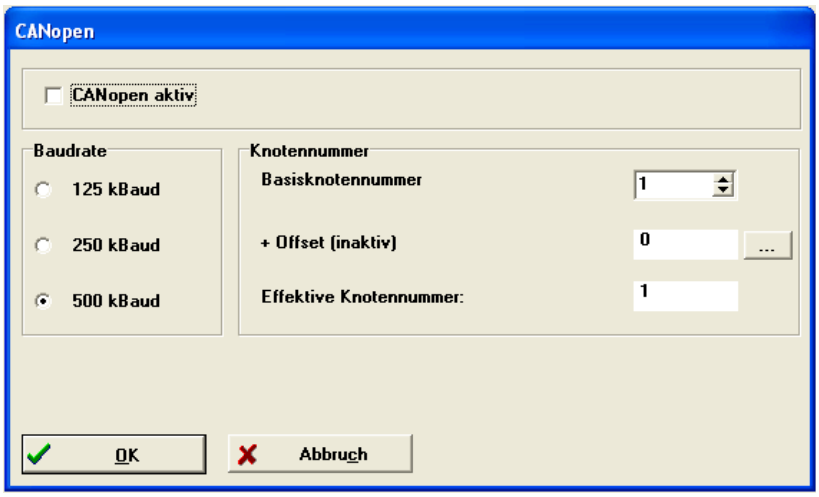
Es wird hierbei kein GND zwischen den einzelnen Knoten mitverbunden, da die Versorgung und Zwischenkreis gleiches Potential benutzen.

Der Schirm des CAN-Kabels wird beidseitig aufgelegt auf Gehäusepotential (PE).

4 Aktivierung von CANopen

4.1 Übersicht

Die Aktivierung des CAN-Interface mit dem Protokoll CANopen erfolgt einmalig über die serielle Schnittstelle des Servoreglers. Das CAN-Protokoll wird über das CANopen-Fenster vom DIS-2 ServoCommander aktiviert.



Es müssen insgesamt 3 Parameter eingestellt werden:

- **Basis-Knotennummer**

  Zur eindeutigen Identifizierung im Netzwerk muss jedem Teilnehmer eine Knotennummer zugeteilt werden, die nur einmal im Netzwerk vorkommen darf. Über diese Knotennummer wird das Gerät adressiert.

  Als zusätzliche Option besteht die Möglichkeit, die Knotennummer des Servopositionierregler von der äußeren Beschaltung abhängig zu machen. Zur Basis-Knotennummer wird einmalig nach dem Reset die Eingangskombination der digitalen Eingänge DIN4 und DIN5 (oder DIN0 … DIN5 je nach Anwahl der Auswertung der AIN / DIN im Menü Digitale Eingänge) addiert.

- **Baudrate**

  Dieser Parameter bestimmt die auf dem CAN-Bus verwendete Baudrate in kBaud.
  Beachten Sie, dass hohe Baudraten eine niedrige maximale Kabellänge erfordern.

Kabellänge	Baudrate (kBaud)
< 100 m	500
< 250 m	250
< 500 m	125

- **CANopen aktivieren**

  Durch Aktivierung der Kommunikation in diesem Feld wird die CAN Kommunikation aktiviert und die CAN Kommunikationsparameter können bis zum Deaktivieren nicht mehr geändert werden.

> Beachten Sie, dass die Parametrierung der CANopen-Funktionalität nach einem Reset nur erhalten bleibt, wenn der Parametersatz des Servopositionierreglers gesichert wurde.

5 Zugriffsverfahren

5.1 Einleitung

CANopen stellt eine einfache und standardisierte Möglichkeit bereit, auf die Parameter des Servoreglers (z.B. den maximalen Motorstrom) zuzugreifen. Dazu ist jedem Parameter (*CAN-Objekt*) eine eindeutige Nummer (*Index und Subindex*) zugeordnet. Die Gesamtheit aller einstellbaren Parameter wird als *Objektverzeichnis* bezeichnet.

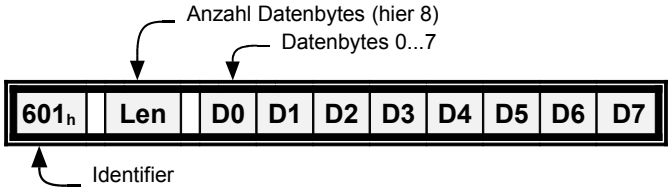
Für den Zugriff auf die CAN-Objekte über den CAN-Bus sind im Wesentlichen zwei Methoden verfügbar: Eine bestätigte Zugriffsart, bei der der Regler jeden Parameterzugriff quittiert (über sog. SDOs) und eine unbestätigte Zugriffsart, bei der keine Quittierung erfolgt (über sog. PDOs).

In der Regel wird der Regler über SDO-Zugriffe sowohl parametriert als auch gesteuert. Für spezielle Anwendungsfälle sind darüber hinaus noch weitere Arten von Nachrichten (sog. Kommunikations-Objekte) definiert, die entweder vom Regler oder der übergeordneten Steuerung gesendet werden:

SDO	**S**ervice **D**ata **O**bject	Werden zur normalen Parametrierung des Reglers verwendet.
PDO	**P**rocess **D**ata **O**bject	Schneller Austausch von Prozessdaten (z.B. Istdrehzahl) möglich.
SYNC	**Sync**hronization Message	Synchronisierung mehrerer CAN-Knoten
EMCY	**Em**ergen**cy** Message	Übermittlung von Fehlermeldungen.
NMT	**N**etwork **M**anagemen**t**	Netzwerkdienst: Es kann z.B. auf alle CAN- Knoten gleichzeitig eingewirkt werden.
HEARTBEAT	Error Control Protocol	Überwachung der Kommunikationsteilnehmer durch regelmäßige Nachrichten.

Jede Nachricht, die auf dem CAN-Bus verschickt wird, enthält eine Art Adresse, mit dessen Hilfe festgestellt werden kann, für welchen Bus-Teilnehmer die Nachricht gedacht ist. Diese Nummer wird als *Identifier* bezeichnet. Je niedriger der Identifier, desto größer ist die Priorität der Nachricht. Für die oben genannten Kommunikationsobjekte sind jeweils Identifier festgelegt.

Die folgende Skizze zeigt den prinzipiellen Aufbau einer CANopen-Nachricht:



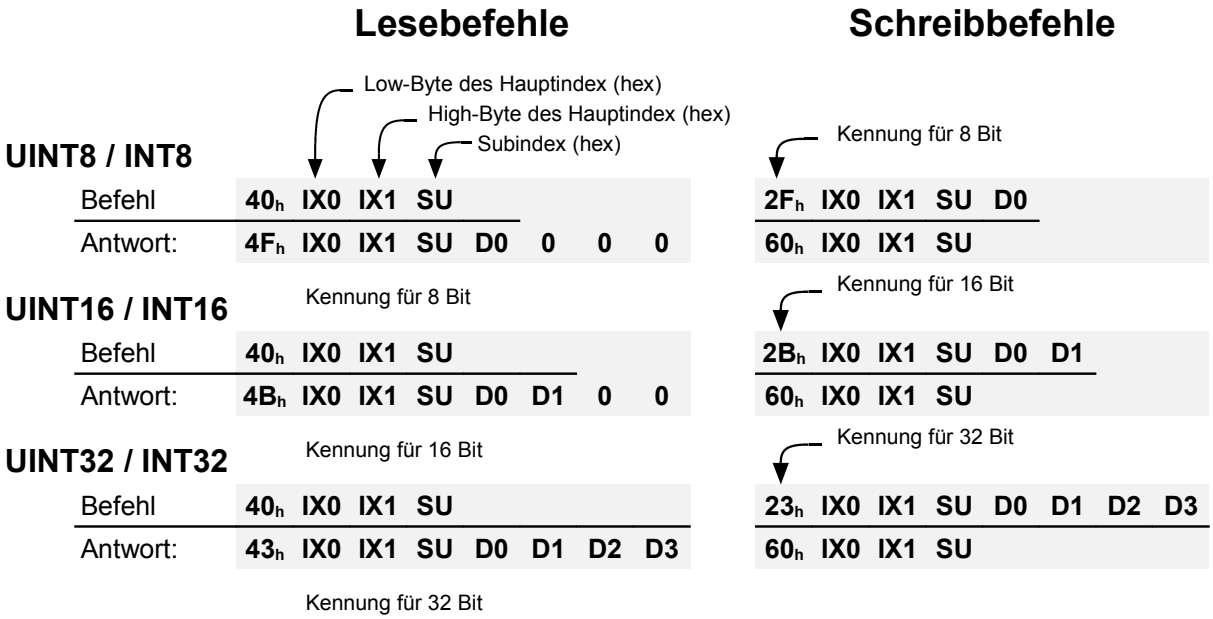
5.2 SDO-Zugriff

Über die **S**ervice-**D**ata-**O**bjekte (SDO) kann auf das Objektverzeichnis des Reglers zugegriffen werden. Dieser Zugriff ist besonders einfach und übersichtlich. Es wird daher empfohlen, die Applikation zunächst nur mit SDOs aufzubauen und erst später einige Objektzugriffe auf die zwar schnelleren, aber auch komplizierteren **P**rocess-**D**ata-**O**bjekte (PDOs) umzustellen.
SDO-Zugriffe gehen immer von der übergeordneten Steuerung (Host) aus. Diese sendet an den Regler entweder einen Schreibbefehl, um einen Parameter des Objektverzeichnisses zu ändern, oder einen Lesebefehl, um einen Parameter auszulesen. Zu jedem Befehl erhält der Host eine Antwort, die entweder den ausgelesenen Wert enthält oder – im Falle eines Schreibbefehls – als Quittung dient.
Damit der Regler erkennt, dass der Befehl für ihn bestimmt ist, muss der Host den Befehl mit einem bestimmten Identifier senden. **Dieser setzt sich aus der Basis 600_h + Knotennummer des betreffenden Reglers zusammen. Der Regler antwortet entsprechend mit dem Identifier 580_h + Knotennummer.**
Der Aufbau der Befehle bzw. der Antworten hängt vom Datentyp des zu lesenden oder schreibenden Objekts ab, da entweder 1, 2 oder 4 Datenbytes gesendet bzw. empfangen werden müssen. Folgende Datentypen werden unterstützt :

UINT8	8-Bit-Wert ohne Vorzeichen	0 ... 255
INT8	8-Bit-Wert mit Vorzeichen	-128 ... 127
UINT16	16-Bit-Wert ohne Vorzeichen	0 ... 65535
INT16	16-Bit-Wert mit Vorzeichen	-32768 ... 32767
UINT32	32-Bit-Wert ohne Vorzeichen	0 ... $(2^{32}-1)$
INT32	32-Bit-Wert mit Vorzeichen	$-(2^{31})$ ... $(2^{31}-1)$

5.2.1 SDO-Sequenzen zum Lesen und Schreiben

Um Objekte dieser Zahlentypen auszulesen oder zu beschreiben sind die nachfolgend aufgeführten Sequenzen zu verwenden. Die Kommandos, um einen Wert in den Regler zu schreiben, beginnen je nach Datentyp mit einer *unterschiedlichen* Kennung. Die Antwort-Kennung ist hingegen stets die gleiche. Lesebefehle beginnen immer mit der gleichen Kennung und der Regler antwortet je nach zurückgegebenem Datentyp unterschiedlich. Alle Zahlen sind in hexadezimaler Schreibweise gehalten.



BEISPIEL

	Lesen von Obj. 6061_00h Rückgabe-Daten: 01h	Schreiben von Obj. 1401_02h Daten: EFh
UINT8 / INT8		
Befehl	40h 61h 60h 00h	2Fh 01h 14h 02h EFh
Antwort:	4Fh 61h 60h 00h 01h	60h 01h 14h 02h
	Lesen von Obj. 6041_00h Rückgabe-Daten: 1234h	Schreiben von Obj. 6040_00h Daten: 03E8h
UINT16 / INT16		
Befehl	40h 41h 60h 00h	2Bh 40h 60h 00h E8h 03h
Antwort:	4Bh 41h 60h 00h 34h 12h	60h 40h 60h 00h
	Lesen von Obj. 6093_01h Rückgabe-Daten: 12345678h	Schreiben von Obj. 6093_01h Daten: 12345678h
UINT32 / INT32		
Befehl	40h 93h 60h 01h	23h 93h 60h 01h 78h 56h 34h 12h
Antwort:	43h 93h 60h 01h 78h 56h 34h 12h	60h 93h 60h 01h

Die Quittierung vom Regler muss in jedem Fall abgewartet werden !
Erst wenn der Regler die Anforderung quittiert hat, dürfen weitere Anforderungen gesendet werden.

5.2.2 SDO-Fehlermeldungen

Im Falle eines Fehlers beim Lesen oder Schreiben (z.B. weil der geschriebene Wert zu groß ist), antwortet der Regler mit einer Fehlermeldung anstelle der Quittierung:

Befehl	...	IX0	IX1	SU	...	...	...	...
Antwort:	**80_h**	IX0	IX1	SU	F0	F1	F2	F3

Fehler-Kennung — Fehlercode (4 Byte)

Fehlercode F3 F2 F1 F0	Bedeutung
06 01 00 00_h	Zugriffsart wird nicht unterstützt.
06 02 00 00_h	Das angesprochene Objekt existiert nicht im Objektverzeichnis
06 04 00 41_h	Das Objekt darf nicht in ein PDO eingetragen werden
06 04 00 42_h	Die Länge der in das PDO eingetragenen Objekte überschreitet die PDO-Länge
06 07 00 10_h	Protokollfehler: Länge des Service-Parameters stimmt nicht überein
06 07 00 12_h	Protokollfehler: Länge des Service-Parameters zu groß
06 07 00 13_h	Protokollfehler: Länge des Service-Parameters zu klein
06 09 00 11_h	Der angesprochene Subindex existiert nicht
06 01 00 01_h	Lesezugriff auf ein Objekt, dass nur geschrieben werden kann
06 01 00 02_h	Schreibzugriff auf ein Objekt, dass nur gelesen werden kann
06 09 00 30_h	Die Daten überschreiten den Wertebereich des Objekts
06 09 00 31_h	Die Daten sind zu groß für das Objekt
06 09 00 32_h	Die Daten sind zu klein für das Objekt
08 00 00 20_h	Daten können nicht übertragen oder gespeichert werden *1)
08 00 00 21_h	Daten können nicht übertragen oder gespeichert werden, da der Regler lokal arbeitet
08 00 00 22_h	Daten können nicht übertragen oder gespeichert werden, da sich der Regler dafür nicht im richtigen Zustand befindet *2)

*1) Werden gemäß DS301 bei fehlerhaftem Zugriff auf store_parameters / restore_parameters zurückgegeben.

*2) „Zustand“ ist hier allgemein zu verstehen: Es kann sich dabei um die falsche Betriebsart handeln. Ein andere Beispiel ist, wenn bei aktiviertem PDO die Anzahl der zu mappenden Objekte beschrieben wird.

5.3 PDO-Message

Mit **P**rocess-**D**ata-**O**bjekten (PDOs) können Daten ereignisgesteuert übertragen werden. Das PDO überträgt dabei einen oder mehrere vorher festgelegte Parameter. Anders als bei einem SDO erfolgt bei der Übertragung eines PDOs keine Quittierung. Nach der PDO-Aktivierung müssen daher alle Empfänger jederzeit eventuell ankommende PDOs verarbeiten können. Dies bedeutet meistens einen erheblichen Softwareaufwand im Host-Rechner. Diesem Nachteil steht der Vorteil gegenüber, dass der Host-Rechner die durch ein PDO übertragenen Parameter nicht zyklisch abzufragen braucht, was zu einer starken Verminderung der CAN-Busauslastung führt.

BEISPIEL

Der Host-Rechner möchte wissen, wann der Regler eine Positionierung von A nach B abgeschlossen hat.

Bei der Verwendung von SDOs muss er hierzu ständig, beispielsweise jede Millisekunde, das Objekt **statusword** abfragen, womit er die Buskapazität stark auslastet.

Bei der Verwendung eines PDOs wird der Regler schon beim Start der Applikation so parametriert, dass er bei jeder Veränderung des Objektes **statusword** ein PDO absetzt, in dem das Objekt **statusword** enthalten ist.

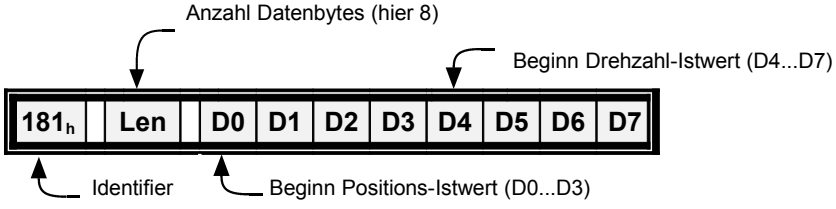
Statt ständig nachzufragen, wird dem Host-Rechner somit automatisch eine entsprechende Meldung zugestellt, sobald das Ereignis eingetreten ist.

Folgende Typen von PDOs werden unterschieden:

Transmit-PDO (T-PDO)	Regler ⇨ Host	Regler sendet PDO bei Auftreten eines bestimmten Ereignisses
Receive-PDO (R-PDO)	Host ⇨ Regler	Regler wertet PDO bei Auftreten eines bestimmen Ereignisses aus

Der Regler verfügt über zwei Transmit- und zwei Receive-PDOs.

In die PDOs können nahezu alle Objekte des Objektverzeichnisses eingetragen (gemappt) werden, d.h. das PDO enthält als Daten z.B. den Drehzahl-Istwert, den Positions-Istwert o.ä. Welche Daten übertragen werden, muss dem Regler vorher mitgeteilt werden, da das PDO lediglich Nutzdaten und keine Information über die Art des Parameters enthält. In dem unteren Beispiel würde in den Datenbytes 0...3 des PDOs der Positions-Istwert und in den Bytes 4...7 der Drehzahl-Istwert übertragen.



Auf diese Art können nahezu beliebige Datentelegramme definiert werden. Die folgenden Kapitel beschreiben die dazu nötigen Einstellungen.

5.3.1 Beschreibung der Objekte

Identifier des PDOs

COB_ID_used_by_PDO

In dem Objekt **COB_ID_used_by_PDO** ist der Identifier einzutragen, auf dem das jeweilige PDO gesendet bzw. empfangen werden soll. Ist **Bit 31** gesetzt, ist das jeweilige PDO deaktiviert. Dies ist die Voreinstellung für alle PDOs. Das **Bit 30** muss immer gesetzt sein da kein Remote Transmit untersützt wird.
Die COB-ID darf nur geändert werden, wenn das PDO deaktiviert, d.h. Bit 31 gesetzt ist. Zur Änderung der COB-ID ist daher folgender Ablauf einzuhalten:

- Auslesen der COB-ID
- Schreiben der ausgelesenen COB-ID + $C0000000_h$
- Schreiben der neuen COB-ID + $C0000000_h$
- Schreiben der neuen COB-ID + 40000000_h, das PDO ist wieder aktiv.

Anzahl zu übertragender Objekte

number_of_mapped_objects

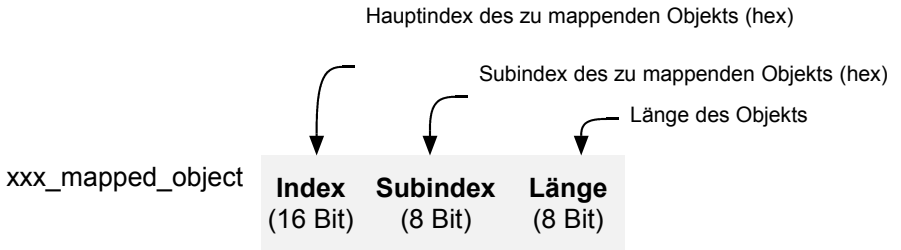
Dieses Objekt gibt an, wie viele Objekte in das entsprechende PDO gemappt werden sollen. Folgende Einschränkungen sind zu beachten:

- Es können pro PDO maximal 4 Objekte gemappt werden
- Ein PDO darf über maximal 64 Bit (8 Byte) verfügen.

Zu übertragende Objekte

first_mapped_object ... fourth_mapped_object

Für jedes Objekt, das im PDO enthalten sein soll muss dem Regler der entsprechende Index, der Subindex und die Länge mitgeteilt werden. Die Längenangabe muss mit der Längenangabe im Object Dictionary übereinstimmen. Teile eines Objekts können nicht gemappt werden.
Die Mapping-Informationen besitzen folgendes Format:



Zur Vereinfachung des Mappings ist folgendes Vorgehen vorgeschrieben:

1.) Die Anzahl der gemappten Objekte wird auf 0 gesetzt.
2.) Die Parameter first_mapped_object...fourth_mapped_object dürfen beschrieben werden (Die Gesamtlänge aller Objekte ist in dieser Zeit nicht relevant).
3.) Die Anzahl der gemappten Objekte wird auf einen Wert zwischen 1...4 gesetzt. Die Länge all dieser Objekte darf jetzt 64 Bit nicht überschreiten

.

Übertragungsart

transmission_type und **inhibit_time**

Für jedes PDO kann festgelegt werden, welches Ereignis zum Aussenden (Transmit-PDO) bzw. Auswerten (Receive-PDO) einer Nachricht führt:

Wert	Bedeutung	Erlaubt bei
00_h –$F0_h$	**SYNC-Message** Der Zahlenwert gibt an, wie viel SYNC-Messages zwischen zwei Aussendungen ignoriert werden, bevor das PDO - gesendet (T-PDO) bzw. - ausgewertet (R-PDO) wird.	TPDOs RPDOs
FE_h	**Zyklisch** Das Transfer-PDO wird vom Regler zyklisch aktualisiert und gesendet. Die Zeitspanne wird durch das Objekt **inhibit_time** festgelegt. Receive-PDOs werden hingegen unmittelbar nach Empfang ausgewertet.	TPDOs (RPDOs)
FF_h	**Änderung** Das Transfer-PDO wird gesendet, wenn sich in den Daten des PDOs mindestens 1 Bit geändert hat. Mit **inhibit_time** kann zusätzlich der minimale Abstand zwischen dem Absenden zweier PDOs in 100µs-Schritten festgelegt werden. Das Receive-PDO wird sofort ausgewertet.	TPDOs RPDOs

Die Verwendung aller anderen Werte ist nicht zulässig.

Maskierung

transmit_mask_high und **transmit_mask_low**

Wird als **transmission_type** „Änderung“ gewählt, wird das TPDO immer gesendet, wenn sich mindestens 1 Bit des TPDOs ändert. Häufig wird es aber benötigt, dass das TPDO nur gesendet wird, wenn sich bestimmte Bits geändert haben. Daher kann das TPDO mit einer Maske versehen werden: Nur die Bits des TPDOs, die in der Maske auf „1“ gesetzt sind, werden zur Auswertung, ob sich das PDO geändert hat herangezogen. Da diese Funktion herstellerspezifisch ist, sind als Defaultwert alle Bits der Masken gesetzt.

BEISPIEL

Folgende Objekte sollen zusammen in einem PDO übertragen werden:

Name des Objekts	Index_Subindex	Bedeutung
statusword	$6041_h\_00_h$	Reglersteuerung
modes_of_operation_display	$6061_h\_00_h$	Betriebsart
digital_inputs	$60FD_h\_00_h$	Digitale Eingänge

Es soll das erste Transmit-PDO (TPDO 1) verwendet werden, welches immer gesendet werden soll, wenn sich eines der digitalen Eingänge ändert, allerdings maximal alle 10 ms. Als Identifier für dieses PDO soll 187_h verwendet werden.

1.) Anzahl der Objekte löschen

Damit das Objektmapping geändert werden darf, Anzahl der Objekte auf Null setzen. ⇒ **number_of_mapped_objects = 0**

2.) Objekte, die gemappt werden sollen, parametrieren

Die oben aufgeführten Objekte müssen jeweils zu einem 32 Bit-Wert zusammengesetzt werden:

Index $=6041_h$ Subin. $= 00_h$ Länge $= 10_h$ ⇒ **first_mapped_object = 60410010_h**

Index $=6061_h$ Subin. $= 00_h$ Länge $= 08_h$ ⇒ **second_mapped_object = 60610008_h**

Index $=60FD_h$ Subin. $= 00_h$ Länge $= 20_h$ ⇒ **third_mapped_object = $60FD0020_h$**

3.) Anzahl der Objekte parametrieren

Es sollen 3 Objekte im PDO enthalten sein ⇒ **number_of_mapped_objects = 3_h**

4.) Übertragungsart parametrieren

Das PDO soll bei Änderung (der oberen digitalen Eingänge) gesendet werden. ⇒ **transmission_type = FF_h**

Damit nur die Änderung der digitalen Eingänge zum Senden führt, wird das PDO maskiert, so dass nur die oberen 16 Bits des Objekts $60FD_h$ „durchkommen".

⇒ **transmit_mask_high = $00FFFF00_h$**

⇒ **transmit_mask_low = 00000000_h**

Das PDO soll höchstens alle 10 ms (100×100µs) gesendet werden. ⇒ **inhibit_time = 64_h**

5.) Identifier parametrieren

Das PDO soll mit Identifier 187h gesendet werden.
Falls das PDO aktiv ist, muss es zuerst deaktiviert werden.

Auslesen des Identifiers: ⇒ **40000181_h = cob_id_used_by_pdo**

Setzen von Bit 31 (deaktivieren): ⇒ **cob_id_used_by_pdo = $C0000181_h$**

Neuen Identifier schreiben: ⇒ **cob_id_used_by_pdo = $C0000187_h$**

Aktivieren durch Löschen von Bit 31: ⇒ **cob_id_used_by_pdo = 40000187_h**

PDO-Parametrierung

Beachten Sie, dass die Parametrierung der PDOs generell nur geändert werden darf, wenn der Netzwerkstatus (NMT) nicht **operational** ist. Siehe hierzu auch Kapitel 5.3.3

5.3.2 Objekte zur PDO-Parametrierung

In den Reglern sind insgesamt 2 Transmit und 2 Receive-PDOs verfügbar. Die einzelnen Objekte, um diese PDOs zu parametrieren sind jeweils für alle 2 TPDOs und alle 2 RPDOs gleich. Daher ist im Folgenden nur die Parameterbeschreibung des ersten TPDOs explizit aufgeführt. Sie ist sinngemäß auch für die anderen PDOs zu verwenden, die im Anschluss tabellarisch aufgeführt sind:

Index	**1800_h**
Name	**transmit_pdo_parameter_tpdo1**
Object Code	RECORD
No. of Elements	3

Sub-Index	**01_h**
Description	**cob_id_used_by_pdo_tpdo1**
Data Type	UINT32
Access	rw
PDO Mapping	no
Units	-
Value Range	181_h...$1FF_h$, Bit 31 darf gesetzt sein, Bit 30 muss gesetzt sein, da **kein remote transmit** unterstützt wird
Default Value	$C0000181_h$

Sub-Index	**02_h**
Description	**transmission_type_tpdo1**
Data Type	UINT8
Access	rw
PDO Mapping	no
Units	-
Value Range	0...$F0_h$, FE_h, FF_h
Default Value	FF_h

Sub-Index	**03_h**
Description	**inhibit_time_tpdo1**
Data Type	UINT16
Access	rw
PDO Mapping	no
Units	100µs (i.e. 10 = 1ms)
Value Range	0..$FFFF_h$,
Default Value	0

Index	**1A00$_h$**
Name	**transmit_pdo_mapping_tpdo1**
Object Code	RECORD
No. of Elements	4

Sub-Index	**00$_h$**
Description	**number_of_mapped_objects_tpdo1**
Data Type	UINT8
Access	rw
PDO Mapping	no
Units	--
Value Range	0...4
Default Value	siehe Tabelle

Sub-Index	**01$_h$**
Description	**first_mapped_object_tpdo1**
Data Type	UINT32
Access	rw
PDO Mapping	no
Units	--
Value Range	--
Default Value	siehe Tabelle

Sub-Index	**02$_h$**
Description	**second_mapped_object_tpdo1**
Data Type	UINT32
Access	rw
PDO Mapping	no
Units	--
Value Range	--
Default Value	siehe Tabelle

Sub-Index	**03$_h$**
Description	**third_mapped_object_tpdo1**
Data Type	UINT32
Access	rw
PDO Mapping	no
Units	--
Value Range	--
Default Value	siehe Tabelle

Sub-Index	**04_h**
Description	**fourth_mapped_object_tpdo1**
Data Type	UINT32
Access	rw
PDO Mapping	no
Units	--
Value Range	--
Default Value	siehe Tabelle

1. Transmit-PDO

Index	Comment	Type	Acc.	Default Value
$1800_h\_00_h$	number of entries	UINT8	ro	03_h
$1800_h\_01_h$	COB-ID used by PDO	UINT32	rw	$C0000181_h$
$1800_h\_02_h$	transmission type	UINT8	rw	FF_h
$1800_h\_03_h$	inhibit time (100 µs)	UINT16	rw	0000_h
$1A00_h\_00_h$	number of mapped objects	UINT8	rw	01_h
$1A00_h\_01_h$	first mapped object	UINT32	rw	60410010_h
$1A00_h\_02_h$	second mapped object	UINT32	rw	00000000_h
$1A00_h\_03_h$	third mapped object	UINT32	rw	00000000_h
$1A00_h\_04_h$	fourth mapped object	UINT32	rw	00000000_h

2. Transmit-PDO

Index	Comment	Type	Acc.	Default Value
$1801_h\_00_h$	number of entries	UINT8	ro	03_h
$1801_h\_01_h$	COB-ID used by PDO	UINT32	rw	$C0000281_h$
$1801_h\_02_h$	transmission type	UINT8	rw	FF_h
$1801_h\_03_h$	inhibit time (100 µs)	UINT16	rw	0000_h
$1A01_h\_00_h$	number of mapped objects	UINT8	rw	02_h
$1A01_h\_01_h$	first mapped object	UINT32	rw	60410010_h
$1A01_h\_02_h$	second mapped object	UINT32	rw	60610008_h
$1A01_h\_03_h$	third mapped object	UINT32	rw	00000000_h
$1A01_h\_04_h$	fourth mapped object	UINT32	rw	00000000_h

tpdo_1_transmit_mask

Index	Comment	Type	Acc.	Default Value
2014_h _00_h	number of entries	UINT8	ro	02_h
2014_h _01_h	tpdo_1_transmit_mask_low	UINT32	rw	$FFFFFFFF_h$
2014_h _02_h	tpdo_1_transmit_mask_high	UINT32	rw	$FFFFFFFF_h$

tpdo_2_transmit_mask

Index	Comment	Type	Acc.	Default Value
2015_h _00_h	number of entries	UINT8	ro	02_h
2015_h _01_h	tpdo_2_transmit_mask_low	UINT32	rw	$FFFFFFFF_h$
2015_h _02_h	tpdo_2_transmit_mask_high	UINT32	rw	$FFFFFFFF_h$

1. Receive PDO

Index	Comment	Type	Acc.	Default Value
1400_h_00_h	number of entries	UINT8	ro	02_h
1400_h_01_h	COB-ID used by PDO	UINT32	rw	$C0000201_h$
1400_h_02_h	transmission type	UINT8	rw	FF_h
1600_h_00_h	number of mapped objects	UINT8	rw	01_h
1600_h_01_h	first mapped object	UINT32	rw	60400010_h
1600_h_02_h	second mapped object	UINT32	rw	00000000_h
1600_h_03_h	third mapped object	UINT32	rw	00000000_h
1600_h_04_h	fourth mapped object	UINT32	rw	00000000_h

2. Receive PDO

Index	Comment	Type	Acc.	Default Value
1401_h_00_h	number of entries	UINT8	ro	02_h
1401_h_01_h	COB-ID used by PDO	UINT32	rw	$C0000301_h$
1401_h_02_h	transmission type	UINT8	rw	FF_h
1601_h_00_h	number of mapped objects	UINT8	rw	02_h
1601_h_01_h	first mapped object	UINT32	rw	60400010_h
1601_h_02_h	second mapped object	UINT32	rw	60600008_h
1601_h_03_h	third mapped object	UINT32	rw	00000000_h
1601_h_04_h	fourth mapped object	UINT32	rw	00000000_h

5.3.3 Aktivierung der PDOs

Damit der Regler PDOs sendet oder empfängt müssen folgende Punkte erfüllt sein:

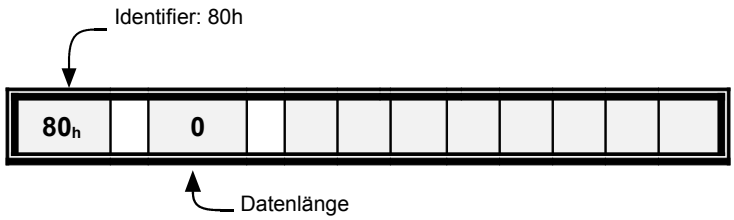
- Das Objekt **number_of_mapped_objects** muß ungleich Null sein.
- Im Objekt **cob_id_used_by_pdos** muss das Bit 31 gelöscht sein.
- Der Kommunikationsstatus des Reglers muss **operational** sein (siehe Kapitel 5.7, Netzwerkmanagement: NMT-Service)

Damit PDOs parametriert werden können, müssen folgende Punkte erfüllt sein:

- Der Kommunikationsstatus des Reglers darf nicht **operational** sein.

5.4 SYNC-Message

Mehrere Geräte einer Anlage können miteinander synchronisiert werden. Hierzu sendet eines der Geräte (meistens die übergeordnete Steuerung) periodisch Synchronisations-Nachrichten aus. Alle angeschlossenen Regler empfangen diese Nachrichten und verwenden sie für die Behandlung der PDOs (siehe Kapitel 5.3).



Der Identifier, auf dem der Regler die SYNC-Message empfängt, ist fest auf 080_h eingestellt. Der Identifier kann über das Objekt **cob_id_sync** ausgelesen werden.

Index	**1005_h**
Name	**cob_id_sync**
Object Code	VAR
Data Type	UINT32

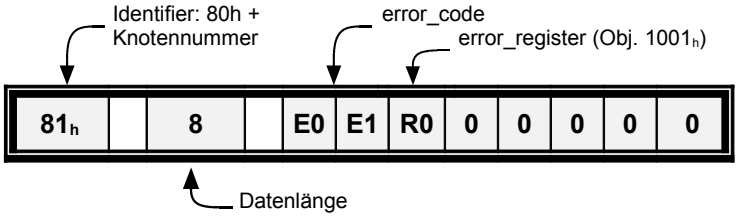
Access	rw
PDO Mapping	no
Units	
Value Range	80000080_h, 00000080_h
Default Value	00000080_h

5.5 EMERGENCY-Message

Der Antriebsregler überwacht die Funktion seiner wesentlichen Baugruppen. Hierzu zählen die Spannungsversorgung, die Endstufe, die Winkelgeberauswertung und die Technologiesteckplätze. Außerdem werden laufend der Motor (Temperatur, Winkelgeber) und die Endschalter überprüft. Auch Fehlparametrierungen können zu Fehlermeldungen führen (Division durch Null etc.).

5.5.1 Aufbau der EMERGENCY-Message

Der Regler sendet beim Auftreten eines Fehlers eine EMERGENCY-Message. Der Identifier dieser Nachricht wird aus dem Identifier **80_h** und der **Knotennummer** des betroffenen Reglers zusammengesetzt.
Die EMERGENCY-Message besteht aus acht Datenbytes, wobei in den ersten beiden Bytes ein **error_code** steht, die in folgender Tabelle aufgeführt sind. Im dritten Byte steht ein weiterer Fehlercode (Objekt 1001_h). Die restlichen fünf Bytes enthalten Nullen.



Folgende Fehlercodes können auftreten:

error_code (hex)	Anzeige	Bedeutung
3	4310	Übertemperaur im Motor
4	4210	Unter-/Übertemperatur Leistungselektronik
5	7392	Fehler SINCOS-Versorgung
6	7391	Fehler SINCOS-RS485-Kommunikation
7	7390	Fehler SINCOS-Spursignale
8	7380	Fehler Resolverspursignale / Trägerausfall
9	5113	Fehler 5V-Elektronikversorgung
10	5114	Fehler 12V-Elektronikversorgung
11	5112	Fehler 24V-Versorgung (out of range)
13	5210	Fehler Offset Strommessung
14	2320	Überstrom Zwischenkreis / Endstufe
15	3220	Unterspannung Zwischenkreis
16	3210	Überspannung Zwischenkreis
19	2312	I^2t-Fehler Motor (I^2t bei 100%)
20	2311	I^2t-Fehler Regler (I^2t bei 100%)
26	2380	I^2t bei 80%
27	4380	Temperatur Motor 5°C unter Maximum
28	4280	Temperatur Endstufe 5°C unter Maximum
29	8611	Schleppfehler Überwachung

error_code (hex)	Anzeige	Bedeutung
31	8612	Fehler Endschalter
35	6199	Timeout bei Schnellhalt
36	8A80	Fehler Referenzfahrt
40	6197	Fehler: Motor- und Winkelgeber-Identifikation
43	6193	Wegprogramm: unbekannter Befehl
44	6192	Wegprogramm: ungültiges Sprungziel
56	7510	Fehler RS232-Kommunikation
57	6191	Fehler Positionsdatensatz
58	6380	Fehler Betriebsart
60	6190	Fehler in Vorberechnung Positionierung
62	6180	Stack-Overflow
63	5581	Checksummenfehler
64	6187	Initialisierungsfehler

5.5.2 Beschreibung der Objekte

5.5.2.1 Objekt 1003_h: pre_defined_error_field

Der jeweilige **error_code** der Fehlermeldungen wird zusätzlich in einem vierstufigen Fehlerspeicher abgelegt. Dieser ist wie ein Schieberegister strukturiert, so dass immer der zuletzt aufgetretene Fehler im Objekt **1003_h_01_h** (**standard_error_field_0**) abgelegt ist. Durch einen Lesezugriff auf das Objekt **1003_h_00_h** (**pre_defined_error_field**) kann festgestellt werden, wie viele Fehlermeldungen zur Zeit im Fehlerspeicher abgelegt sind. Der Fehlerspeicher wird durch das Einschreiben des Wertes 00_h in das Objekt **1003_h_00_h** (**pre_defined_error_field**) gelöscht. Um nach einem Fehler die Endstufe des Reglers wieder aktivieren zu können, muss zusätzlich eine **Fehlerquittierung** (siehe Kapitel 7.1: Zustandsänderung 15) durchgeführt werden.

Index	**1003_h**
Name	**pre_defined_error_field**
Object Code	ARRAY
No. of Elements	4
Data Type	UINT32

Sub-Index	**00_h**
Description	**pre_defined_error_field**
Data Type	UINT8
Access	rw
PDO Mapping	no
Units	--
Value Range	0 (write acces): Fehlerpuffer löschen 0..4 (read access): Anzahl der Fehler im Fehlerpuffer

Default Value	--

Sub-Index	**01$_h$**
Description	**standard_error_field_0**
Access	ro
PDO Mapping	no
Units	--
Value Range	--
Default Value	--

Sub-Index	**02$_h$**
Description	**standard_error_field_1**
Access	ro
PDO Mapping	no
Units	--
Value Range	--
Default Value	--

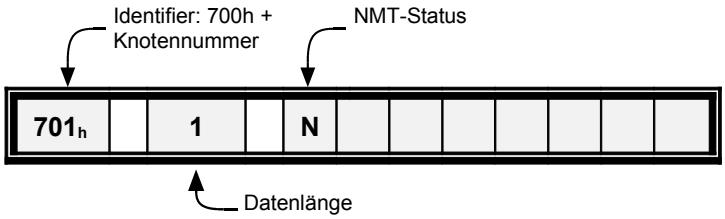
Sub-Index	**03$_h$**
Description	**standard_error_field_2**
Access	ro
PDO Mapping	no
Units	--
Value Range	--
Default Value	--

Sub-Index	**04$_h$**
Description	**standard_error_field_3**
Access	ro
PDO Mapping	no
Units	--
Value Range	--
Default Value	--

5.6 Heartbeat / Bootup (Error Control Protocol)

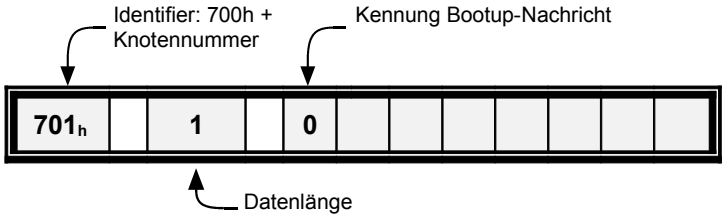
5.6.1 Aufbau der Heartbeat-Nachricht

Zur Überwachung der Kommunikation zwischen Slave (Antrieb) und Master ist das sogenannte Heartbeat-Protokoll implementiert: Hierbei sendet der Antrieb zyklisch Nachrichten an den Master. Der Master kann das zyklische Auftreten dieser Nachrichten überprüfen und entsprechende Maßnahmen einleiten, wenn diese ausbleiben. Das Heartbeat-Telegramm wird mit dem Identifier **700_h + Knotennummer** gesendet. Es enthält nur 1 Byte Nutzdaten, den NMT-Status des Reglers (siehe Kapitel 5.7, Netzwerkmanagement: NMT-Service).



5.6.2 Aufbau der Bootup-Nachricht

Nach dem Einschalten der Spannungsversorgung oder nach einem Reset, meldet der Regler über eine Bootup-Nachricht, dass die Initialisierungsphase beendet ist. Der Regler ist dann im NMT-Status **preoperational** (siehe Kapitel 5.7, Netzwerkmanagement: NMT-Service)



Die Bootup-Nachricht ist nahezu identisch zur Heartbeat-Nachricht aufgebaut. Lediglich wird statt des NMT-Status eine Null gesendet.

5.6.3 Beschreibung der Objekte

5.6.3.1 Objekt 1017_h: producer_heartbeat_time

Die Zeit zwischen zwei Heartbeat-Telegrammen kann über das Object **producer_heartbeat_time** festgelegt werden.

Index	**1017_h**
Name	**producer_heartbeat_time**
Object Code	VAR
Data Type	UINT16

Access	rw
PDO Mapping	no
Units	ms
Value Range	0...65536
Default Value	0

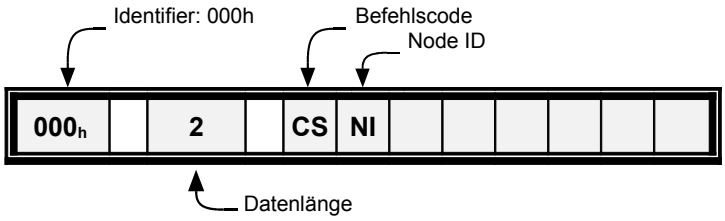
Die **producer_heartbeat_time** kann im Parametersatz gespeichert werden. Startet der Regler mit einer **producer_heartbeat_time** ungleich Null, gilt die Bootup-Nachricht als erstes Heartbeat.

5.7 Netzwerkmanagement (NMT-Service)

Alle CANopen-Geräte können über das Netzwerkmanagement angesteuert werden. Hierfür ist der Identifier mit der höchsten Priorität (000_h) reserviert.

Mittels NMT können Befehle an einen oder alle Regler gesendet werden. Jeder Befehl besteht aus zwei Bytes, wobei das erste Byte den Befehlscode (command specifier, cs) und das zweite Byte die Knotenadresse (node id, ni) des angesprochenen Reglers beinhaltet. Über die Knotenadresse Null können gleichzeitig alle im Netzwerk befindlichen Knoten angesprochen werden. Es ist somit möglich, dass z.B. in allen Geräten gleichzeitig ein Reset ausgelöst wird. Die Regler quittieren die NMT-Befehle nicht. Es kann nur indirekt (z.B. durch die Einschaltmeldung nach einem Reset) auf die erfolgreiche Durchführung geschlossen werden.

Aufbau der NMT-Nachricht:



Für den NMT-Status des CANopen-Knotens sind Zustände in einem Zustandsdiagramm festgelegt. Über das Byte **CS** in der NMT-Nachricht können Zustandsänderungen ausgelöst werden. Diese sind im Wesentlichen am Ziel-Zustand orientiert.

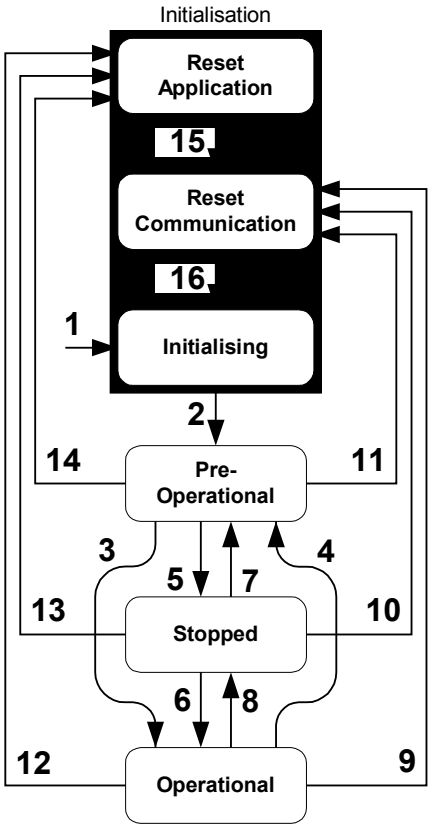


Abbildung 5.3: NMT-State machine

Über folgende Befehle kann der NMT-Status des Reglers beeinflusst werden:

CS	Bedeutung	Übergänge	Ziel-Zustand
01_h	Start Remote Node	3, 6	**Operational**
02_h	Stop Remote Node	5, 8	**Stopped**
80_h	Enter Pre-Operational	4, 7	**Pre-Operational**
81_h	Reset Application	12, 13, 14	**Reset Application**
82_h	Reset Communication	9, 10, 11	**Reset Communication**

Alle anderen Zustands-Übergänge werden vom Regler selbsttätig ausgeführt, z.B. weil die Initialisierung abgeschlossen ist.

Im Parameter **NI** muss die Knotennummer des Reglers angegeben werden oder Null, wenn alle im Netzwerk befindlichen Knoten adressiert werden sollen (Broadcast). Je nach NMT-Status können bestimmte Kommunikationsobjekte nicht benutzt werden: So ist es z.B. unbedingt notwendig den NMT-Status auf **Operational** zu stellen , damit der Regler PDOs sendet.

Name	Bedeutung	SDO	PDO	NMT
Reset Application	Keine Kommunikation. Alle CAN-Objekte werden auf ihre Resetwerte (Applikations-Parametersatz) zurückgesetzt	-	-	-
Reset Communication	Keine Kommunikation Der CAN-Controller wird neu initialisiert.	-	-	-
Initialising	Zustand nach Hardware-Reset. Zurücksetzen des CAN-Knotens, Senden der Bootup-Message	-	-	-
Pre-Operational	Kommunikation über SDOs möglich PDOs nicht aktiv (Kein Senden / Auswerten)	X	-	X
Operational	Kommunikation über SDOs möglich Alle PDOs aktiv (Senden / Auswerten)	X	X	X
Stopped	Keine Kommunikation außer Heartbeating	-	-	X

Der Kommunikationsstatus muss auf **operational** eingestellt werden, damit der Regler PDOs sendet und empfängt.

5.8 Tabelle der Identifier

Die folgende Tabelle gibt eine Übersicht über die verwendeten Identifier:

Objekt-Typ	Identifier (hexadezimal)	Bemerkung
SDO (Host an Regler)	**600_h+Knotennummer**	
SDO (Regler an Host)	**580_h +Knotennummer**	
TPDO1	**181_h**	Standardwerte. Können bei Bedarf geändert werden.
TPDO2	**281_h**	
RPDO1	**201_h**	
RPDO2	**301_h**	
SYNC	**080_h**	
EMCY	**080_h +Knotennummer**	
HEARTBEAT	**700_h +Knotennummer**	
BOOTUP	**700_h +Knotennummer**	
NMT	**000_h**	

6 Parameter einstellen

Bevor der Servoregler die gewünschte Aufgabe (Momenten-, Drehzahlregelung, Positionierung) ausführen kann, müssen zahlreiche Parameter des Reglers an den verwendeten Motor und die spezifische Applikation angepasst werden. Dabei sollte in der Reihenfolge der anschließenden Kapitel vorgegangen werden.

Im Anschluss an die Einstellung der Parameter wird die Gerätesteuerung und die Nutzung der jeweiligen Betriebsarten erläutert.

6.1 Parametersätze laden und speichern

6.1.1 Übersicht

Der Regler verfügt über drei Parametersätze:

- **Aktueller Parametersatz**

  Dieser Parametersatz befindet sich im flüchtigen Speicher (RAM) des Reglers. Er kann mit dem Parametrierprogramm DIS-2 ServoCommander oder über den CAN-Bus beliebig gelesen und beschrieben werden. Beim Einschalten des Reglers wird der **Applikations-Parametersatz** in den **aktuellen Parametersatz** kopiert.

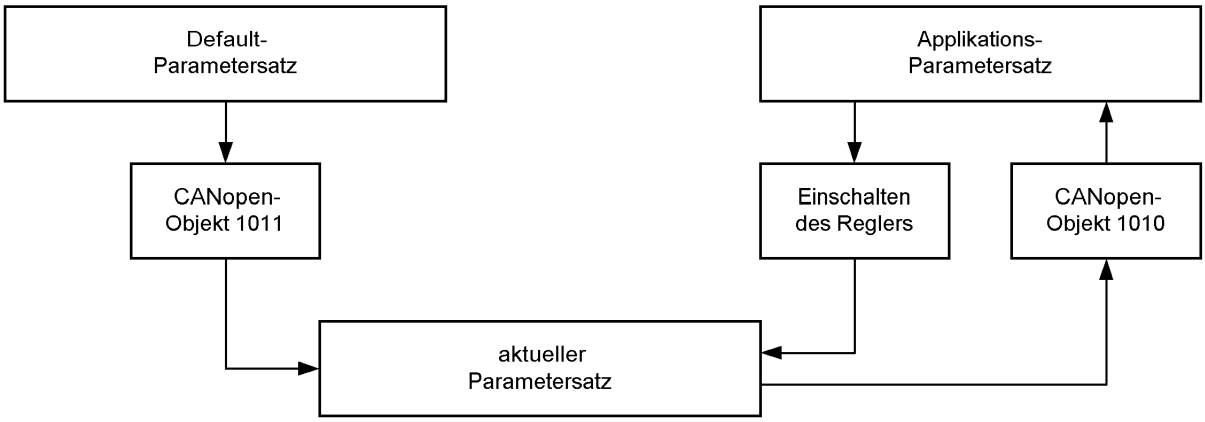
- **Default-Parametersatz**

  Dieses ist der vom Hersteller standardmäßig vorgegebene unveränderliche Parametersatz des Antriebsreglers. Durch einen Schreibvorgang in das CANopen-Objekt **1011_h_01_h** (**restore_all_default_parameters**) kann der **Default-Parametersatz** in den **aktuellen Parametersatz** kopiert werden. Dieser Kopiervorgang ist nur bei ausgeschalteter Endstufe möglich.

- **Applikations-Parametersatz**

  Der **aktuelle Parametersatz** kann in den nichtflüchtigen Flash-Speicher gesichert werden. Der Speichervorgang wird mit einem Schreibzugriff auf das CANopen-Objekt **1010_h_01_h** (**save_all_parameters**) ausgelöst. Beim Einschalten des Reglers wird automatisch der **Applikations-Parametersatz** in den **aktuellen Parametersatz** kopiert.

Die nachfolgende Grafik veranschaulicht die Zusammenhänge zwischen den einzelnen Parametersätzen.



Es sind zwei unterschiedliche Konzepte zur Parametersatzverwaltung denkbar:

1. Der Parametersatz wird mit dem Parametrierprogramm DIS-2 ServoCommander erstellt und ebenfalls mit dem DIS-2 ServoCommander komplett in die einzelnen Regler übertragen. Bei diesem Verfahren müssen nur die ausschließlich via CANopen zugänglichen Objekte über den CAN-Bus eingestellt werden. **Nachteilig ist hierbei, dass für jede Inbetriebnahme einer neuen Maschine oder im Falle einer Reparatur (Regleraustausch) die Parametriersoftware benötigt wird. Dieses Verfahren ist daher nur bei Einzelstücken sinnvoll.**
2. Diese Variante basiert auf der Tatsache, dass die meisten applikationsspezifischen Parametersätze nur in wenigen Parametern vom **Default-Parametersatz** abweichen. Dadurch ist es möglich, dass der **aktuelle Parametersatz** nach jedem Einschalten der Anlage über den CAN-Bus neu aufgebaut wird. Hierzu wird von der übergeordneten Steuerung zunächst der **Default-Parametersatz** geladen (Aufruf des CANopen-Objekts **1011_h_01_h** (**restore_all_default_parameters**). Danach werden nur die abweichenden Objekte übertragen. Der gesamte Vorgang dauert pro Regler unter 1 Sekunde. Vorteilhaft ist, dass dieses Verfahren auch bei unparametrierten Reglern funktioniert, so dass die Inbetriebnahme von neuen Anlagen oder der Austausch einzelner Regler unproblematisch ist und die Parametriersoftware hierfür nicht benötigt wird.

Es wird empfohlen, nach der 2. Variante zu arbeiten. Es ist allerdings darauf zu achten, dass nicht alle Parameter über CAN einstellbar sind. Ist es dennoch erforderlich sonstige Parameter einzustellen muss nach der 1. Variante vorgegangen werden.

Stellen sie vor dem allerersten Einschalten der Endstufe sicher, dass der Regler wirklich die von Ihnen gewünschten Parameter enthält.

Ein falsch parametrierter Regler kann unkontrolliert drehen und Personen- oder Sachschäden verursachen.

6.1.2 Beschreibung der Objekte

6.1.2.1 Objekt 1011_h: restore_default_parameters

Index	**1011_h**
Name	**restore_parameters**
Object Code	ARRAY
No. of Elements	1
Data Type	UINT32

Sub-Index	**01_h**
Description	**restore_all_default_parameters**
Access	rw
PDO Mapping	no
Units	-
Value Range	$64616F6C_h$ („load“)
Default Value	1 (read access)

Das Objekt **1011_h_01_h** (**restore_all_default_parameters**) ermöglicht, den **aktuellen Parametersatz** in einen definierten Zustand zu versetzen. Hierfür wird der **Default-Parametersatz** in den **aktuellen Parametersatz** kopiert. Der Kopiervorgang wird durch einen Schreibzugriff auf dieses Objekt ausgelöst, wobei als Datensatz der String „load“ in hexadezimaler Form zu übergeben ist.
Dieser Befehl wird nur bei deaktivierter Endstufe ausgeführt. Andernfalls wird der SDO-Fehler „Daten können nicht übertragen oder gespeichert werden, da sich der Regler dafür nicht im richtigen Zustand befindet“ erzeugt. Wird die falsche Kennung gesendet, wird der Fehler „Daten können nicht übertragen oder gespeichert werden“ erzeugt. Wird lesend auf das Objekt zugegriffen, wird eine 1 zurückgegeben, um anzuzeigen, dass das Zurücksetzen auf Defaultwerte unterstützt wird.
Die Parameter der CAN-Kommunikation (Knoten-Nr., Baudrate und Betriebsmodus) bleiben unverändert.

6.1.2.2 Objekt 1010_h: store_parameter

Index	**1010_h**
Name	**store_parameters**
Object Code	ARRAY
No. of Elements	1
Data Type	UINT32

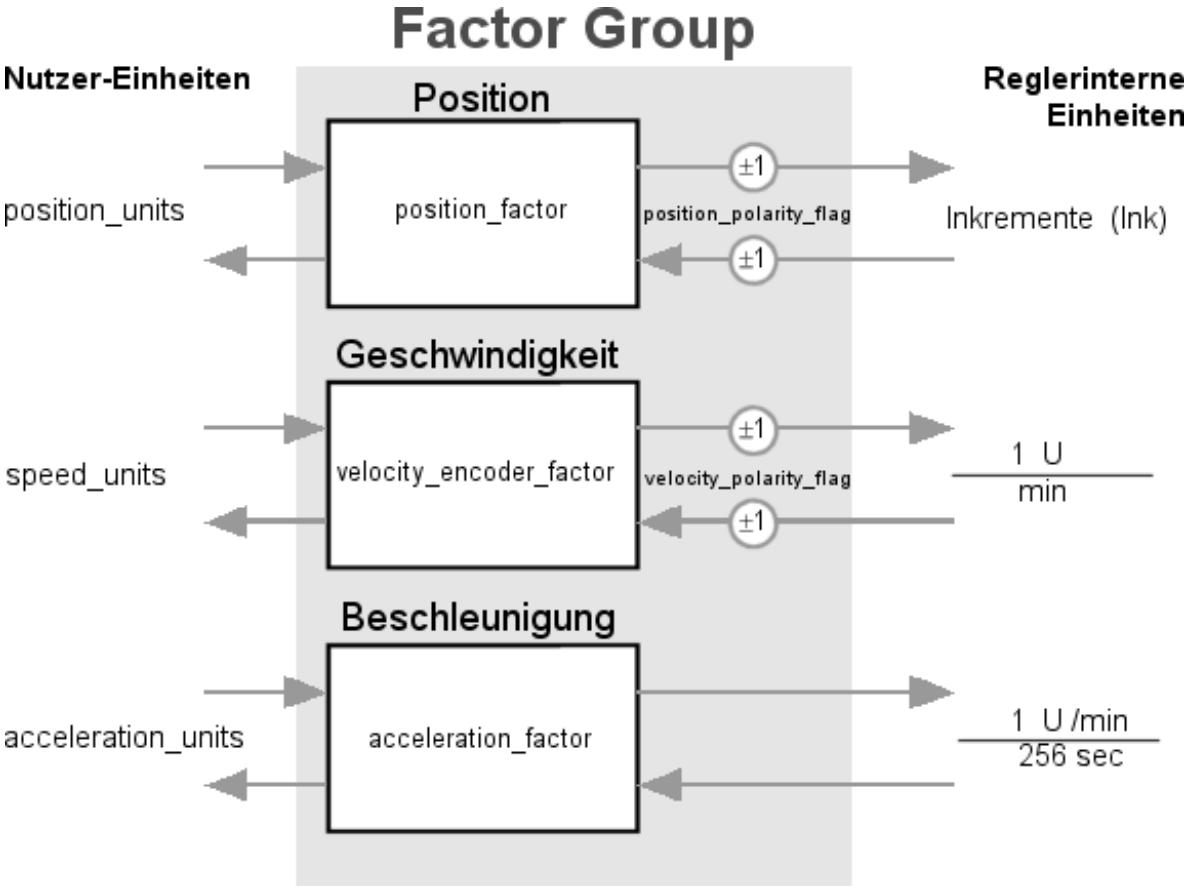
Sub-Index	**01_h**
Description	**save_all_parameters**
Access	rw
PDO Mapping	no
Units	-
Value Range	65766173_h („save“)
Default Value	1 (read access)

Soll der Default-Parametersatz auch in den Applikations-Parametersatz übernommen werden, dann muss außerdem auch das Objekt **1010_h_01_h** (**save_all_parameters**) aufgerufen werden.

6.2 Umrechnungsfaktoren (Factor Group)

6.2.1 Übersicht

Servoregler werden in einer Vielzahl von Anwendungsfällen eingesetzt: Als Direktantrieb, mit nachgeschaltetem Getriebe, für Linearantriebe etc. Um für alle diese Anwendungsfälle eine einfache Parametrierung zu ermöglichen, kann der Regler mit Hilfe der Factor Group so parametriert werden, dass der Nutzer alle Größen wie z.B. die Drehzahl direkt in den gewünschten Einheiten am Abtrieb angeben bzw. auslesen kann (z.B. bei einer Linearachse Positionswerte in Millimeter und Geschwindigkeiten in Millimeter pro Sekunde). Der Regler rechnet die Eingaben dann mit Hilfe der Factor Group in seine internen Einheiten um. Für jede physikalische Größe (Position, Geschwindigkeit und Beschleunigung) ist ein Umrechnungsfaktor vorhanden, um die Nutzer-Einheiten an die eigene Applikation anzupassen. Die durch die Factor Group eingestellten Einheiten werden allgemein als **position_units**, **speed_units** oder **acceleration_units** bezeichnet. Die folgende Skizze verdeutlicht die Funktion der Factor Group:



Alle Parameter werden im Regler grundsätzlich in seinen internen Einheiten gespeichert und erst beim Einschreiben oder Auslesen mit Hilfe der Factor Group umgerechnet.

Daher sollte die Factor Group vor der allerersten Parametrierung eingestellt werden und während einer Parametrierung nicht geändert werden.

Standardmäßig ist die Factor Group auf folgende Einheiten eingestellt:

Größe	Bezeichnung	Einheit	Erklärung
Länge	position_units	**Inkremente**	65536 Inkremente pro Umdrehung
Geschwindigkeit	speed_units	**min^{-1}**	Umdrehungen pro Minute
Beschleunigung	acceleration_units	**$(min^{-1})/256s$**	Drehzahlerhöhung pro 256 Sekunden

6.2.2 Beschreibung der Objekte

6.2.2.1 In diesem Kapitel behandelte Objekte

Index	Objekt	Name	Typ	Attr.
6093_h	ARRAY	position_factor	UINT32	rw
6094_h	ARRAY	velocity_encoder_ factor	UINT32	rw
6097_h	ARRAY	acceleration_factor	UINT32	rw
$607E_h$	VAR	polarity	UINT8	rw

6.2.2.2 Objekt 6093_h: position_factor

Das Objekt **position_factor** dient zur Umrechnung aller Längeneinheiten der Applikation von **position_units** in die interne Einheit **Inkremente** (65536 Inkremente entsprechen 1 Umdrehung). Es besteht aus Zähler und Nenner.

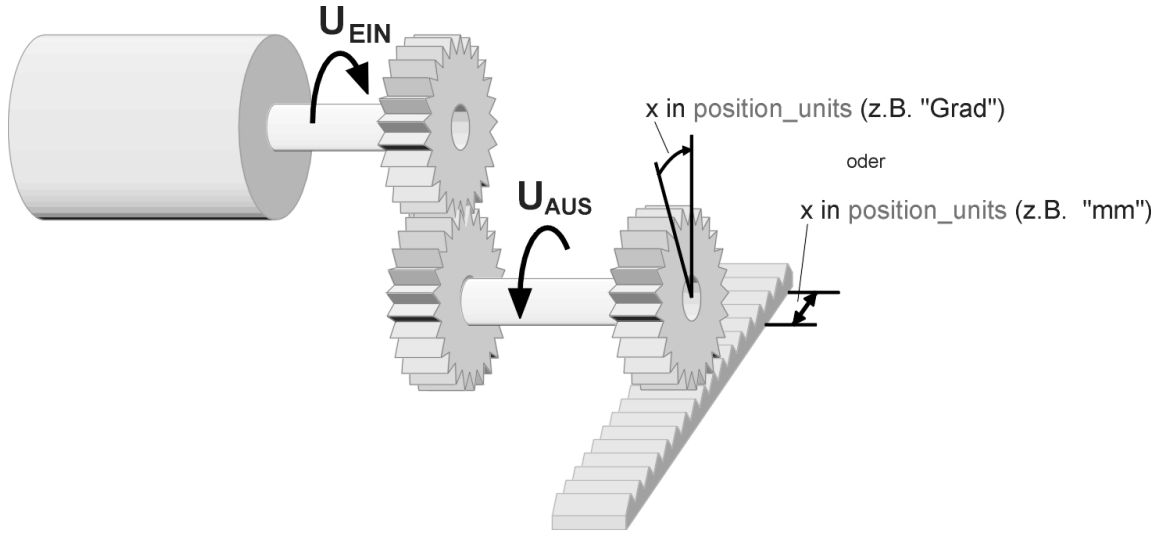


Abbildung 6.4: Übersicht: Factor Group

Index	**6093$_h$**
Name	**position_factor**
Object Code	ARRAY
No. of Elements	2
Data Type	UINT32

Sub-Index	**01$_h$**
Description	**numerator**
Access	rw
PDO Mapping	yes
Units	--
Value Range	--
Default Value	1

Sub-Index	**02$_h$**
Description	**divisor**
Access	rw
PDO Mapping	yes
Units	--
Value Range	--
Default Value	1

In die Berechnungsformel des **position_factor** gehen folgende Größen ein:

gear_ratio Getriebeverhältnis zwischen Umdrehungen am Eintrieb (U_{EIN}) und Umdrehungen am Abtrieb (U_{AUS})

feed_constant Verhältnis zwischen Umdrehungen am Abtrieb (U_{AUS}) und Bewegung in **position_units** (z.B. 1 U = 360° Grad)

Die Berechnung des **position_factors** erfolgt mit folgender Formel:

$$\textbf{position\_factor} = \frac{\textbf{numerator}}{\textbf{divisor}} = \frac{\textbf{65536} \cdot \textbf{gear\_ratio}}{\textbf{feed\_constant}}$$

Der **position_factor** muss getrennt nach Zähler und Nenner in den Regler geschrieben werden. Daher kann es notwendig sein, den Bruch durch geeignete Erweiterung auf ganze Zahlen zu bringen.

BEISPIEL

1. Gewünschte Einheit am Abtrieb (position_units)
2. feed_constant: Wie viel position_units sind 1 Umdrehung (UAUS)
3. Getriebefaktor (gear_ratio): UEIN pro UAUS
4. Werte in Formel einsetzen

1.	2.	3.	4.	ERGEBNIS Gekürzt
Ink.	1 U_{AUS} = **65536** Ink	**1/1**	$\frac{65536 \frac{\text{Ink}}{\text{U}} \cdot \frac{1\text{U}}{1\text{U}} \cdot \frac{1}{1}}{\frac{65536\,\text{Ink}}{1\text{U}}} = \frac{\mathbf{1}\,\text{Ink}}{\mathbf{1}\,\text{Ink}}$	**num:** 1 **div:** 1
1/10 Grad ($^{\circ}/_{10}$)	1 U_{AUS} = **3600** $^{\circ}/_{10}$	**1/1**	$\frac{65536 \frac{\text{Ink}}{\text{U}} \cdot \frac{1\text{U}}{1\text{U}}}{\frac{3600\,^{\circ}/_{10}}{1\text{U}}} = \frac{\mathbf{65536}\,\text{Ink}}{\mathbf{3600}\,^{\circ}/_{10}}$	**num:** 4096 **div:** 225
1/100 Umdr. ($^{U}/_{100}$)	1 U_{AUS} = **100** $^{U}/_{100}$	**1/1**	$\frac{65536 \frac{\text{Ink}}{\text{U}} \cdot \frac{1\text{U}}{1\text{U}}}{\frac{100\,U/_{100}}{1\text{U}}} = \frac{\mathbf{65536}\,\text{Ink}}{\mathbf{100}\,U/_{100}}$	**num:** 16384 **div:** 25
1/100 Umdr. ($^{U}/_{100}$)		**2/3**	$\frac{65536 \frac{\text{Ink}}{\text{U}} \cdot \frac{2\text{U}}{3\text{U}}}{\frac{100\,U/_{100}}{1\text{U}}} = \frac{\mathbf{131072}\,\text{Ink}}{\mathbf{300}\,U/_{100}}$	**num:** 32768 **div:** 75
1/10 mm ($^{mm}/_{10}$)	1 U_{AUS} = **631.5** $^{mm}/_{10}$	**1/1**	$\frac{65536 \frac{\text{Ink}}{\text{U}} \cdot \frac{1\text{U}}{1\text{U}}}{\frac{631.5\,mm/_{10}}{1\text{U}}} = \frac{\mathbf{655360}\,\text{Ink}}{\mathbf{6315}\,mm/_{10}}$	**num:** 131072 **div:** 1263
1/10 mm ($^{mm}/_{10}$)		**4/5**	$\frac{65536 \frac{\text{Ink}}{\text{U}} \cdot \frac{4\text{U}}{5\text{U}}}{\frac{631.5\,mm/_{10}}{1\text{U}}} = \frac{\mathbf{2621440}\,\text{Ink}}{\mathbf{31575}\,mm/_{10}}$	**num:** 524288 **div:** 6315

6.2.2.3 Objekt 6094_h: velocity_encoder_factor

Das Objekt **velocity_encoder_factor** dient zur Umrechnung aller Geschwindigkeitswerte der Applikation von **speed_units** in die interne Einheit **Umdrehungen pro Minute**. Es besteht aus Zähler und Nenner.

Index	**6094_h**
Name	**velocity_encoder_factor**
Object Code	ARRAY
No. of Elements	2
Data Type	UINT32

Sub-Index	**01_h**
Description	**numerator**
Access	rw
PDO Mapping	yes
Units	--
Value Range	--
Default Value	1

Sub-Index	**02_h**
Description	**divisor**
Access	rw
PDO Mapping	yes
Units	--
Value Range	--
Default Value	1

Die Berechnung des **velocity_encoder_factor** setzt sich im Prinzip aus zwei Teilen zusammen: Einem Umrechnungsfaktor von internen Längeneinheiten in **position_units** und einem Umrechnungsfaktor von internen Zeiteinheiten in benutzerdefinierte Zeiteinheiten (z.B. von Sekunden in Minuten). Der erste Teil entspricht der Berechnung des **position_factor** für den zweiten Teil kommt ein zusätzlicher Faktor zur Berechnung hinzu:

time_factor_v Verhältnis zwischen interner Zeiteinheit und benutzerdefinierter Zeiteinheit.

gear_ratio Getriebeverhältnis zwischen Umdrehungen am Eintrieb (U_{EIN}) und Umdrehungen am Abtrieb (U_{AUS})

feed_constant Verhältnis zwischen Umdrehungen am Abtrieb (U_{AUS}) und Bewegung in **position_units** (z.B. 1 U = 360° Grad)

Die Berechnung des **velocity_encoder_factors** erfolgt mit folgender Formel:

$$\mathbf{velocity\_encoder\_factor} = \frac{\mathbf{numerator}}{\mathbf{divisor}} = \frac{\mathbf{gear\_ratio} \cdot \mathbf{time\_factor\_v}}{\mathbf{feed\_constant}}$$

Wie der **position_factor** wird auch der **velocity_encoder_factor** getrennt nach Zähler und Nenner in den Regler geschrieben werden. Daher kann es notwendig sein, den Bruch durch geeignete Erweiterung auf ganze Zahlen zu bringen.

BEISPIEL

1. **Gewünschte Einheit am Abtrieb (*speed_units*)**
2. ***feed_constant*: Wie viel *position_units* sind 1 Umdrehung (U_{AUS})?**
3. ***time_factor_v*: Gewünschte Zeiteinheit pro interner Zeiteinheit**
4. **Getriebefaktor (*gear_ratio*) U_{EIN} pro U_{AUS}**
5. **Werte in Formel einsetzen**

1.	2.	3.	4.	5.	ERGEBNIS Gekürzt
$^{U}/_{min}$	1 U_{AUS} = 65536 Ink	1 $\frac{1}{min}$ = 1 $\frac{1}{min}$	1/1	$\frac{\frac{1U}{1U}\cdot\frac{1U}{1U}\cdot\frac{1/min}{1/min}}{\frac{1U}{1U}} = \frac{1\,U/min}{1\,U/min}$	num: 1 div: 1
1/10 $^{\circ}/_{s}$ ($^{\circ}/_{10s}$)	1 U_{AUS} = 3600 $^{\circ}/_{10}$	1 $\frac{1}{s}$ = 1/60 $\frac{1}{min}$	1/1	$\frac{\frac{1U}{1U}\cdot\frac{1U}{1U}\cdot\frac{1/min}{60/min}}{\frac{3600\,^{\circ}/_{10}}{1U}} = \frac{1\,U/min}{216000\,^{\circ}/_{10s}}$	num: 1 div: 216000
1/100 $^{U}/_{min}$ ($^{U}/_{100\ min}$)	1 U_{AUS} = 100 $^{U}/_{100}$	1 $\frac{1}{min}$ = 1 $\frac{1}{min}$	1/1	$\frac{\frac{1U}{1U}\cdot\frac{1U}{1U}\cdot\frac{1/min}{1/min}}{\frac{100\,U/100}{1U}} = \frac{1\,U/min}{100\,U/100min}$	num: 1 div: 100
1/100 $^{U}/_{min}$ ($^{U}/_{100\ min}$)			2/3	$\frac{\frac{1U}{1U}\cdot\frac{2U}{3U}\cdot\frac{1/min}{1/min}}{\frac{100\,U/100}{1U}} = \frac{2\,U/min}{300\,U/100min}$	num: 2 div: 300
1/10 $^{mm}/_{s}$ ($^{mm}/_{10s}$)	1 U_{AUS} = 631.5 $^{mm}/_{10}$	1 $\frac{1}{s}$ = 1/60 $\frac{1}{min}$	1/1	$\frac{\frac{1U}{1U}\cdot\frac{1U}{1U}\cdot\frac{1/min}{60/min}}{\frac{631.5\,mm/10}{1U}} = \frac{1\,U/min}{37890\,mm/10s}$	num: 1 div: 37890
1/10 $^{mm}/_{s}$ ($^{mm}/_{10s}$)			4/5	$\frac{\frac{1U}{1U}\cdot\frac{4U}{5U}\cdot\frac{1/min}{60/min}}{\frac{631.5\,U/100}{1U}} = \frac{4\,U/4096min}{189450\,U/100min}$	num: 2 div: 94725

6.2.2.4 Objekt 6097_h: acceleration_factor

Das Objekt **acceleration_factor** dient zur Umrechnung aller Beschleunigungswerte der Applikation von **acceleration_units** in die interne Einheit **Umdrehungen pro Minute pro 256 Sekunden** (1 Umdrehung entspricht 65536 Inkremente). Es besteht aus Zähler und Nenner.

Index	**6097_h**
Name	**acceleration_factor**
Object Code	ARRAY
No. of Elements	2
Data Type	UINT32

Sub-Index	**01_h**
Description	**numerator**
Access	rw
PDO Mapping	yes
Units	--
Value Range	--
Default Value	1

Sub-Index	**02_h**
Description	**divisor**
Access	rw
PDO Mapping	yes
Units	--
Value Range	--
Default Value	1

Die Berechnung des **acceleration_factor** setzt sich ebenfalls aus zwei Teilen zusammen: Einem Umrechnungsfaktor von internen Längeneinheiten in **position_units** und einem Umrechnungsfaktor von internen Zeiteinheiten zum Quadrat in benutzerdefinierte Zeiteinheiten zum Quadrat (z.B. von Sekunden^2 in Minuten^2). Der erste Teil entspricht der Berechnung des **position_factor** für den zweiten Teil kommt ein zusätzlicher Faktor hinzu:

time_factor_a Verhältnis zwischen interner Zeiteinheit zum Quadrat und benutzerdefinierter Zeiteinheit zum Quadrat
(z.B. **1 min^2** = 1 min·1min = **60s·1min**)

gear_ratio Getriebeverhältnis zwischen Umdrehungen am Eintrieb (U_{EIN}) und Umdrehungen am Abtrieb (U_{AUS})

feed_constant Verhältnis zwischen Umdrehungen am Abtrieb (U_{AUS}) und Bewegung in **position_units** (z.B. 1 U = 360° Grad)

Die Berechnung des **acceleration_factors** erfolgt mit folgender Formel:

$$\textbf{acceleration\_factor} = \frac{\textbf{numerator}}{\textbf{divisor}} = \frac{\textbf{256} \cdot \textbf{gear\_ratio} \cdot \textbf{time\_factor\_a}}{\textbf{feed\_constant}}$$

Auch der **acceleration_factor** wird getrennt nach Zähler und Nenner in den Regler geschrieben werden, so dass eventuell erweitert werden muss.

BEISPIEL

1. **Gewünschte Einheit am Abtrieb (acceleration_*units*)**
2. ***feed_constant*: Wie viel *position_units* sind 1 Umdrehung (U_{AUS})?**
3. ***time_factor_a*: Gewünschte Zeiteinheit² besteht aus wie viel Sek² ?**
4. **Getriebefaktor (*gear_ratio*) U_{EIN} pro U_{AUS}**
5. **Werte in Formel einsetzen**

1.	2.	3.	4.	5.	ERGEBNIS Gekürzt
$^{U}/_{min\ s}$ ($^{U}/_{min\ s}$)	1 U_{AUS} = 1 U_{EIN}	$1 \frac{1}{min \cdot s}$ = $256 \frac{1}{256 \cdot min \cdot s}$	1/1	$\frac{1U}{1U} \cdot \frac{1U}{1U} \cdot \frac{256\,{}^{1}/_{256\ min \cdot s}}{1\,{}^{1}/_{1min \cdot s}} = \frac{256\,{}^{U}/_{256 \cdot s}}{1\,{}^{U}/_{s}}$	num: 256 div: 1
1/10 $^{°}/_{s^2}$ ($^{°}/_{10s^2}$)	1 U_{AUS} = 3600 $^{°}/_{10}$	$1 \frac{1}{s^2}$ = $1/60 \frac{1}{min \cdot s}$ = $256/60 \frac{1}{256 \cdot min \cdot s}$	1/1	$\frac{1U}{1U} \cdot \frac{1U}{1U} \cdot \frac{256\,{}^{1}/_{256\ min \cdot s}}{60\,{}^{1}/_{min \cdot s}} = \frac{256\,{}^{U}/_{256 \cdot s}}{216000\,{}^{°}/_{10s^2}}$	num: 4 div: 3375
1/100 $^{U}/_{min^2}$ ($^{U}/_{100\ min^2}$)	1 U_{AUS} = 100 $^{U}/_{100}$	$1 \frac{1}{min^2}$ = $60 \frac{1}{min \cdot s}$ = $256 \cdot 60 \frac{1}{256 \cdot min \cdot s}$	1/1	$\frac{1U}{1U} \cdot \frac{1U}{1U} \cdot \frac{15360\,{}^{1}/_{256\ min \cdot s}}{1\,{}^{1}/_{min \cdot min}} = \frac{15360\,{}^{U}/_{256 \cdot s}}{100\,{}^{U}/_{100min^2}}$	num: 3840 div: 25
1/100 $^{U}/_{min^2}$ ($^{U}/_{100\ min^2}$)			2/3	$\frac{1U}{1U} \cdot \frac{2U}{3U} \cdot \frac{15360\,{}^{1}/_{256\ min \cdot s}}{1\,{}^{1}/_{min \cdot min}} = \frac{30720\,{}^{U}/_{256 \cdot s}}{300\,{}^{U}/_{100min^2}}$	num: 2560 div: 25
1/10 $^{mm}/_{s^2}$ ($^{mm}/_{10s^2}$)	1 U_{AUS} = 631.5 $^{mm}/_{10}$	$1 \frac{1}{s^2}$ = $1/60 \frac{1}{min \cdot s}$ = $256/60 \frac{1}{256 \cdot min \cdot s}$	1/1	$\frac{1U}{1U} \cdot \frac{1U}{1U} \cdot \frac{256\,{}^{1}/_{256\ min \cdot s}}{60\,{}^{1}/_{min \cdot min}} = \frac{256\,{}^{U}/_{256 \cdot s}}{37890\,{}^{U}/_{100min^2}}$	num: 128 div: 18945
1/10 $^{mm}/_{s^2}$ ($^{mm}/_{10s^2}$)			4/5	$\frac{1U}{1U} \cdot \frac{4U}{5U} \cdot \frac{256\,{}^{1}/_{256\ min \cdot s}}{60\,{}^{1}/_{min \cdot min}} = \frac{1024\,{}^{U}/_{256 \cdot s}}{189450\,{}^{U}/_{100min^2}}$	num: 512 div: 94725

6.2.2.5 Objekt $607E_h$: polarity

Das Vorzeichen der Positions- und Geschwindigkeitswerte des Reglers kann mit dem entsprechenden polarity_flag eingestellt werden. Dieses kann dazu dienen, die Drehrichtung des Motors bei gleichen Sollwerten zu invertieren.
In den meisten Applikationen ist es sinnvoll, das **position_polarity_flag** und das **velocity_polarity_flag** auf den gleichen Wert zu setzen.

Index	**$607E_h$**
Name	**polarity**
Object Code	VAR
Data Type	UINT8

Access	rw
PDO Mapping	yes
Units	--
Value Range	0_h, 40_h, 80_h, $C0_h$
Default Value	0

Bit	Wert	Name	Bedeutung
6	40_h	**velocity_polarity_flag**	0: multiply by 1 (default) 1: multiply by –1 (invers)
7	80_h	**position_polarity_flag**	0: multiply by 1 (default) 1: multiply by –1 (invers)

6.3 Endstufenparameter

6.3.1 Übersicht

Aus dem Zwischenkreis wird der Motor über die IGBTs gespeist. Die Endstufe enthält eine Reihe von Sicherheitsfunktionen, die zum Teil parametriert werden können:

- Reglerfreigabelogik (Software- und Hardwarefreigabe)
- Überstromüberwachung
- Überspannungs- / Unterspannungs-Überwachung des Zwischenkreises
- Leistungsteilüberwachung

6.3.2 Beschreibung der Objekte

Index	Objekt	Name	Typ	Attr.
6510_h	VAR	drive_data		

6.3.2.1 Objekt 6510_h_10_h: enable_logic

Mit der steigende Flanke an dem digitalen Eingang DIN9 wird die **Reglerfreigabe** erkannt. Damit die Endstufe des Antriebsreglers aktiviert werden kann, muss sich der Regler in dem Zustand „Betriebsbereit, wach“ befinden. Ist das der Fall wechselt der Regler in den Zustand „Endstufe einschalten“ in dem direkt die Ansteuersignale der Leistungstransistoren freigeschaltet werden. Diese Aufgabe wird vom Mikroprozessor übernommen. Das wiederum bedeutet, dass bei defektem Mikroprozessor eine vorher eingeschaltete Endstufe nicht ausgeschaltet werden kann. Die einzige Möglichkeit die Endstufe in einem solchen Fall abzuschalten, ist den Zwischenkreis und Logikversorgung abzuschalten. Die **Reglerfreigabe** wird vom Mikrokontroller des Reglers verarbeitet. Je nach Betriebsart reagiert der Regler nach der Wegnahme dieses Signals unterschiedlich:

- **Positionierbetrieb und drehzahlgeregelter Betrieb**

  Der Motor wird nach der Wegnahme des Signals mit einer definierten Bremsrampe abgebremst. Die Endstufe wird erst abgeschaltet, wenn die Motordrehzahl unterhalb 10 min^{-1} liegt und die eventuell vorhandene Haltebremse angezogen hat.
- **Momentengeregelter Betrieb**

  Die Endstufe wird unmittelbar nach der Wegnahme des Signals abgeschaltet. Gleichzeitig wird eine eventuell vorhandene Haltebremse angezogen. Der Motor trudelt also ungebremst aus bzw. wird nur durch die eventuell vorhandene Haltebremse gestoppt

ACHTUNG !
DIN9 garantiert nicht, dass der Motor spannungsfrei ist.

Beim Betrieb des Reglers über den CAN-Bus kann die Freigabe über den CAN-Bus gesteuert werden. Dazu muss das Objekt **6510_h_10_h** (**enable_logic**) auf 2 gesetzt werden.

Index	**6510$_h$**
Name	**drive_data**
Object Code	RECORD
No. of Elements	44

Sub-Index	**10$_h$**
Description	**enable_logic**
Data Type	UINT16
Access	rw
PDO Mapping	no
Units	
Value Range	0...2
Default Value	0

Wert	Bedeutung
0	Reglerfreigabe durch digitalen Eingang DIN9
1	Reglerfreigabe durch digitalen Eingang DIN9 + RS232
2	Reglerfreigabe durch digitalen Eingang DIN9 + CAN

6.4 Stromregler und Motoranpassung

Vorsicht !

Falsche Einstellungen der Stromreglerparameter und der Strombegrenzungen können den **Motor** und unter Umständen auch den **Servoregler** innerhalb kürzester Zeit **zerstören**!

6.4.1 Übersicht

Der Parametersatz des Servoreglers muss für den angeschlossenen Motor und den verwendeten Kabelsatz angepasst werden. Betroffen sind folgende Parameter:

- Nennstrom Abhängig vom Motor
- Überlastbarkeit Abhängig vom Motor
- Polzahl Abhängig vom Motor
- Stromregler Abhängig vom Motor
- Drehsinn Abhängig vom Motor und der Phasenfolge im Motor- und Winkelgeberkabel
- Offsetwinkel Abhängig vom Motor und der Phasenfolge im Motor- und Winkelgeberkabel

Diese Daten müssen beim erstmaligen Einsatz eines Motortyps mit dem Programm DIS-2 ServoCommander bestimmt werden. Für eine Reihe von Motoren können Sie auch fertige Parametersätze über Ihren Händler beziehen. Bitte beachten Sie, dass Drehsinn und Offsetwinkel auch vom verwendeten Kabelsatz abhängen. Die Parametersätze arbeiten daher nur bei identischer Verkabelung.

Bei verdrehter Phasenfolge im Motor- oder Winkelgeberkabel kann es zu einer Mitkopplung kommen, so dass die Drehzahl im Motor nicht geregelt werden kann. Der Motor kann unkontrolliert durchdrehen !

6.4.2 Beschreibung der Objekte

Index	Objekt	Name	Typ	Attr.
6075_h	VAR	motor_rated_current	UINT32	rw
6073_h	VAR	max_current	UINT16	rw
$604D_h$	VAR	pole_number	UINT8	rw
6410_h	RECORD	motor_data	UINT32	rw
$60F6_h$	RECORD	torque_control_parameters	UINT16	rw

6.4.2.1 Objekt 6075_h: motor_rated_current

Dieser Wert ist dem Motortypenschild zu entnehmen und wird in der Einheit Milliampere eingegeben. Es wird immer der Effektivwert (RMS) angenommen. Es kann kein Strom vorgegeben werden, der oberhalb des Reglernennstromes liegt.

Index	**6075_h**
Name	**motor_rated_current**
Object Code	VAR
Data Type	UINT32

Access	rw
PDO Mapping	yes
Units	mA
Value Range	0...nominal_current
Default Value	2870

Wird das Objekt **6075_h** (**motor_rated_current**) mit einem neuen Wert beschrieben, muss in jedem Fall auch das Objekt **6073_h** (**max_current**) neu parametriert werden.

6.4.2.2 Objekt 6073_h: max_current

Servomotoren dürfen in der Regel für einen bestimmten Zeitraum überlastet werden. Mit diesem Objekt wird der höchstzulässige Motorstrom eingestellt. Er bezieht sich auf den Motornennstrom (Objekt **6075_h**: **motor_rated_current**) und wird in Tausendsteln eingestellt. Der Wertebereich wird nach oben durch den maximalen Reglerstrom begrenzt. Viele Motoren dürfen kurzzeitig um den Faktor 2 überlastet werden. In diesem Fall ist in dieses Objekt der Wert 2000 einzuschreiben.

Das Objekt **6073_h** (**max_current**) darf erst beschrieben werden, wenn zuvor das Objekt **6075_h** (**motor_rated_current**) gültig beschrieben wurde.

Index	**6073_h**
Name	**max_current**
Object Code	VAR
Data Type	UINT16

Access	rw
PDO Mapping	yes
Units	per thousands of rated current
Value Range	-
Default Value	1968

6.4.2.3 Objekt $604D_h$: pole_number

Die Polzahl des Motors ist dem Motordatenblatt oder dem Parametrierprogramm DIS-2 ServoCommander zu entnehmen. Die Polzahl ist immer geradzahlig. Oft wird statt der Polzahl die Polpaarzahl angegeben. Die Polzahl entspricht dann der doppelten Polpaarzahl.

Index	**$604D_h$**
Name	**pole_number**
Object Code	VAR
Data Type	UINT8

Access	rw
PDO Mapping	yes
Units	--
Value Range	2... 128
Default Value	2

6.4.2.4 Objekt 6410_h_03_h: iit_time_motor

Servomotoren dürfen in der Regel für einen bestimmten Zeitraum überlastet werden. Über dieses Objekt wird angegeben, wie lange der angeschlossene Motor mit dem im Objekt **6073_h** (**max_current**) angegebenen Strom bestromt werden darf. Nach Ablauf der IIT-Zeit wird der Strom zum Schutz des Motors automatisch auf den im Objekt **6075_h** (**motor_rated_current**) angegebenen Wert begrenzt. Die Standardeinstellung liegt bei zwei Sekunden und trifft für die meisten Motoren zu.

Index	**6410_h**
Name	**motor_data**
Object Code	RECORD
No. of Elements	5

Sub-Index	**03_h**
Description	**iit_time_motor**
Data Type	UINT16
Access	rw
PDO Mapping	no
Units	ms
Value Range	0...10000
Default Value	2000

6.4.2.5 Objekt 6410_h_04_h: iit_ratio_motor

Über das Objekt kann **iit_ratio_motor** kann die aktuelle Auslastung der I^2t-Begrenzung in Promille ausgelesen werden.

Sub-Index	**04_h**
Description	**iit_ratio_motor**
Data Type	UINT16
Access	ro
PDO Mapping	no
Units	promille
Value Range	--
Default Value	--

6.4.2.6 Objekt 6410_h_10_h: phase_order

In der Phasenfolge (**phase_order**) werden Verdrehungen zwischen Motorkabel und Winkelgeberkabel berücksichtigt. Sie kann dem Parametrierprogramm DIS-2 ServoCommander entnommen werden.

Sub-Index	**10_h**
Description	**phase_order**
Data Type	UINT16
Access	rw
PDO Mapping	no
Units	--
Value Range	0, 1
Default Value	1

Wert	Bedeutung
1	Rechts
0	Links

6.4.2.7 Objekt 6410_h_11_h: encoder_offset_angle

Bei den verwendeten Servomotoren befinden sich Dauermagnete auf dem Rotor. Diese erzeugen ein magnetisches Feld, dessen Ausrichtung zum Stator von der Rotorlage abhängt. Für die elektronische Kommutierung muss der Regler das elektromagnetische Feld des Stators immer im richtigen Winkel zu diesem Permanentmagnetfeld einstellen. Er bestimmt hierzu laufend mit einem Winkelgeber (Resolver etc.) die Rotorlage.
Die Orientierung des Winkelgebers zum Dauermagnetfeld muss in das Objekt **encoder_offset_angle** eingetragen werden. Mit dem Parametrierprogramm kann dieser Winkel bestimmt werden (Parameter / Geräteparameter / Winkelgeber-Einstellungen). Der mit DIS-2 ServoCommander bestimmte Winkel liegt im Bereich von ±180°. Er muss folgendermaßen umgerechnet werden:

$$\textbf{encoder\_offset\_angle} = \text{„Offsetwinkel des Winkelgebers“} \times \frac{32767}{180°}$$

Index	**6410_h**
Name	**motor_data**
Object Code	RECORD
No. of Elements	5

Sub-Index	**11_h**
Description	**encoder_offset_angle**
Data Type	INT16
Access	rw
PDO Mapping	yes
Units	
Value Range	-32767...32767
Default Value	$2C60_h$ (62,4°)

6.4.2.8 Objekt 2415_h: current_limitation

Mit der Objektgruppe **current_limitation** kann unabhängig von der Betriebsart (Drehzahlregelung, Positionierung) der Maximalstrom für den Motor begrenzt werden, wodurch z.B. ein drehmomentbegrenzter Drehzahlbetrieb ermöglicht wird. Über das Objekt **limit_current_input_channel** wird die Sollwert-Quelle des Begrenzungsmoment vorgegeben. Hier kann zwischen der Vorgabe eines direkten Sollwerts (CAN / RS232) oder der Vorgabe über einen analogen Eingang gewählt werden. Über das Objekt **limit_current** wird je nach gewählter Quelle entweder das Begrenzungsmoment (Quelle = CAN / RS232) oder der Skalierungsfaktor für die Analogeingänge (Quelle = Analogeingang) vorgegeben. Im ersten Fall wird direkt auf den momentproportionalen Strom in mA begrenzt, im zweiten Fall wird der Strom in mA angegeben, der einer anliegenden Spannung von 10V entsprechen soll.

Index	**2415_h**
Name	**current_limitation**
Object Code	RECORD
No. of Elements	2

Sub-Index	**01_h**
Description	**limit_current_input_channel**
Data Type	INT8
Access	rw
PDO Mapping	no
Units	--
Value Range	0...4
Default Value	0

Sub-Index	**02_h**
Description	**limit_current**
Data Type	INT32
Access	rw
PDO Mapping	no
Units	mA
Value Range	--
Default Value	5650

Wert	Bedeutung
0	Keine Begrenzung
1	AIN0
2	AIN1
3	RS232
4	CAN

6.4.2.9 Objekt $60F6_h$: torque_control_parameters

Die Daten des Stromreglers müssen dem Parametrierprogramm entnommen werden. Hierbei sind folgende Umrechungen zu beachten:
Die Verstärkung des Stromreglers muss mit 256 multipliziert werden. Bei einer Verstärkung von 1.5 im Menü „Stromregler“ des Parametrierprogramms DIS-2 ServoCommander ist in das Objekt **torque_control_gain** der Wert 384 = 180_h einzuschreiben.
Die Zeitkonstante des Stromreglers ist im Parametrierprogramm in Millisekunden angegeben. Um diese Zeitkonstante in das Objekt **torque_control_time** übertragen zu können, muss sie zuvor in Mikrosekunden umgerechnet werden. Bei einer angegebenen Zeit von 0.6 Millisekunden ist entsprechend der Wert 600 in das Objekt **torque_control_time** einzutragen.

Index	**$60F6_h$**
Name	**torque_control_parameters**
Object Code	RECORD
No. of Elements	2

Sub-Index	**01_h**
Description	**torque_control_gain**
Data Type	UINT16
Access	rw
PDO Mapping	no
Units	256 = „1“
Value Range	0...32*256
Default Value	256

Sub-Index	**02_h**
Description	**torque_control_time**
Data Type	UINT16
Access	rw
PDO Mapping	no
Units	µs
Value Range	100... 65500
Default Value	2000

6.5 Drehzahlregler

6.5.1 Übersicht

Der Parametersatz des Servoreglers muss für die Applikation angepasst werden. Besonders die Verstärkung ist stark abhängig von eventuell an den Motor angekoppelten Massen. Die Daten müssen bei der Inbetriebnahme der Anlage mit Hilfe des Programms DIS-2 ServoCommander optimal bestimmt werden.

Falsche Einstellungen der Drehzahlreglerparameter können zu starken Schwingungen führen und eventuell Teile der Anlage zerstören!

6.5.2 Beschreibung der Objekte

Index	Objekt	Name	Typ	Attr.
$60F9_h$	RECORD	velocity_control_parameters		rw

6.5.2.1 Objekt $60F9_h$: velocity_control_parameters_Set

Die Daten des Drehzahlreglers müssen dem Parametrierprogramm entnommen werden. Hierbei sind folgende Umrechungen zu beachten:
Die Verstärkung des Drehzahlreglers muss mit 256 multipliziert werden.
Bei einer Verstärkung von 1.5 im Menü „Drehzahlregler“ des Parametrierprogramms DIS-2 ServoCommander ist in das Objekt **velocity_control_gain** der Wert 384 = 180_h einzuschreiben.
Die Zeitkonstante des Drehzahlreglers ist im Parametrierprogramm DIS-2 ServoCommander in Millisekunden angegeben. Um diese Zeitkonstante in das Objekt **velocity_control_time** übertragen zu können, muss sie zuvor in Mikrosekunden umgerechnet werden. Bei einer angegebenen Zeit von 2.0 Millisekunden ist entsprechend der Wert 2000 in das Objekt **velocity_control_time** einzutragen.

Index	**60F9$_h$**
Name	**velocity_control_parameter_set**
Object Code	RECORD
No. of Elements	3

Sub-Index	**01$_h$**
Description	**velocity_control_gain**
Data Type	UINT16
Access	rw
PDO Mapping	no
Units	256 = Gain 1
Value Range	26…64*256 (16384)
Default Value	179 (0,7)

Sub-Index	**02$_h$**
Description	**velocity_control_time**
Data Type	UINT16
Access	rw
PDO Mapping	no
Units	µs
Value Range	200...32000
Default Value	8000

Sub-Index	**04$_h$**
Description	**velocity_control_filter_time**
Data Type	UINT16
Access	rw
PDO Mapping	no
Units	µs
Value Range	200...32000
Default Value	1600

6.6 Lageregler (Position Control Function)

6.6.1 Übersicht

In diesem Kapitel sind alle Parameter beschrieben, die für den Lageregler erforderlich sind. Am Eingang des Lagereglers liegt der Lage-Sollwert (**position_demand_value**) vom Fahrkurven-Generator an. Außerdem wird der Lage-Istwert (**position_actual_value**) vom Winkelgeber (Resolver, Inkrementalgeber etc.) zugeführt. Das Verhalten des Lagereglers kann durch Parameter beeinflusst werden. Um den Lageregelkreis stabil zu halten, ist eine Begrenzung der Ausgangsgröße (**control_effort**) möglich. Die Ausgangsgröße wird als Drehzahl-Sollwert dem Drehzahlregler zugeführt. Alle Ein- und Ausgangsgrößen des Lagereglers werden in der **Factor Group** von den applikationsspezifischen Einheiten in die jeweiligen internen Einheiten des Reglers umgerechnet. Die internen Größen werden mit einem Sternchen gekenzeichnet.

Folgende Unterfunktionen sind in diesem Kapitel definiert:

1. Schleppfehler (Following_Error)

Als Schleppfehler wird die Abweichung des Lage-Istwertes (**position_actual_value**) vom Lage-Sollwert (**position_demand_value**) bezeichnet. Wenn dieser Schleppfehler für einen bestimmten Zeitraum größer ist als im Schleppfehler-Fenster (**following_error_window**) angegeben, so wird das Bit 13 **following_error** im Objekt **statusword** gesetzt. Der zulässige Zeitraum kann über das Objekt **following_error_time_out** vorgegeben werden.

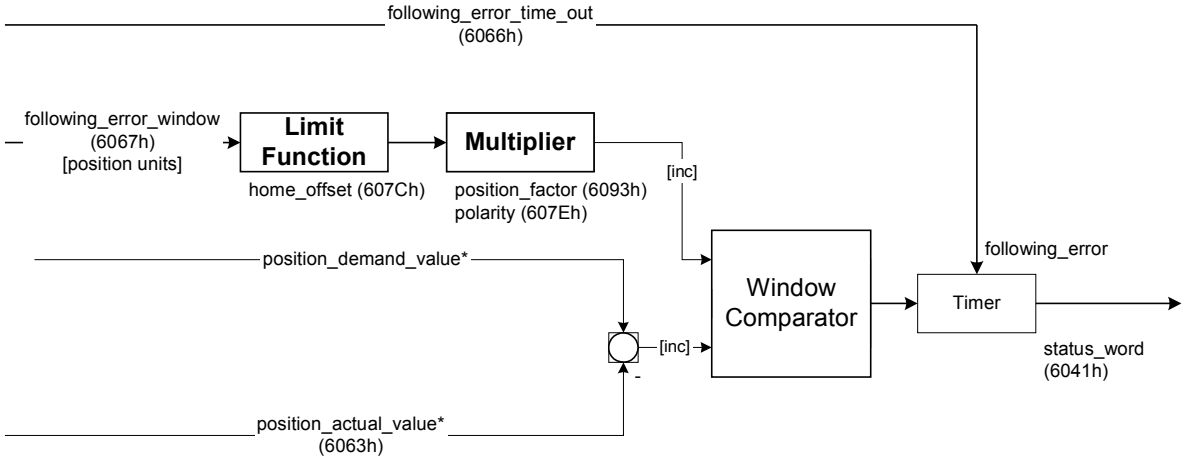


Abbildung 6.5: Schleppfehler – Funktionsübersicht

Die Abbildung 6.6 zeigt, wie die Fensterfunktion für die Meldung „Schleppfehler“ definiert ist. Symmetrisch um die Sollposition (**position_demand_value**) x_i ist der Bereich zwischen x_i-x_0 und x_i+x_0 definiert. Die Positionen x_{t2} und x_{t3} liegen z.B. außerhalb dieses Fensters (**following_error_window**). Wenn der Antrieb dieses Fenster verlässt und nicht in der im Objekt **following_error_time_out** vorgegebenen Zeit in das Fenster zurückkehrt, dann wird das Bit 13 **following_error** im **statusword** gesetzt.

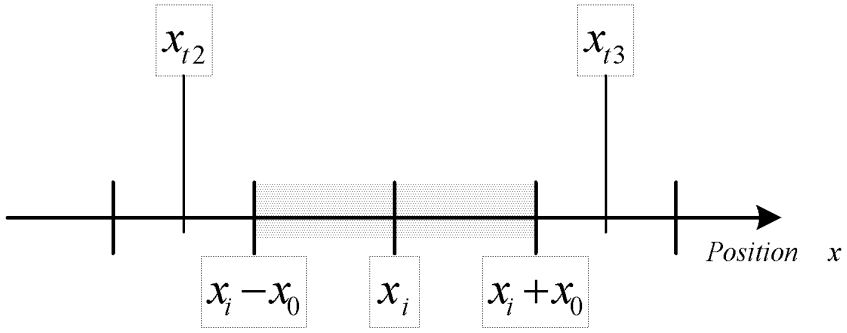


Abbildung 6.6: Schleppfehler

2. Position erreicht (Position Reached)

Diese Funktion bietet die Möglichkeit, ein Positionsfenster um die Zielposition (**target_position**) herum zu definieren. Wenn sich die Ist-Position des Antriebs für eine bestimmte Zeit – die **position_window_time** – in diesem Bereich befindet, dann wird das damit verbundene Bit 10 (**target_reached**) im **statusword** gesetzt.

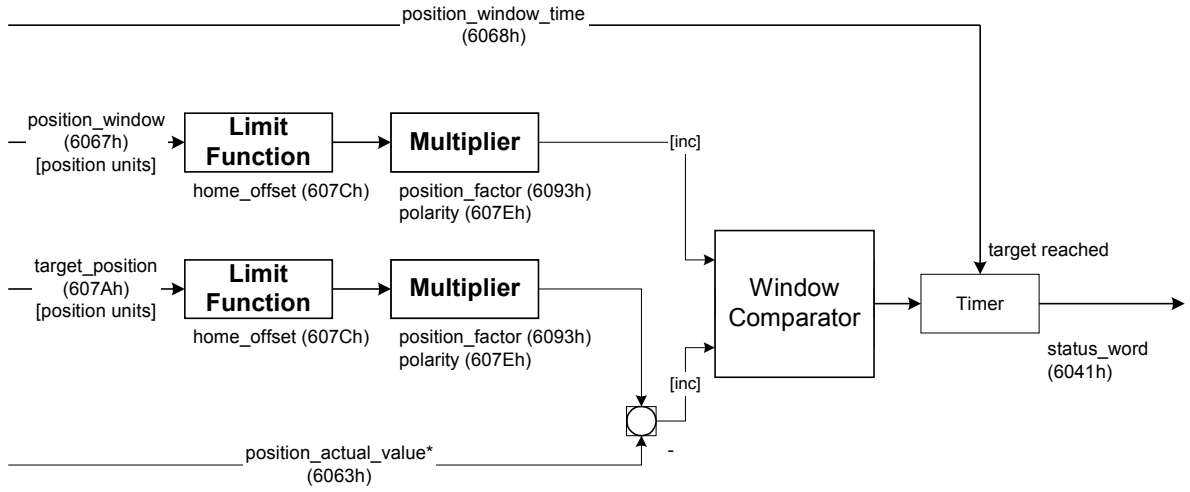


Abbildung 6.7: Position erreicht – Funktionsübersicht

Die Abbildung 6.8 zeigt, wie die Fensterfunktion für die Meldung „Position erreicht“ definiert ist. Symmetrisch um die Zielposition (**target_position**) x_i ist der Positionsbereich zwischen x_i-x_0 und x_i+x_0 definiert. Die Positionen x_{t0} und x_{t1} liegen z.B. innerhalb dieses Positionsfensters (**position_window**). Wenn sich der Antrieb in diesem Fenster befindet, dann wird im Regler ein Timer gestartet. Wenn dieser Timer die im Objekt **position_window_time** vorgegebene Zeit erreicht und sich der Antrieb während dieser Zeit ununterbrochen im gültigen Bereich zwischen x_i-x_0 und x_i+x_0 befindet, dann wird Bit 10 **target_reached** im **statusword** gesetzt. Sobald der Antrieb den zulässigen Bereich verlässt, wird sowohl das Bit 10 als auch der Timer auf Null gesetzt.

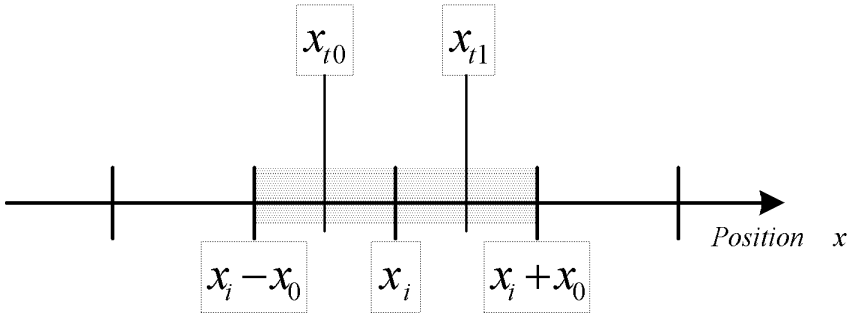


Abbildung 6.8: Position erreicht

6.6.2 Beschreibung der Objekte

6.6.2.1 In diesem Kapitel behandelte Objekte

Index	Objekt	Name	Typ	Attr.
6062_h	VAR	position_demand_value	INT32	ro
6063_h	VAR	position_actual_value*	INT32	ro
6064_h	VAR	position_actual_value	INT32	ro
6065_h	VAR	following_error_window	UINT32	rw
6066_h	VAR	following_error_time_out	UINT16	rw
6067_h	VAR	position_window	UINT32	rw
6068_h	VAR	position_window_time	UINT16	rw
$60FA_h$	VAR	control_effort	INT32	ro
$60FB_h$	RECORD	position_control_parameter_set		rw

6.6.2.2 Betroffene Objekte aus anderen Kapiteln

Index	Objekt	Name	Typ	Kapitel
$607A_h$	VAR	target_position	INT32	8.3 Betriebsart Positionieren
$607C_h$	VAR	home_offset	INT32	8.2 Referenzfahrt
$607E_h$	VAR	polarity	UINT8	6.2 Umrechnungsfaktoren
6093_h	VAR	position_factor	UINT32	6.2 Umrechnungsfaktoren
6094_h	ARRAY	velocity_encoder_factor	UINT32	6.2 Umrechnungsfaktoren
6040_h	VAR	controlword	INT16	7 Gerätesteuerung
6041_h	VAR	statusword	UINT16	7 Gerätesteuerung

6.6.2.3 Objekt $60FB_h$: position_control_parameter_set

Der Parametersatz des Servoreglers muss für die Applikation angepasst werden. Die Daten des Lagereglers müssen bei der Inbetriebnahme der Anlage mit Hilfe des Programms DIS-2 ServoCommander optimal bestimmt werden.

Falsche Einstellungen der Lagereglerparameter können zu starken Schwingungen führen und eventuell Teile der Anlage zerstören !

Der Lageregler vergleicht die Soll-Lage mit der Ist-Lage und bildet aus der Differenz unter Berücksichtigung der Verstärkung und eventuell des Integrators eine Korrekturgeschwindigkeit (Objekt **$60FA_h$**: **control_effort**), die dem Drehzahlregler zugeführt wird. Der Lageregler ist, gemessen am Strom- und Drehzahlregler, relativ langsam. Der Regler arbeitet daher intern mit Aufschaltungen, so dass die Ausregelarbeit für den Lageregler minimiert wird und der Regler schnell einschwingen kann.
Als Lageregler genügt ein Proportional-Glied. Die Verstärkung des Lagereglers muss mit 256 multipliziert werden. Bei einer Verstärkung von 1.5 im Menü „Lageregler“ des Parametrierprogramms DIS-2 ServoCommander ist in das Objekt **position_control_gain** der Wert 384 einzuschreiben.
Da der Lageregler schon kleinste Lageabweichungen in nennenswerte Korrekturgeschwindigkeiten umsetzt, würde es im Falle einer kurzen Störung (z.B. kurzzeitiges Klemmen der Anlage) zu sehr heftigen Ausregelvorgängen mit sehr großen Korrekturgeschwindigkeiten kommen. Dieses ist zu vermeiden, wenn der Ausgang des Lagereglers über das Objekt **position_control_v_max** sinnvoll (z.B. 500 min^{-1}) begrenzt wird.
Mit dem Objekt **position_error_tolerance_window** kann die Größe einer Lageabweichung definiert werden, bis zu der der Lageregler nicht eingreift (Totbereich). Dieses kann zur Stabilisierung eingesetzt werden, wenn z.B. Spiel in der Anlage vorhanden ist.

Index	**60FB$_h$**
Name	**position_control_parameter_set**
Object Code	RECORD
No. of Elements	4

Sub-Index	**01$_h$**
Description	**position_control_gain**
Data Type	UINT16
Access	rw
PDO Mapping	no
Units	256 = „1“
Value Range	0...64*256 (16384)
Default Value	52 (0,20)

Sub-Index	**04$_h$**
Description	**position_control_v_max**
Data Type	UINT32
Access	rw
PDO Mapping	no
Units	speed units
Value Range	0...32767 min^{-1}
Default Value	500 min^{-1}

Sub-Index	**05$_h$**
Description	**position_error_tolerance_window**
Data Type	UINT32
Access	rw
PDO Mapping	no
Units	position units
Value Range	0...65536 (1 U)
Default Value	13 (0,00020 U)

6.6.2.4 Objekt 6062_h: position_demand_value

Über dieses Objekt kann der aktuelle Lage-Sollwert ausgelesen werden. Diese wird vom Fahrkurven-Generator in den Lageregler eingespeist.

Index	**6062_h**
Name	**position_demand_value**
Object Code	VAR
Data Type	INT32

Access	ro
PDO Mapping	yes
Units	position units
Value Range	--
Default Value	--

6.6.2.5 Objekt 6064_h: position_actual_value

Über dieses Objekt kann die Ist-Lage ausgelesen werden. Diese wird dem Lageregler vom Winkelgeber aus zugeführt. Dieses Objekt wird in benutzerdefinierten Einheiten angegeben.

Index	**6064_h**
Name	**position_actual_value**
Object Code	VAR
Data Type	INT32

Access	ro
PDO Mapping	yes
Units	position units
Value Range	--
Default Value	--

6.6.2.6 Objekt 6065_h: following_error_window

Das Objekt **following_error_window** (Schleppfehler-Fenster) definiert um den Lage-Sollwert (**position_demand_value**) einen symmetrischen Bereich. Wenn sich der Lage-Istwert (**position_actual_value**) außerhalb des Schleppfehler-Fensters (**following_ error_window**) befindet, dann tritt ein Schleppfehler auf und das Bit 13 im Objekt **statusword** wird gesetzt. Folgende Ursachen können einen Schleppfehler verursachen:

- der Antrieb ist blockiert
- die Positioniergeschwindigkeit ist zu groß
- die Beschleunigungswerte sind zu groß
- das Objekt **following_error_window** ist mit einem zu kleinen Wert besetzt
- der Lageregler ist nicht richtig parametriert

Index	**6065_h**
Name	**following_error_window**
Object Code	VAR
Data Type	UINT32

Access	rw
PDO Mapping	yes
Units	position units
Value Range	$0...7FFFFFFF_h$
Default Value	$238E_h$ (ca. 50°)

6.6.2.7 Objekt 6066_h: following_error_time_out

Tritt ein Schleppfehler – länger als in diesem Objekt definiert – auf, dann wird das zugehörige Bit 13 **following_error** im **statusword** gesetzt.

Index	**6066_h**
Name	**following_error_time_out**
Object Code	VAR
Data Type	UINT16

Access	rw
PDO Mapping	yes
Units	ms
Value Range	0...26214
Default Value	100

6.6.2.8 Objekt $60FA_h$: control_effort

Die Ausgangsgröße des Lagereglers kann über dieses Objekt ausgelesen werden. Dieser Wert wird intern dem Drehzahlregler als Sollwert zugeführt.

Index	**$60FA_h$**
Name	**control_effort**
Object Code	VAR
Data Type	INT32

Access	ro
PDO Mapping	yes
Units	speed units
Value Range	--
Default Value	--

6.6.2.9 Objekt 6067_h: position_window

Mit dem Objekt **position_window** wird um die Zielposition (**target_position**) herum ein symmetrischer Bereich definiert. Wenn der Lage-Istwert (**position_actual_value**) eine bestimmte Zeit innerhalb dieses Bereiches liegt, wird die Zielposition (**target_position**) als erreicht angesehen.

Index	**6067_h**
Name	**position_window**
Object Code	VAR
Data Type	UINT32

Access	rw
PDO Mapping	yes
Units	position units
Value Range	--
Default Value	1820 (1820 / 65536 U = 10°)

6.6.2.10 Objekt 6068_h: position_window_time

Wenn sich die Ist-Position des Antriebes innerhalb des Positionierfensters (**position_window**) befindet und zwar solange, wie in diesem Objekt definiert, dann wird das zugehörige Bit 10 **target_reached** im **statusword** gesetzt.

Index	**6068_h**
Name	**position_window_time**
Object Code	VAR
Data Type	UINT16

Access	rw
PDO Mapping	yes
Units	ms
Value Range	0...262146
Default Value	100

6.7 Analoge Eingänge

6.7.1 Übersicht

Die Antriebsregler der Reihe DIS-2 verfügen über zwei analoge Eingänge, über die dem Regler beispielsweise Sollwerte vorgegeben werden können. Diese analogen Eingänge können nur über das Programm DIS-2 ServoCommander parametriert werden.

6.8 Digitale Ein- und Ausgänge

6.8.1 Übersicht

Alle digitalen Eingänge des Reglers können über den CAN-Bus gelesen und 2 digitalen Ausgänge können beliebig gesetzt werden.

6.8.2 Beschreibung der Objekte

Index	Objekt	Name	Typ	Attr.
$60FD_h$	VAR	digital_inputs	UINT32	ro
$60FE_h$	ARRAY	digital_outputs	UINT32	rw

6.8.2.1 Objekt $60FD_h$: digital_inputs

Über das Objekt **$60FD_h$** können die digitalen Eingänge ausgelesen werden:

Index	**$60FD_h$**
Name	**digital_inputs**
Object Code	VAR
Data Type	UINT32

Access	ro
PDO Mapping	yes
Units	--
Value Range	gemäß u. Tabelle
Default Value	0

Bit	Wert	digitaler Eingang
0	00000001_h	Negativer Endschalter
1	00000002_h	Positiver Endschalter
3	00000008_h	Interlock (Reglerfreigabe (DIN9) fehlt)
16...25	$03FF0000_h$	DIN0...DIN9

6.8.2.2 Objekt $60FE_h$: digital_outputs

Über das Objekt **$60FE_h$** können die digitalen Ausgänge angesteuert werden. Über das Objekt **digital_outputs_data** können die 2 Ausgänge dann beliebig gesetzt werden. Es ist zu beachten, dass bei der Ansteuerung der digitalen Ausgänge eine Verzögerung von bis zu 1 ms auftreten kann. Wann die Ausgänge wirklich gesetzt werden, kann durch Zurücklesen des Objekts **$60FE_h$** festgestellt werden.

Index	**$60FE_h$**
Name	**digital_outputs**
Object Code	ARRAY
No. of Elements	2
Data Type	UINT32

Sub-Index	**01_h**
Description	**digital_outputs_data**
Access	rw
PDO Mapping	yes
Units	--
Value Range	--
Default Value	0

Bit	Wert	Digitaler Ausgang
0	00000001_h	Bremse
16	00010000_h	Betriebsbereit
17, 18	00060000_h	DOUT1, DOUT2

6.9 Endschalter

6.9.1 Übersicht

Für die Definition der Referenzposition des Antriebreglers können Endschalter (limit switch) verwendet werden. Nähere Informationen zu den möglichen Referenzfahrt-Methoden finden sie im Kapitel *8.2, Betriebsart Referenzfahrt (Homing Mode)*.

6.9.2 Beschreibung der Objekte

Index	Objekt	Name	Typ	Attr.
6510h	RECORD	drive_data		rw

6.9.2.1 Objekt 6510_h_11_h: limit_switch_polarity

Die Polarität der Endschalter kann durch das Objekt **6510_h_11_h** (**limit_switch_polarity**) programmiert werden. Für öffnende Endschalter ist in dieses Objekt eine Null, bei der Verwendung von schließenden Kontakten ist eine Eins einzutragen. Dies bezieht sich immer auf beide Endschalter!

Index	**6510_h**
Name	**drive_data**
Object Code	RECORD
No. of Elements	44

Sub-Index	**11_h**
Description	**limit_switch_polarity**
Data Type	INT16
Access	rw
PDO Mapping	no
Units	--
Value Range	0, 1
Default Value	0

Wert	Bedeutung
0	Öffner
1	Schließer

6.9.2.2 Objekt 6510_h_15_h: limit_switch_deceleration

Das Objekt **limit_switch_deceleration** legt die Beschleunigung fest, mit der gebremst wird, wenn während des normalen Betriebs der Endschalter erreicht wird (Endschalter-Nothalt-Rampe).

Sub-Index	**15_h**
Description	**limit_switch_deceleration**
Data Type	INT32
Access	rw
PDO Mapping	no
Units	acceleration units
Value Range	0...200000 min^{-1}/s
Default Value	200000 min^{-1}/s

6.10 Geräteinformationen

Index	Objekt	Name	Typ	Attr.
1018h	RECORD	identity_object		rw
6510h	RECORD	drive_data		rw

Über zahlreiche CAN-Objekte können die verschiedensten Informationen wie Reglertyp, verwendete Firmware, etc. aus dem Gerät ausgelesen werden.

6.10.1 Beschreibung der Objekte

6.10.1.1 Objekt 1018_h: identity_object

Über das in der DS301 festgelegte **identity_object** kann der Regler in einem CANopen-Netzwerk eindeutig identifiziert werden. Zu diesem Zweck kann der Herstellercode (**vendor_id**), ein eindeutiger Produktcode (**product_code**), die Revisionsnummer der CANopen-Implementation (**revision_number**) und die Seriennummer des Geräts (**serial_number**) ausgelesen werden.

Index	**1018_h**
Name	**identity_object**
Object Code	RECORD
No. of Elements	4

Sub-Index	**01_h**
Description	**vendor_id**
Data Type	UINT32
Access	ro
PDO Mapping	no
Units	--
Value Range	0000003B
Default Value	0000003B

Sub-Index	**02$_h$**
Description	**product_code**
Data Type	UINT32
Access	ro
PDO Mapping	no
Units	--
Value Range	s.u.
Default Value	s.u.

Wert	Bedeutung
1121$_h$	DIS-2 48/10
1122$_h$	DIS-2 24/8

Sub-Index	**03$_h$**
Description	**revision_number**
Data Type	UINT32
Access	ro
PDO Mapping	no
Units	MMMMSSSS$_h$ (M: main version, S: sub version)
Value Range	--
Default Value	--

Sub-Index	**04$_h$**
Description	**serial_number**
Data Type	UINT32
Access	ro
PDO Mapping	no
Units	--
Value Range	--
Default Value	--

6.10.1.2 Objekt 6510_h_$A1_h$: drive_type

Über das Objekt **drive_type** kann der Gerätetyp des Reglers ausgelesen werden. Dieses Objekt dient der Kompatibilität zu früheren Versionen.

Sub-Index	**$A1_h$**
Description	**drive_type**
Data Type	UINT32
Access	ro
PDO Mapping	no
Units	
Value Range	siehe 1018_h_02_h, product_code
Default Value	siehe 1018_h_02_h, product_code

6.10.1.3 Objekt 6510_h_$A9_h$: firmware_main_version

Über das Objekt **firmware_main_version** kann die Hauptversionsnummer der Firmware (Produktstufe) ausgelesen werden.

Sub-Index	**$A9_h$**
Description	**firmware_main_version**
Data Type	UINT32
Access	ro
PDO Mapping	no
Units	$MMMMSSSS_h$ (M: main version, S: sub version)
Value Range	--
Default Value	--

6.10.1.4 Objekt 6510_h_AA_h: firmware_custom_version

Über das Objekt **firmware_custom_version** kann die Versionsnummer der kunden-spezifischen Variante der Firmware ausgelesen werden.

Sub-Index	**AA_h**
Description	**firmware_custom_version**
Data Type	UINT32
Access	ro
PDO Mapping	no
Units	$MMMMSSSS_h$ (M: main version, S: sub version)
Value Range	--
Default Value	--

7 Gerätesteuerung (Device Control)

7.1 Zustandsdiagramm (State Machine)

7.1.1 Übersicht

Das nachfolgende Kapitel beschreibt, wie der Regler unter CANopen gesteuert wird, also wie beispielsweise die Endstufe eingeschaltet oder ein Fehler quittiert wird.
Unter CANopen wird die gesamte Steuerung des Reglers über zwei Objekte realisiert: Über das **controlword** kann der Host den Regler steuern, während der Status des Reglers im Objekt **statusword** zurückgelesen werden kann. Zur Erklärung der Reglersteuerung werden die folgenden Begriffe verwandt:

Zustand: (State)	Je nachdem ob beispielsweise die Endstufe eingeschaltet oder ein Fehler aufgetreten ist befindet sich der Regler in verschiedenen Zuständen. Die unter CANopen definierten Zustände werden im Laufe des Kapitels vorgestellt. Beispiel: **SWITCH_ON_DISABLED**
Zustandsübergang (State Transition)	Ebenso wie die Zustände ist es unter CANopen ebenfalls definiert, wie man von einem Zustand zu einem anderen gelangt (z.B. um einen Fehler zu quittieren). Zustandsübergänge werden vom Host durch Setzen von Bits im **controlword** ausgelöst oder intern durch den Regler, wenn dieser beispielsweise einen Fehler erkennt.
Kommando (Command)	Zum Auslösen von Zustandsübergängen müssen bestimmte Kombinationen von Bits im **controlword** gesetzt werden. Eine solche Kombination wird als Kommando bezeichnet. Beispiel: **Enable Operation**
Zustandsdiagramm (State Machine)	Die Zustände und Zustandsübergänge bilden zusammen das Zustandsdiagramm, also die Übersicht über alle Zustände und die von dort möglichen Übergänge.

7.1.2 Das Zustandsdiagramm des Reglers (State Machine)

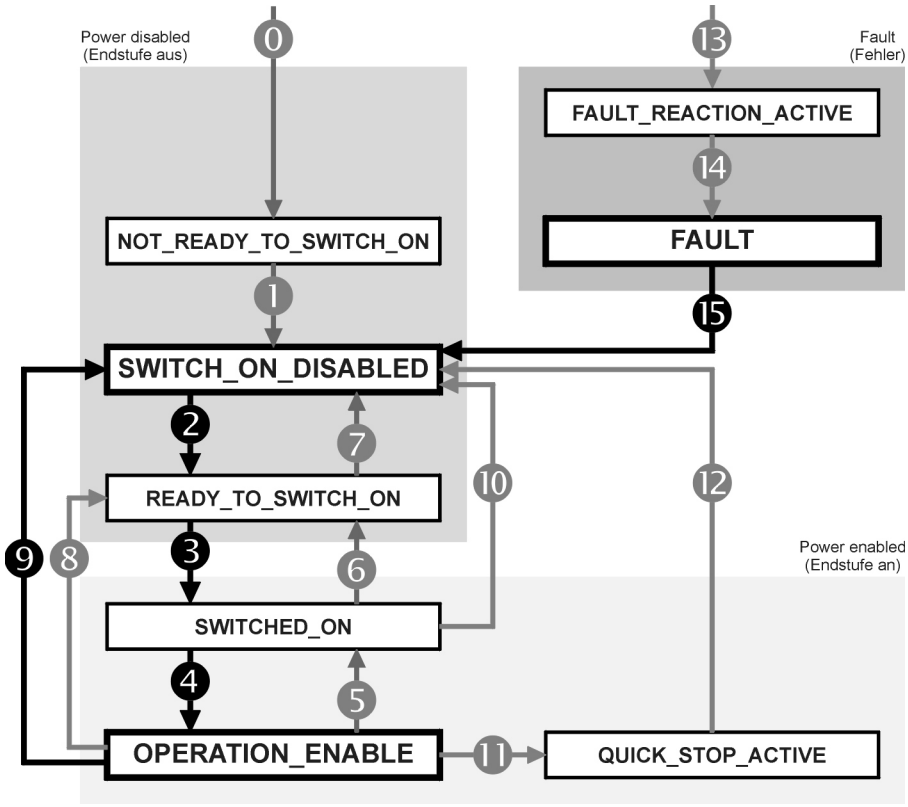


Abbildung 7.9: Zustandsdiagramm des Reglers

Das Zustandsdiagramm kann grob in drei Bereiche aufgeteilt werden: „Power Disabled“ bedeutet, dass die Endstufe ausgeschaltet ist und „Power Enabled“ dass die Endstufe eingeschaltet ist. Im Bereich „Fault“ sind die zur Fehlerbehandlung notwendigen Zustände zusammengefasst.

Die wichtigsten Zustände des Reglers sind im Diagramm hervorgehoben dargestellt. Nach dem Einschalten initialisiert sich der Regler und erreicht schließlich den Zustand **SWITCH_ON_DISABLED**. In diesem Zustand ist die CAN-Kommunikation voll funktionsfähig und der Regler kann parametriert werden (z.B. die Betriebsart „Drehzahlregelung“ eingestellt werden). Die Endstufe ist ausgeschaltet und die Welle ist somit frei drehbar. Durch die Zustandsübergänge 2, 3, 4 – was im Prinzip der CAN-Reglerfreigabe entspricht – gelangt man in den Zustand **OPERATION_ENABLE**. In diesem Zustand ist die Endstufe eingeschaltet und der Motor wird gemäß der eingestellten Betriebsart geregelt Stellen Sie daher vorher unbedingt sicher, dass der Antrieb richtig parametriert ist und ein entsprechender Sollwert gleich Null ist.

Tritt ein Fehler auf so wird (egal aus welchem Zustand) letztlich in den Zustand **FAULT** verzweigt. Je nach Schwere des Fehlers können vorher noch bestimmte Aktionen, wie z.B. eine Notbremsung ausgeführt werden (**FAULT_REACTION_ACTIVE**).

Um die genannten Zustandsübergänge auszuführen müssen bestimmte Bitkombinationen im **controlword** (siehe unten) gesetzt werden. Die unteren 4 Bits des **controlwords** werden gemeinsam ausgewertet, um einen Zustandsübergang auszulösen. Im Folgenden werden zunächst nur die wichtigsten Zustandsübergänge 2, 3, 4, 9 und 15 erläutert. Eine Tabelle aller möglichen Zustände und Zustandsübergänge findet sich am Ende dieses Kapitels.

Die folgende Tabelle enthält in der 1. Spalte den gewünschten Zustandsübergang und in der 2. Spalte die dazu notwendigen Voraussetzungen (Meistens ein Kommando durch den Host, hier mit Rahmen dargestellt). Wie dieses Kommando erzeugt wird, d.h. welche Bits im **controlword** zu setzen sind, ist in der 3. Spalte ersichtlich (x = nicht relevant).

Nr.	Wird durchgeführt wenn	Bitkombination (controlword)					Aktion
		Bit	3	2	1	0	
2	Reglerfreig. vorh. + Kommando **Shutdown**	**Shutdown** =	**x**	**1**	**1**	**0**	Keine
3	Kommando **Switch On**	**Switch On** =	**x**	**1**	**1**	**1**	Einschalten der Endstufe
4	Kommando **Enable Operation**	**Enable Operation** =	**1**	**1**	**1**	**1**	Regelung gemäß eingestellter Betriebsart
9	Kommando **Disable Voltage**	**Disable Voltage** =	**x**	**x**	**0**	**x**	Endstufe wird gesperrt. Motor ist frei drehbar.
15	Fehler behoben+ Kommando **Fault Reset**	**Fault Reset** =	Bit 7 = ↑				Fehler quittieren

Abbildung 7.10: Wichtigste Zustandsübergänge des Reglers

BEISPIEL

Nachdem der Regler parametriert wurde, soll der Regler „freigegeben", d.h. die Endstufe eingeschaltet werden:

1.) Der Regler ist im Zustand **SWITCH_ON_DISABLED**

2.) Der Regler soll in den Zustand **OPERATION_ENABLE**

3.) Laut Zustandsdiagramm (Abbildung 7.9) sind die Übergänge 2, 3 und 4 auszuführen.

4.) Aus Abbildung 7.10 folgt:

Übergang 2: controlword = 0006_h Neuer Zustand: READY_TO_SWITCH_ON *1)

Übergang 3: controlword = 0007_h Neuer Zustand: SWITCHED_ON *1)

Übergang 4: controlword = $000F_h$ Neuer Zustand: OPERATION_ENABLE *1)

Hinweise:

1.) Das Beispiel geht davon aus, dass keine weiteren Bits im **controlword** gesetzt sind (Für die Übergänge sind ja nur die Bits 0..3 wichtig).

2.) Die Übergänge 3 und 4 können zusammengefasst werden, indem das **controlword** gleich auf $000F_h$ gesetzt wird. Für den Zustandsübergang 2 ist das gesetzte Bit 3 nicht relevant.

*1) Der Host muss warten, bis der Zustand im **statusword** zurückgelesen werden kann. Dieses wird weiter unten noch ausführlich erläutert.

7.1.2.1 Zustandsdiagramm: Zustände

In der folgenden Tabelle sind alle Zustände und deren Bedeutung aufgeführt:

Name	Bedeutung
NOT_READY_TO_SWITCH_ON	Der Regler führt einen Selbsttest durch. Die CAN-Kommunikation arbeitet noch nicht.
SWITCH_ON_DISABLED	Der Regler hat seinen Selbsttest abgeschlossen. CAN-Kommunikation ist möglich.
READY_TO_SWITCH_ON	Der Regler wartet bis der digitale Eingang DIN9 „Reglerfreigabe“ an 24 V liegt. (Reglerfreigabelogik „Digitaler Eingang und CAN“).
SWITCHED_ON *1)	Die Endstufe kann eingeschaltet werden.
OPERATION_ENABLE *1)	Der Motor liegt an Spannung und wird entsprechend der Betriebsart geregelt.
QUICKSTOP_ACTIVE *1)	Die **Quick Stop Function** wird ausgeführt (siehe: **quick_stop_option_ code**). Der Motor liegt an Spannung und wird entsprechend der **Quick Stop Function** geregelt.
FAULT_REACTION_ACTIVE *1)	Es ist ein Fehler aufgetreten. Bei kritischen Fehlern wird sofort in den Status **Fault** gewechselt. Ansonsten wird die im **fault_reaction_option_code** vorgegebene Aktion ausgeführt. Der Motor liegt an Spannung und wird entsprechend der **Fault Reaction Function** geregelt.
FAULT	Es ist ein Fehler aufgetreten. Der Motor ist spannungsfrei.

***1)** Die Endstufe ist eingeschaltet.

7.1.2.2 Zustandsdiagramm: Zustandsübergänge

In der folgenden Tabelle sind alle Zustände und deren Bedeutung aufgeführt:

Nr.	Wird durchgeführt wenn	Bitkombination (controlword)					Aktion
		Bit	3	2	1	0	
0	Eingeschaltet o. Reset erfolgt	interner Übergang					Selbsttest ausführen
1	Selbsttest erfolgreich	interner Übergang					Aktivierung der CAN-Kommunikation
2	Reglerfreig. vorh. + Kommando **Shutdown**	**Shutdown** =	**x**	**1**	**1**	**0**	-
3	Kommando **Switch On**	**Switch On** =	**x**	**1**	**1**	**1**	-
4	Kommando **Enable Operation**	**Enable Operation** =	**1**	**1**	**1**	**1**	Die Endstufe wird eingeschaltet
5	Kommando **Disable Operation**	**Disable Operation** =	**0**	**1**	**1**	**1**	Endstufe wird gesperrt. Motor ist frei drehbar
6	Kommando **Shutdown**	**Shutdown** =	**x**	**1**	**1**	**0**	Endstufe wird gesperrt. Motor ist frei drehbar
7	Kommando **Quick Stop**	**Quick Stop** =	**x**	**0**	**1**	**x**	-
8	Kommando **Shutdown**	**Shutdown** =	**x**	**1**	**1**	**0**	Endstufe wird gesperrt. Motor ist frei drehbar
9	Kommando **Disable Voltage**	**Disable Voltage** =	**x**	**x**	**0**	**x**	Endstufe wird gesperrt.

Nr.	Wird durchgeführt wenn	Bitkombination (controlword)					Aktion
		Bit	3	2	1	0	
							Motor ist frei drehbar.
10	Kommando **Disable Voltage**	**Disable Voltage** =	**x**	**x**	**0**	**x**	Endstufe wird gesperrt. Motor ist frei drehbar
11	Kommando **Quick Stop**	**Quick Stop** =	**x**	**0**	**1**	**x**	Es wird eine Bremsung eingeleitet.
12	Bremsung beendet o. Kommando **Disable Voltage**	**Disable Voltage** =	**x**	**x**	**0**	**x**	Endstufe wird gesperrt. Motor ist frei drehbar
13	Fehler aufgetreten	interner Übergang					Bei unkritischen Fehlern Reaktion gemäß **fault_ reaction_option_code**. Bei kritischen Fehlern folgt Übergang 14
14	Fehlerbehandlung ist beendet	interner Übergang					Endstufe wird gesperrt. Motor ist frei drehbar
15	Fehler behoben+ Kommando **Fault Reset**	**Fault Reset** =	Bit 7 = ⎍				Fehler quittieren (bei steigender Flanke)

Endstufe gesperrt...

...bedeutet, dass die Leistungshalbleiter (Transistoren) nicht mehr angesteuert werden. **Wenn dieser Zustand bei einem drehenden Motor eingenommen wird, so trudelt dieser ungebremst aus.** Eine eventuell vorhandene mechanische Motorbremse wird hierbei automatisch angezogen.

Vorsicht: Das Signal garantiert nicht, dass der Motor wirklich spannungsfrei ist.

Endstufe freigegeben...

...bedeutet, dass der Motor entsprechend der gewählten Betriebsart angesteuert und geregelt wird. Eine eventuell vorhandene mechanische Motorbremse wird automatisch gelöst. Bei einem Defekt oder einer Fehlparametrierung (Motorstrom, Polzahl, Resolveroffsetwinkel etc.) kann es zu einem unkontrollierten Verhalten des Antriebes kommen.

7.1.3 controlword (Steuerwort)

7.1.3.1 Objekt 6040_h: controlword

Mit dem **controlword** kann der aktuelle Zustand des Reglers geändert bzw. direkt eine bestimmte Aktion (z.B. Start der Referenzfahrt) ausgelöst werden. Die Funktion der Bits 4, 5, 6 und 8 hängt von der aktuellen Betriebsart (**modes_of_operation**) des Reglers ab, die nach diesem Kapitel erläutert wird.

Index	**6040_h**
Name	**controlword**
Object Code	VAR
Data Type	UINT16

Access	rw
PDO Mapping	yes
Units	--
Value Range	--
Default Value	0

Bit	Wert	Funktion
0	0001_h	Steuerung der Zustandsübergänge. (Diese Bits werden gemeinsam ausgewertet)
1	0002_h	
2	0004_h	
3	0008_h	
4	0010_h	new_set_point / start_homing_operation / enable_ip_mode
5	0020_h	change_set_immediatly
6	0040_h	absolute / relative
7	0080_h	reset_fault
8	0100_h	halt
9	0200_h	reserved set to 0
10	0400_h	reserved set to 0
11	0800_h	reserved set to 0
12	1000_h	reserved set to 0
13	2000_h	reserved set to 0
14	4000_h	reserved set to 0
15	8000_h	reserved set to 0

Tabelle 7.1: Bitbelegung des controlword

Wie bereits umfassend beschrieben können mit den Bits 0..3 Zustandsübergänge ausgeführt werden. Die dazu notwendigen Kommandos sind hier noch einmal in einer Übersicht

dargestellt. Das Kommando **Fault Reset** wird durch einen positiven Flankenwechsel (von 0 nach 1) von Bit 7 erzeugt.

Kommando:	Bit 7	Bit 3	Bit 2	Bit 1	Bit 0
	0080_h	0008_h	0004_h	0002_h	0001_h
Shutdown	×	×	**1**	**1**	**0**
Switch On	×	×	**1**	**1**	**1**
Disable Voltage	×	×	×	**0**	×
Quick Stop	×	×	**0**	**1**	×
Disable Operation	×	**0**	**1**	**1**	**1**
Enable Operation	×	**1**	**1**	**1**	**1**
Fault Reset	↑	×	×	×	×

Tabelle 7.2: Übersicht aller Kommandos (× = nicht relevant)

Da einige Statusänderungen einen gewissen Zeitraum beanspruchen, müssen alle über das **controlword** ausgelösten Statusänderungen über das **statusword** zurückgelesen werden. Erst wenn der angeforderte Status auch im **statusword** gelesen werden kann, darf über das **controlword** ein weiteres Kommando eingeschrieben werden.

Nachfolgend sind die restlichen Bits des **controlwords** erläutert. Einige Bits haben dabei je nach Betriebsart (**modes_of_operation**), d.h. ob der Regler z.B. drehzahl- oder momentengeregelt wird, unterschiedliche Bedeutung:

Bit 4 Abhängig von **modes_of_operation:**

new_set_point Im **Profile Position Mode**:

Eine steigende Flanke signalisiert dem Regler, dass ein neuer Fahrauftrag übernommen werden soll. Siehe dazu unbedingt auch Kapitel 8.3.

start_homing_operation Im **Homing Mode**:

Eine steigende Flanke bewirkt, dass die parametrierte Referenzfahrt gestartet wird. Eine fallende Flanke bricht eine laufende Referenzfahrt vorzeitig ab.

enable_ip_mode Im **Interpolated Position Mode**:

Dieses Bit muss gesetzt werden, wenn die Interpolations-Datensätze ausgewertet werden sollen. Es wird durch das Bit **ip_mode_active** im **statusword** quittiert. Siehe hierzu unbedingt auch Kapitel 8.4

Bit 5	**change_set_immediatly**	Nur im **Profile Position Mode**: Wenn dieses Bit nicht gesetzt ist, so wird bei einem neuen Fahrauftrag zuerst ein eventuell laufender abgearbeitet und erst dann mit dem neuen begonnen. Bei gesetztem Bit wird eine laufende Positionierung sofort abgebrochen und durch den neuen Fahrauftrag ersetzt. Siehe dazu unbedingt auch Kapitel 8.3.
Bit 6	**relative**	Nur im **Profile Position Mode**: Bei gesetztem Bit bezieht der Regler die Zielposition (**target_position***)* des aktuellen Fahrauftrages auf die Sollposition (**position_demand_value**) des Lagereglers.
Bit 7	**reset_fault**	
		Beim Übergang von Null auf Eins versucht der Regler die vorhandenen Fehler zu quittieren. Dies gelingt nur, wenn die Ursache für den Fehler behoben wurde.
Bit 8		Abhängig von **modes_of_operation:**
	halt	Im **Profile Position Mode**: Bei gesetztem Bit wird die laufende Positionierung abgebrochen. Gebremst wird hierbei mit der **profile_deceleration**. Nach Beendigung des Vorgangs wird im **statusword** das Bit **target_reached** gesetzt. Das Löschen des Bits hat keine Auswirkung.
	halt	Im **Profile Velocity Mode**: Bei gesetztem Bit wird die Drehzahl auf Null abgesenkt. Gebremst wird hierbei mit der **profile_deceleration**. Das Löschen des Bits bewirkt, dass der Regler wieder beschleunigt.
	halt	Im **Profile Torque Mode**: Bei gesetztem Bit wird das Drehmoment auf Null gesetzt. Das Löschen des Bits bewirkt, dass der Regler wieder beschleunigt.
	halt	Im **Homing Mode**: Bei gesetztem Bit wird die laufende Referenzfahrt abgebrochen. Das Löschen des Bits hat keine Auswirkung.

7.1.4 Auslesen des Reglerzustands

Ähnlich wie über die Kombination mehrerer Bits des **controlwords** verschiedene Zustandsübergänge ausgelöst werden können, kann über die Kombination verschiedener Bits des **statusword** ausgelesen werden, in welchem Zustand sich der Regler befindet.
Die folgende Tabelle listet die möglichen Zustände des Zustandsdiagramms sowie die zugehörige Bitkombination auf, mit der sie im **statusword** angezeigt werden.

Zustand	Bit 6	Bit 5	Bit 3	Bit 2	Bit 1	Bit 0	Maske	Wert
	0040_h	0020_h	0008_h	0004_h	0002_h	0001_h		
NOT_READY_TO_SWITCH_ON	0	×	0	0	0	0	$004F_h$	0000_h
SWITCH_ON_DISABLED	1	×	0	0	0	0	$004F_h$	0040_h
READY_TO_SWITCH_ON	0	1	0	0	0	1	$006F_h$	0021_h
SWITCHED_ON	0	1	0	0	1	1	$006F_h$	0023_h
OPERATION_ENABLE	0	1	0	1	1	1	$006F_h$	0027_h
FAULT	0	×	1	1	1	1	$004F_h$	$000F_h$
FAULT_REACTION_ACTIVE	0	×	1	1	1	1	$004F_h$	$000F_h$
QUICK_STOP_ACTIVE	0	0	0	1	1	1	$006F_h$	0007_h

Tabelle 7.3: Gerätestatus (× = nicht relevant)

BEISPIEL

Das obige Beispiel zeigt, welche Bits im **controlword** gesetzt werden müssen, um den Regler freizugeben. Jetzt soll dabei der neu eingeschriebene Zustand aus dem **statusword** ausgelesen werden:
Übergang von **SWITCH_ON_DISABLED** zu **OPERATION_ENABLE**:

1.) Zustandsübergang 2 ins **controlword** schreiben.
2.) Warten, bis der Zustand **READY_TO_SWITCH_ON** im **statusword** angezeigt wird.

Übergang 2: controlword = 0006_h Warten bis (statusword & $006F_h$) = 0021_h *[1)]

3.) Zustandsübergang 3 und 4 können zusammengefasst ins **controlword** geschrieben werden.
4.) Warten, bis der Zustand **OPERATION_ENABLE** im **statusword** angezeigt wird.

Übergang 3+4: controlword = $000F_h$ Warten bis (statusword & $006F_h$) = 0027_h *[1)]

Hinweis:
Das Beispiel geht davon aus, dass keine weiteren Bits im **controlword** gesetzt sind (Für die Übergänge sind ja nur die Bits 0..3 wichtig).

*[1)] Für die Identifizierung der Zustände müssen auch nicht gesetzte Bits ausgewertet werden (siehe Tabelle). Daher muss das **statusword** entsprechend maskiert werden.

7.1.5 statusword (Statuswort)

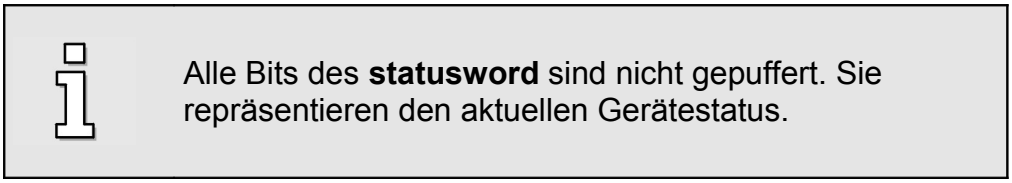
7.1.5.1 Objekt 6041_h: statusword

Index	**6041_h**
Name	**statusword**
Object Code	VAR
Data Type	UINT16

Access	ro
PDO Mapping	yes
Units	--
Value Range	--
Default Value	--

Bit	Wertigkeit	Name
0	0001_h	Zustand des Reglers (s. Tabelle 7.3). (Diese Bits müssen gemeinsam ausgewertet werden)
1	0002_h	
2	0004_h	
3	0008_h	
4	0010_h	voltage_enabled
5	0020_h	Zustand des Reglers (s. Tabelle 7.3).
6	0040_h	
7	0080_h	warning
8	0100_h	unused
9	0200_h	remote
10	0400_h	target_reached
11	0800_h	internal_limit_active
12	1000_h	set_point_acknowledge / speed_0 / homing_attained / ip_mode_active
13	2000_h	following_error / homing_error
14	4000_h	unused
15	8000_h	reserved

Tabelle 7.4: Bitbelegung im statusword



Neben dem Reglerstatus werden im **statusword** diverse Ereignisse angezeigt, d.h. jedem Bit ist ein bestimmtes Ereignis wie z.B. Schleppfehler zugeordnet. Die einzelnen Bits haben dabei folgende Bedeutung:

Bit 4	**voltage_enabled**	
		Dieses Bit ist gesetzt, wenn die Endstufentransistoren eingeschaltet sind.
		 **ACHTUNG:** Bei einem Defekt kann der Motor trotzdem unter Spannung stehen.
Bit 5	**quick_stop**	
		Bei gelöschtem Bit führt der Antrieb einen **Quick Stop** gemäß **quick_stop_option_code** aus.
Bit 7	**warning**	
		Dieses Bit ist undefiniert. Es darf nicht ausgewertet werden.
Bit 8	**manufacturer specific**	
		Dieses Bit ist unbenutzt und darf nicht ausgewertet werden.
Bit 9	**remote**	
		Dieses Bit zeigt an, dass die Endstufe des Reglers über das CAN-Netzwerk freigegeben werden kann. Es ist gesetzt, wenn die Reglerfreigabelogik über das Objekt **enable_logic** entsprechend eingestellt ist.
Bit 10		Abhängig von **modes_of_operation:**
	target_reached	Im **Profile Position Mode**: Das Bit wird gesetzt, wenn die aktuelle Zielposition erreicht ist und sich die aktuelle Position (**position_ actual_value**) im parametrierten Positionsfenster (**position_window**) befindet. Außerdem wird es gesetzt, wenn der Antrieb bei gesetztem **Halt**-Bit zum Stillstand kommt. Es wird gelöscht, sobald ein neues Ziel vorgegeben wird.
	target_reached	Im **Profile Velocity Mode**: Das Bit wird gesetzt, wenn sich die Drehzahl (**velocity_actual_value**) des Antriebs im Toleranzfenster befindet. Dieses Toleranzfenster ist über D2SC einstellbar.
Bit 11	**internal_limit_active**	
		Dieses Bit zeigt an, dass die I^2t-Begrenzung aktiv ist.
Bit 12		Abhängig von **modes_of_operation:**

	set_point_acknowledge	Im **Profile Position Mode**: Dieses Bit wird gesetzt, wenn der Regler das gesetzte Bit **new_set_point** im **controlword** erkannt hat. Es wird wieder gelöscht, nachdem das Bit **new_set_point** im **controlword** auf Null gesetzt wurde. Siehe dazu unbedingt auch Kapitel 8.3.
	speed_0	Im **Profile Velocity Mode**: Dieses Bit wird gesetzt, wenn sich die aktuelle Ist-Drehzahl (**velocity_actual_value**) des Antriebes im zugehörigen Toleranzfenster befindet.
	homing_attained	Im **Homing Mode**: Dieses Bit wird gesetzt, wenn die Referenzfahrt ohne Fehler beendet wurde.
	ip_mode_active	Im **Interpolated Position Mode**: Dieses Bit zeigt an, dass die Interpolation aktiv ist und die Interpolations-Datensätze ausgewertet werden. Es wird gesetzt, wenn dies durch das Bit **enable_ip_mode** im **controlword** angefordert wurde. Siehe hierzu unbedingt auch Kapitel 8.4
Bit 13		Abhängig von **modes_of_operation:**
	following_error	Im **Profile Position Mode**: Dieses Bit wird gesetzt, wenn die aktuelle Ist-Position (**position_actual_value**) von der Soll-Position (**position_demand_value**) soweit abweicht, dass die Differenz außerhalb des parametrierten Toleranzfensters liegt (**following_error_window, following_error_time_out**).
	homing_error	Im **Homing Mode**: Dieses Bit wird gesetzt, wenn die Referenzfahrt unterbrochen wird (**Halt**-Bit), beide Endschalter gleichzeitig ansprechen oder die bereits zurückgelegte Endschaltersuchfahrt größer als der vorgegebene Positionierraum ist.
Bit 14	**unused**	
		Dieses Bit ist unbenutzt und darf nicht ausgewertet werden.
Bit 15	**reserved**	herstellerspezifisch
		Dieses Bit darf nicht ausgewertet werden.

8 Betriebsarten

8.1 Einstellen der Betriebsart

8.1.1 Übersicht

Der Antriebsregler kann in eine Vielzahl von Betriebsarten versetzt werden. Nur einige sind unter CANopen detailliert spezifiziert:

- momentengeregelter Betrieb — profile torque mode
- drehzahlgeregelter Betrieb — profile velocity mode
- Referenzfahrt — homing mode
- Positionierbetrieb — profile position mode
- Synchrone Positionsvorgabe — interpolated position mode

8.1.2 Beschreibung der Objekte

8.1.2.1 In diesem Kapitel behandelte Objekte

Index	Objekt	Name	Typ	Attr.
6060_h	VAR	modes_of_operation	INT8	wo
6061_h	VAR	modes_of_operation_display	INT8	ro

8.1.2.2 Objekt 6060_h: modes_of_operation

Mit dem Objekt **modes_of_operation** wird die Betriebsart des Reglers eingestellt. Mit dem Lesezugriff wird die zuletzt über CAN geschriebene Betriebsart geliefert. Mit diesem Lesezugriff wird nicht die aktuelle Betriebsart ausgelesen.

Index	**6060_h**
Name	**modes_of_operation**
Object Code	VAR
Data Type	INT8

Access	rw
PDO Mapping	yes
Units	--
Value Range	1, 3, 4, 6, 7
Default Value	--

Wert	Aktion
1	Profile Position Mode (Lageregler mit Positionierbetrieb)
3	Profile Velocity Mode (Drehzahlregler mit Sollwertrampe)
4	Torque Profile Mode (Momentenregler mit Sollwertrampe)
6	Homing Mode (Referenzfahrt)
7	Interpolated Position Mode

Die aktuelle Betriebsart kann nur im Objekt **modes_of_operation_display** gelesen werden !
Da ein Wechsel der Betriebsart etwas Zeit in Anspruch nehmen kann, **muss** solange gewartet werden, bis der neu ausgewählte Modus im Objekt **modes_of_operation_display** erscheint.

8.1.2.3 Objekt 6061_h: modes_of_operation_display

Im Objekt **modes_of_operation_display** kann die aktuelle Betriebsart des Reglers gelesen werden. Eine interne Betriebsart wird dabei ausgelesen, wenn interne Selektoren so eingestellt sind, dass kein Betrieb über CANopen möglich ist, bis eine CANopen spezifische Betriebsart eingestellt wird.

Index	**6061_h**
Name	**modes_of_operation_display**
Object Code	VAR
Data Type	INT8

Access	ro
PDO Mapping	yes
Units	--
Value Range	-1, 1, 3, 4, 6, 7, -11, -12, -13, -14, -15
Default Value	3

Wert	Aktion
-1	Unbekannte Betriebsart / Betriebsartenwechsel
1	Profile Position Mode (Lageregler mit Positionierbetrieb)
3	Profile Velocity Mode (Drehzahlregler mit Sollwertrampe)
4	Torque Profile Mode (Momentenregler mit Sollwertrampe)
6	Homing Mode (Referenzfahrt)
7	Interpolated Position Mode
-11	Interne Positioniersteuerung
-12	Interne Drehzahlregelung ohne Sollwertrampe
-13	Interne Drehzahlregelung mit Sollwertrampe
-14	Interne Drehmomentregelung
-15	Interne Lageregelung

Die Betriebsart kann nur über das Objekt **modes_of_operation** gesetzt werden.
Da ein Wechsel der Betriebsart etwas Zeit in Anspruch nehmen kann, **muss** solange gewartet werden, bis der neu ausgewählte Modus im Objekt **modes_of_operation_display** erscheint. Während dieses Zeitraumes kann es passieren, dass kurzzeitig ungültige Betriebsarten (-1) angezeigt werden.

8.2 Betriebsart Referenzfahrt (Homing Mode)

8.2.1 Übersicht

In diesem Kapitel wird beschrieben, wie der Antriebsregler die Anfangsposition sucht (auch Bezugspunkt, Referenzpunkt oder Nullpunkt genannt). Es gibt verschiedene Methoden diese Position zu bestimmen, wobei entweder die Endschalter am Ende des Positionierbereiches benutzt werden können oder aber ein Referenzschalter (Nullpunkt-Schalter) innerhalb des möglichen Verfahrweges. Um eine möglichst große Reproduzierbarkeit zu erreichen, kann bei einigen Methoden der Nullimpuls des verwendeten Winkelgebers (Resolver, Inkrementalgeber etc.) mit einbezogen werden.

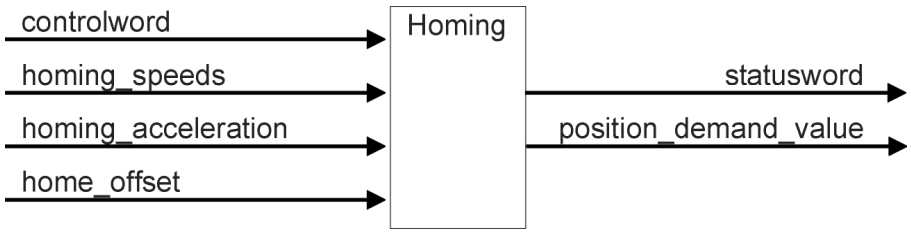


Abbildung 8.1: Die Referenzfahrt

Der Benutzer kann die Geschwindigkeit, Beschleunigung und die Art der Referenzfahrt bestimmen. Mit dem Objekt **home_offset** kann die Nullposition des Antriebs an eine beliebige Stelle verschoben werden.
Es gibt zwei Referenzfahrgeschwindigkeiten. Die höhere Suchgeschwindigkeit (**speed_during_search_for_switch**) wird benutzt, um den Endschalter bzw. den Referenzschalter zu finden. Um dann die Position der betreffenden Schaltflanke exakt bestimmen zu können, wird auf die Kriechgeschwindigkeit (**speed_during_search_for_zero**) umgeschaltet.

Die Fahrt auf die Nullposition ist unter CANopen in der Regel nicht Bestandteil der Referenzfahrt. Sind dem Regler alle erforderlichen Größen bekannt (z.B. weil er die Lage des Nullimpulses bereits kennt), wird keine physikalische Bewegung ausgeführt.

8.2.2 Beschreibung der Objekte

8.2.2.1 In diesem Kapitel behandelte Objekte

Index	Objekt	Name	Typ	Attribute
$607C_h$	VAR	home_offset	INT32	rw
6098_h	VAR	homing_method	INT8	rw
6099_h	ARRAY	homing_speeds	UINT32	rw
$609A_h$	VAR	homing_acceleration	UINT32	rw

8.2.2.2 Betroffene Objekte aus anderen Kapiteln

Index	Objekt	Name	Typ	Kapitel
6040_h	VAR	controlword	UINT16	7 Gerätesteuerung
6041_h	VAR	statusword	UINT16	7 Gerätesteuerung

8.2.2.3 Objekt $607C_h$: home_offset

Das Objekt **home_offset** legt die Verschiebung der Nullposition gegenüber der ermittelten Referenzposition fest.

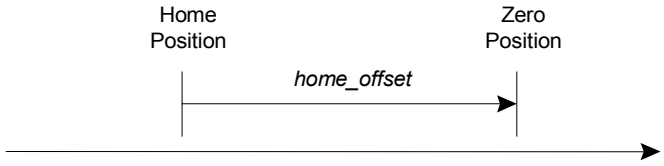


Abbildung 8.2: Home Offset

Index	**$607C_h$**
Name	**home_offset**
Object Code	VAR
Data Type	INT32

Access	rw
PDO Mapping	yes
Units	position units
Value Range	--
Default Value	0

8.2.2.4 Objekt 6098_h: homing_method

Für eine Referenzfahrt werden eine Reihe unterschiedlicher Methoden bereitgestellt. Über das Objekt **homing_method** kann die für die Applikation benötigte Variante ausgewählt werden. Es gibt drei mögliche Referenzfahrt-Signale: den negativen und positiven Endschalter, und den (periodischen) Nullimpuls des Winkelgebers. Außerdem kann der Regler sich ganz ohne zusätzliches Signal auf den negativen oder positiven Anschlag referenzieren. Wenn über das Objekt **homing_method** eine Methode zum Referenzieren bestimmt wird, so werden hiermit folgende Einstellungen gemacht:

- Die Referenzquelle (neg./pos. Endschalter, neg. / pos. Anschlag)
- Die Richtung und der Ablauf der Referenzfahrt
- Die Art der Auswertung des Nullimpulses vom verwendeten Winkelgeber

Index	**6098_h**
Name	**homing_method**
Object Code	VAR
Data Type	INT8

Access	rw
PDO Mapping	yes
Units	
Value Range	-18, -17, -2, -1, 1, 2, 17, 18, 32, 33, 34
Default Value	17

Wert	Richtung	Ziel	Bezugspunkt für Null
-18	positiv	Anschlag	Anschlag
-17	negativ	Anschlag	Anschlag
-2	positiv	Anschlag	Nullimpuls
-1	negativ	Anschlag	Nullimpuls
1	negativ	Endschalter	Nullimpuls
2	positiv	Endschalter	Nullimpuls
17	negativ	Endschalter	Endschalter
18	positiv	Endschalter	Endschalter
32	negativ	Nullimpuls	Nullimpuls
33	positiv	Nullimpuls	Nullimpuls
34		Keine Fahrt	Aktuelle Ist-Position

Der Ablauf der einzelnen Methoden ist im Folgenden ausführlich erläutert.

8.2.2.5 Objekt 6099_h: homing_speeds

Dieses Objekt bestimmt die Geschwindigkeiten, die während der Referenzfahrt benutzt werden.

Index	**6099_h**
Name	**homing_speeds**
Object Code	ARRAY
No. of Elements	2
Data Type	UINT32

Sub-Index	**01_h**
Description	**speed_during_search_for_switch**
Access	rw
PDO Mapping	yes
Units	speed units
Value Range	--
Default Value	100 min^{-1}

Sub-Index	**02_h**
Description	**speed_during_search_for_zero**
Access	rw
PDO Mapping	yes
Units	speed units
Value Range	--
Default Value	10 min^{-1}

8.2.2.6 Objekt $609A_h$: homing_acceleration

Das Objekt **homing_acceleration** legt die Beschleunigung fest, die während der Referenzfahrt für alle Beschleunigungs- und Bremsvorgänge verwendet wird.

Index	**$609A_h$**
Name	**homing_acceleration**
Object Code	VAR
Data Type	UINT32

Access	rw
PDO Mapping	yes
Units	acceleration units
Value Range	--
Default Value	250 min^{-1} / s

Intern enthält der Regler 4 Beschleunigungen für die Referenzfahrt. 2 für Suchfahrt (Beschleunigen und Bremsen) und 2 für Kriechfahrt (Beschleunigen und Bremsen). Wenn man sicher gehen will, dass in allen der gleiche Wert steht, muss man homing_acceleration einmal parametrieren. Beim Auslesen des Objektes wird die Beschleunigung der Kriechfahrt geliefert.

8.2.3 Referenzfahrt-Abläufe

Die verschiedenen Referenzfahrt-Methoden sind in den folgenden Abbildungen dargestellt. Die eingekreisten Nummern entsprechen dem im Objekt **homing_method** einzutragenden Code.

8.2.3.1 Methode 1:Negativer Endschalter mit Nullimpulsauswertung

Bei dieser Methode bewegt sich der Antrieb zunächst relativ schnell in negativer Richtung, bis er den negativen Endschalter erreicht. Dieses wird im Diagramm durch die steigende Flanke dargestellt. Danach fährt der Antrieb langsam zurück und sucht die genaue Position des Endschalters. Die Nullposition bezieht sich auf den ersten Nullimpuls des Winkelgebers in positiver Richtung vom Endschalter.

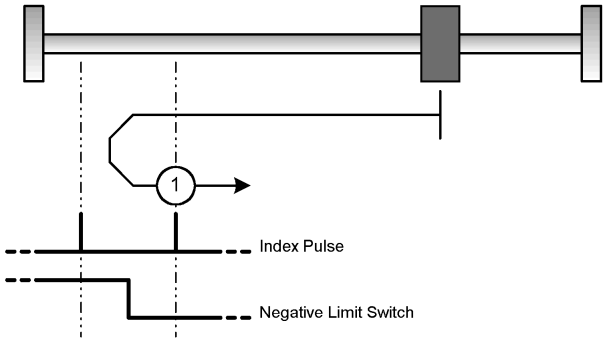


Abbildung 8.3: Referenzfahrt auf den negativen Endschalter mit Auswertung des Nullimpulses

8.2.3.2 Methode 2:Positiver Endschalter mit Nullimpulsauswertung

Bei dieser Methode bewegt sich der Antrieb zunächst relativ schnell in positiver Richtung, bis er den positiven Endschalter erreicht. Dieses wird im Diagramm durch die steigende Flanke dargestellt. Danach fährt der Antrieb langsam zurück und sucht die genaue Position des Endschalters. Die Nullposition bezieht sich auf den ersten Nullimpuls des Winkelgebers in negativer Richtung vom Endschalter.

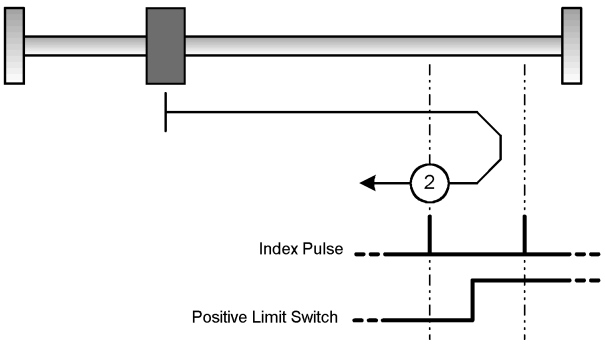


Abbildung 8.4: Referenzfahrt auf den positiven Endschalter mit Auswertung des Nullimpulses

8.2.3.3 Methode 17: Referenzfahrt auf den negativen Endschalter

Bei dieser Methode bewegt sich der Antrieb zunächst relativ schnell in negativer Richtung, bis er den negativen Endschalter erreicht. Dieses wird im Diagramm durch die steigende Flanke dargestellt. Danach fährt der Antrieb langsam zurück und sucht die genaue Position des Endschalters. Die Nullposition bezieht sich auf die fallende Flanke vom negativen Endschalter.

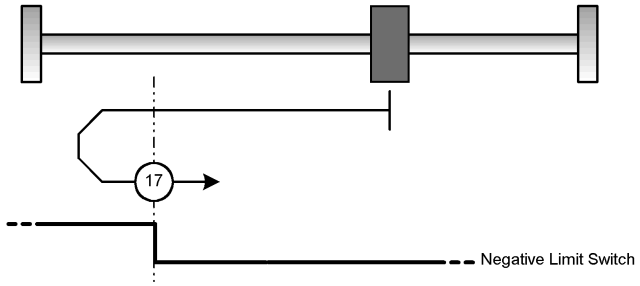


Abbildung 8.5: Referenzfahrt auf den negativen Endschalter

8.2.3.4 Methode 18: Referenzfahrt auf den positiven Endschalter

Bei dieser Methode bewegt sich der Antrieb zunächst relativ schnell in positiver Richtung, bis er den positiven Endschalter erreicht. Dieses wird im Diagramm durch die steigende Flanke dargestellt. Danach fährt der Antrieb langsam zurück und sucht die genaue Position des Endschalters. Die Nullposition bezieht sich auf die fallende Flanke vom positiven Endschalter.

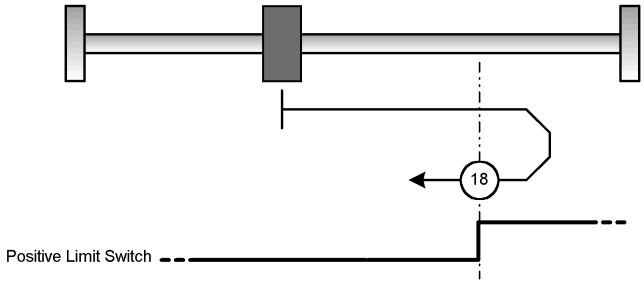


Abbildung 8.6: Referenzfahrt auf den positiven Endschalter

8.2.3.5 Methode –1: negativer Anschlag mit Nullimpulsauswertung

Bei dieser Methode bewegt sich der Antrieb in negativer Richtung, bis er den Anschlag erreicht. Hierbei steigt das I^2t-Integral des Motors auf maximal 90%. Der Anschlag muss mechanisch so dimensioniert sein, dass er bei dem parametrierten Maximalstrom keinen Schaden nimmt. Die Nullposition bezieht sich auf den ersten Nullimpuls des Winkelgebers in positiver Richtung vom Anschlag.

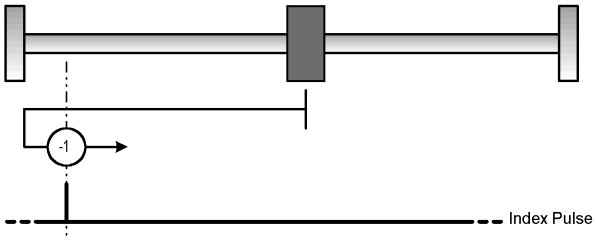


Abbildung 8.7: Referenzfahrt auf den negativen Anschlag mit Auswertung des Nullimpulses

8.2.3.6 Methode –2: positiver Anschlag mit Nullimpulsauswertung

Bei dieser Methode bewegt sich der Antrieb in positiver Richtung, bis er den Anschlag erreicht. Hierbei steigt das I^2t-Integral des Motors auf maximal 90%. Der Anschlag muss mechanisch so dimensioniert sein, dass er bei dem parametrierten Maximalstrom keinen Schaden nimmt. Die Nullposition bezieht sich auf den ersten Nullimpuls des Winkelgebers in negativer Richtung vom Anschlag.

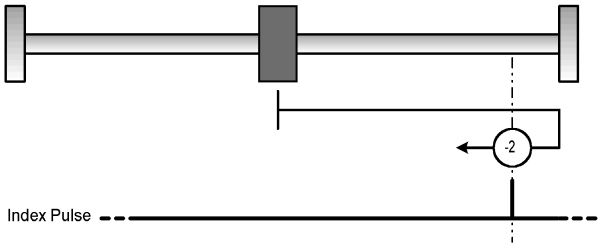


Abbildung 8.8: Referenzfahrt auf den positiven Anschlag mit Auswertung des Nullimpulses

8.2.3.7 Methoden 33 und 34: Referenzfahrt auf den Nullimpuls

Bei den Methoden 33 und 34 ist die Richtung der Referenzfahrt negativ bzw. positiv. Die Nullposition bezieht sich auf den ersten Nullimpuls vom Winkelgeber in Suchrichtung.

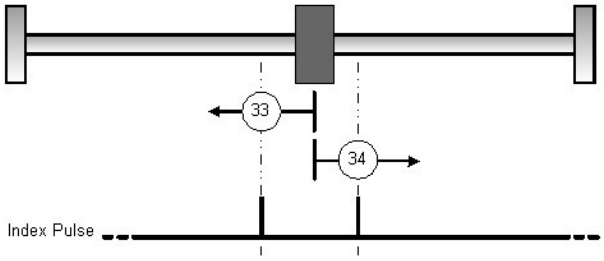


Abbildung 8.9: Referenzfahrt nur auf den Nullimpuls bezogen

8.2.3.8 Methode 35: Referenzfahrt auf die aktuelle Position

Bei der Methode 35 wird die Nullposition auf die aktuelle Position bezogen.

8.2.4 Steuerung der Referenzfahrt

Die Referenzfahrt wird durch das **controlword** / **statusword** gesteuert und überwacht. Das Starten erfolgt durch Setzen des Bit 4 im **controlword**. Der erfolgreiche Abschluss der Fahrt wird durch ein gesetztes Bit 12 im Objekt **statusword** angezeigt. Ein gesetztes Bit 13 im Objekt **statusword** zeigt an, dass während der Referenzfahrt ein Fehler aufgetreten ist. Die Fehlerursache kann über die Objekte **error_register** und **pre_defined_error_field** bestimmt werden.

Bit 4	Bedeutung
0	Referenzfahrt ist nicht aktiv
0 → 1	Referenzfahrt starten
1	Referenzfahrt ist aktiv
1 → 0	Referenzfahrt unterbrechen

Tabelle 8.1: Beschreibung der Bits im controlword

Bit 13	Bit 12	Bedeutung
0	0	Referenzfahrt ist noch nicht fertig
0	1	Referenzfahrt erfolgreich durchgeführt
1	0	Referenzfahrt nicht erfolgreich durchgeführt
1	1	verbotener Zustand

Tabelle 8.2: Beschreibung der Bits im statusword

8.3 Betriebsart Positionieren (Profile Position Mode)

8.3.1 Übersicht

Die Struktur dieser Betriebsart wird in Abbildung 8.10 ersichtlich:

Die Zielposition (**target_position**) wird dem Fahrkurven-Generator übergeben. Dieser erzeugt einen Lage-Sollwert (**position_demand_value**) für den Lageregler, der in dem Kapitel **Lageregler** beschrieben wird (Position Control Function, Kapitel 6.6). Diese zwei Funktionsblöcke können unabhängig voneinander eingestellt werden.

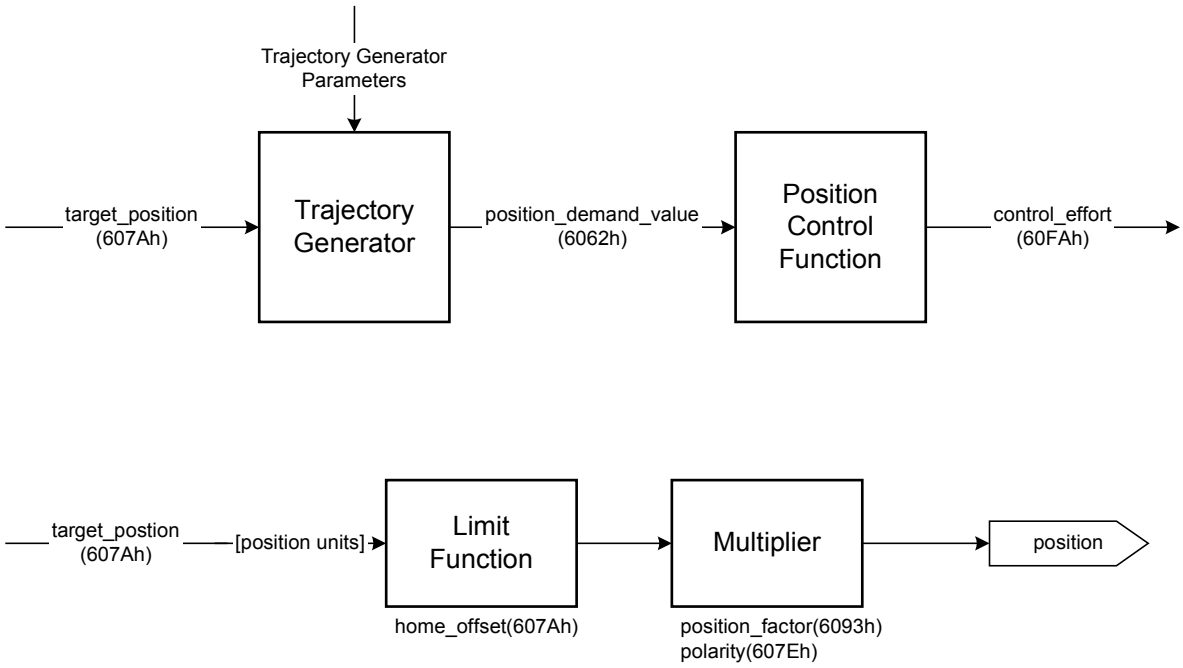


Abbildung 8.10: Fahrkurven-Generator und Lageregler

Alle Eingangsgrößen des Fahrkurven-Generators werden mit den Größen der Factor-Group (s. Kap. 6.2) in die internen Einheiten des Reglers umgerechnet. Die internen Größen werden hier mit einem Sternchen gekennzeichnet und werden vom Anwender in der Regel nicht benötigt.

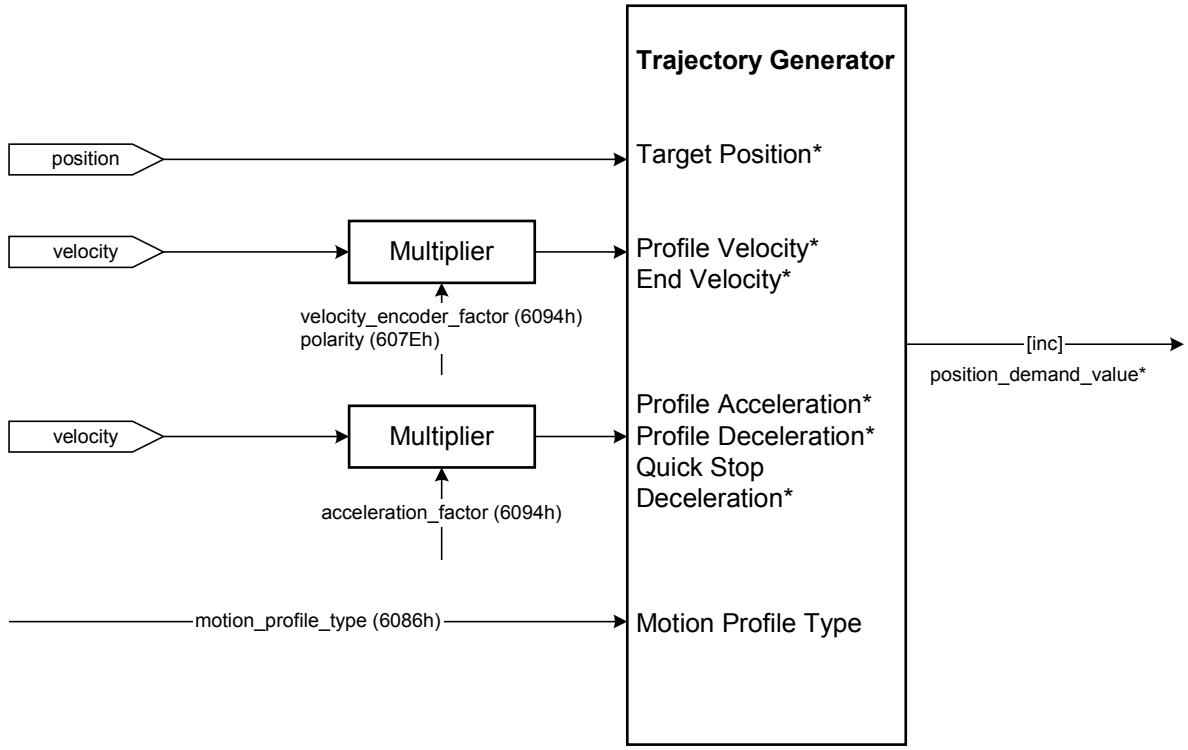


Abbildung 8.11: Der Fahrkurven-Generator

8.3.2 Beschreibung der Objekte

8.3.2.1 In diesem Kapitel behandelte Objekte

Index	Objekt	Name	Typ	Attr.
$607A_h$	VAR	target_position	INT32	rw
6081_h	VAR	profile_velocity	UINT32	rw
6082_h	VAR	end_velocity	UINT32	rw
6083_h	VAR	profile_acceleration	UINT32	rw
6084_h	VAR	profile_deceleration	UINT32	rw
6085_h	VAR	quick_stop_deceleration	UINT32	rw
6086_h	VAR	motion_profile_type	INT16	rw

8.3.2.2 Betroffene Objekte aus anderen Kapiteln

Index	Objekt	Name	Typ	Kapitel
6040h	VAR	controlword	INT16	7 Gerätesteuerung
6041h	VAR	statusword	UINT16	7 Gerätesteuerung
607Eh	VAR	polarity	UINT8	6.2 Umrechnungsfaktoren
6093h	ARRAY	position_factor	UINT32	6.2 Umrechnungsfaktoren
6094h	ARRAY	velocity_encoder_factor	UINT32	6.2 Umrechnungsfaktoren
6097h	ARRAY	acceleration_factor	UINT32	6.2 Umrechnungsfaktoren

8.3.2.3 Objekt $607A_h$: target_position

Das Objekt **target_position** (Zielposition) bestimmt, an welche Position der Antriebsregler fahren soll. Dabei muss die aktuelle Einstellung der Geschwindigkeit, der Beschleunigung, der Bremsverzögerung und die Art des Fahrprofils (**motion_profile_type**) etc. berücksichtigt werden. Die Zielposition (**target_position**) wird entweder als absolute oder relative Angabe interpretiert (**controlword**, Bit 6).

Index	**$607A_h$**
Name	**target_position**
Object Code	VAR
Data Type	INT32

Access	rw
PDO Mapping	yes
Units	position units
Value Range	--
Default Value	0

8.3.2.4 Objekt 6081_h: profile_velocity

Das Objekt **profile_velocity** gibt die Geschwindigkeit an, die normalerweise während einer Positionierung am Ende der Beschleunigungsrampe erreicht wird. Das Objekt **profile_velocity** wird in speed units angegeben.

Index	**6081_h**
Name	**profile_velocity**
Object Code	VAR
Data Type	UINT32

Access	rw
PDO Mapping	yes
Units	speed units
Value Range	--
Default Value	0

8.3.2.5 Objekt 6082_h: end_velocity

Das Objekt **end_velocity** (Endgeschwindigkeit) definiert die Geschwindigkeit, die der Antrieb haben muss, wenn er die Zielposition (**target_position**) erreicht. Dieses Objekt ist auf Null zu setzen, damit der Regler beim Erreichen der Zielposition (**target_position**) stoppt. Lückenlose Positionierungen mit einer von Null abweichenden Geschwindigkeit werden **NICHT** unterstützt. Das Objekt **end_velocity** wird in denselben Einheiten wie das Objekt **profile_velocity** angegeben.

Index	**6082_h**
Name	**end_velocity**
Object Code	VAR
Data Type	UINT32

Access	rw
PDO Mapping	yes
Units	speed units
Value Range	0
Default Value	0

8.3.2.6 Objekt 6083_h: profile_acceleration

Das Objekt **profile_acceleration** gibt die Beschleunigung an, mit der auf den Sollwert beschleunigt wird. Es wird in benutzerdefinierten Beschleunigungseinheiten (acceleration units) angegeben. (siehe Kapitel 6.2 Fehler: Referenz nicht gefunden).

Index	**6083_h**
Name	**profile_acceleration**
Object Code	VAR
Data Type	UINT32

Access	rw
PDO Mapping	yes
Units	acceleration units
Value Range	--
Default Value	10000 min^{-1} /s

8.3.2.7 Objekt 6084_h: profile_deceleration

Das Objekt **profile_deceleration** gibt die Beschleunigung an, mit der gebremst wird. Es wird in benutzerdefinierten Beschleunigungseinheiten (acceleration units) angegeben. (siehe Kapitel 6.2 Fehler: Referenz nicht gefunden).

Index	**6084_h**
Name	**profile_deceleration**
Object Code	VAR
Data Type	UINT32

Access	rw
PDO Mapping	yes
Units	speed units
Value Range	--
Default Value	10000 min^{-1} /s

8.3.2.8 Objekt 6085_h: quick_stop_deceleration

Das Objekt **quick_stop_deceleration** gibt an, mit welcher Bremsverzögerung der Motor stoppt, wenn ein **Quick Stop** ausgeführt wird (siehe Kapitel 7.1.2.2). Das Objekt **quick_stop_deceleration** wird in derselben Einheit wie das Objekt **profile_deceleration** angegeben.

Index	**6085_h**
Name	**quick_stop_deceleration**
Object Code	VAR
Data Type	UINT32

Access	rw
PDO Mapping	yes
Units	acceleration units
Value Range	--
Default Value	250000 min^{-1} /s

8.3.2.9 Objekt 6086_h: motion_profile_type

Das Objekt **motion_profile_type** wird verwendet, um die Art des Positionierprofils auszuwählen. Es steht die lineare Rampenform und ruckfreie Beschleunigung zur Verfügung.

Index	**6086_h**
Name	**motion_profile_type**
Object Code	VAR
Data Type	INT16

Access	rw
PDO Mapping	yes
Units	--
Value Range	0, 1
Default Value	0

Wert	Kurvenform
0	Lineare Rampe
1	ruckfreie Beschleunigung

8.3.3 Funktionsbeschreibung

Es gibt zwei Möglichkeiten Positionierungen auszuführen:

1.) Einfacher Fahrauftrag

Wenn der Regler eine Zielposition erreicht hat, signalisiert er dies dem Host mit dem Bit **target_reached** (Bit 10 im Objekt **statusword**). Der Regler stoppt , wenn er das Ziel erreicht hat.
Der folgende Fahrauftrag kann bereits übertragen und gestartet werden, während der erste Fahrauftrag noch ausgeführt wird. Dadurch wird unmittelbar nach der ersten Positionierung die folgende gestartet.

2.) Unterbrechung eines laufenden Fahrauftrags mit einem neuen

Die laufende Positionierung wird sofort unterbrochen und mit dem neuen Fahrauftrag begonnen.

Diese zwei Methoden werden durch die Bits **new_set_point** und **change_set_immediatly** in dem Objekt **controlword** und **set_point_acknowledge** in dem Objekt **statusword** kontrolliert. Diese Bits stehen in einem Frage-Antwort-Verhältnis zueinander. Hierdurch wird es möglich, einen Fahrauftrag vorzubereiten, während ein anderer noch läuft.

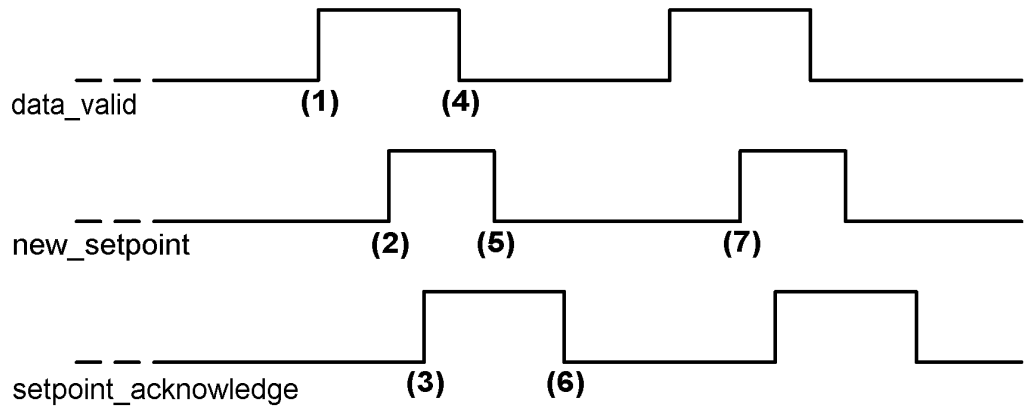


Abbildung 8.12: Fahrauftrag-Übertragung von einem Host

In Abbildung 8.12 können Sie sehen, wie der Host und der Regler über den CAN-Bus miteinander kommunizieren:

Zuerst werden die Positionierdaten (Zielposition, Fahrgeschwindigkeit und die Beschleunigung) an den Regler übertragen. Wenn der Positionierdatensatz vollständig eingeschrieben ist (1), kann der Host die Positionierung starten, indem er das Bit **new_set_point** im **controlword** auf „1“ setzt (2). Nachdem der Regler die neuen Daten erkannt und in seinen Puffer übernommen hat, meldet er dies dem Host durch das Setzen des Bits **set_point_acknowledge** im **statusword** (3).
Daraufhin kann der Host bereits wieder beginnen, einen neuen Positionierdatensatz in den Regler einzuschreiben (4) und das Bit **new_set_point** wieder zu löschen (5). Erst wenn der Regler einen neuen Fahrauftrag akzeptieren kann (6), signalisiert er dies durch eine „0“ im

set_point_acknowledge-Bit,. Vorher darf vom Host keine neue Positionierung gestartet werden (7).
In Abbildung 8.13 wird eine neue Positionierung erst gestartet, nachdem die vorherige vollständig abgeschlossen wurde. Der Host wertet hierzu das Bit **target_reached** im Objekt **statusword** aus.

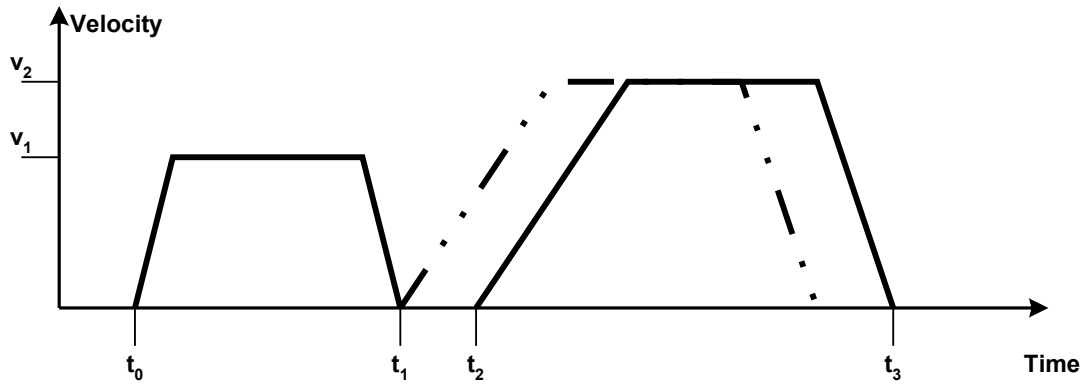


Abbildung 8.13: Einfacher Fahrauftrag

Wird neue Positionierung bereits gestartet während der erste Fahrauftrag noch abgearbeitet wird so ist gemäß Abbildung 8.13 $t_1 = t_2$ (strichpunktierter Verlauf).
Wenn im **controlword** neben dem Bit **new_set_point** auch das Bit **change_set_immediately** auf „1“ gesetzt wird, weist der Host den Regler damit an, *sofort* den neuen Fahrauftrag zu beginnen. Ein bereits in Bearbeitung befindlicher Fahrauftrag wird in diesem Fall abgebrochen wie in Abbildung 8.14 zu sehen ist. Der Host übergibt hierzu dem Regler das nachfolgende Ziel nach dem der Regler mit dem Löschen des Bits **set_point_acknowledge** signalisiert, dass er den Puffer gelesen und die zugehörige Positionierung gestartet hat.

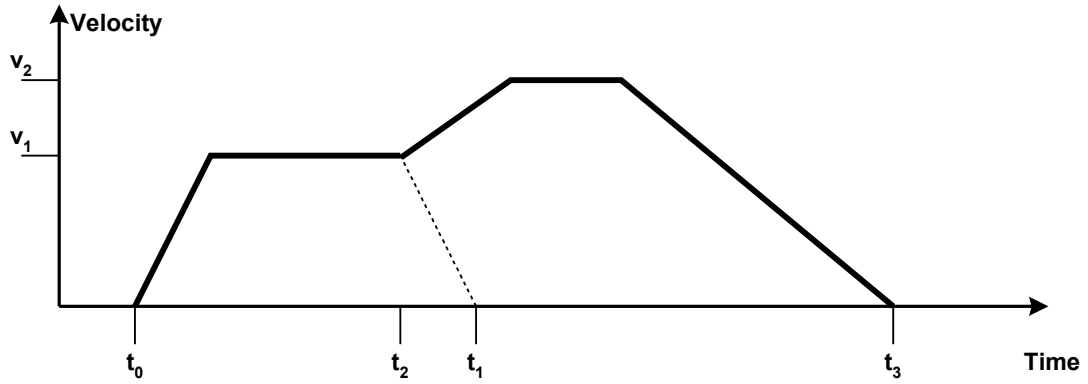


Abbildung 8.14: Unterbrechung einer laufenden Positionierung

Wird eine laufende Positionierung mit einem neuen Fahrauftrag mit relativer Positionierung unterbrochen, so kann man nicht genau vorhersagen, wo sich die Zielposition befindet, da man den genauen Zeitpunkt der Unterbrechung nicht kennt.

8.4 Interpolated Position Mode

8.4.1 Übersicht

Der Interpolated Position Mode (**IP**) ermöglicht die Vorgabe von Lagesollwerten in einer mehrachsigen Anwendung des Reglers. Dazu werden in einem festen Zeitraster (Synchronisations-Intervall) Synchronisations-Telegramme (SYNC) und Lagesollwerte von einer übergeordneten Steuerung vorgegeben. Da in der Regel das Intervall größer als ein Lagereglerzyklus ist, interpoliert der Regler selbständig die Datenwerte zwischen zwei vorgegebenen Positionswerten, wie in der folgenden Grafik skizziert.

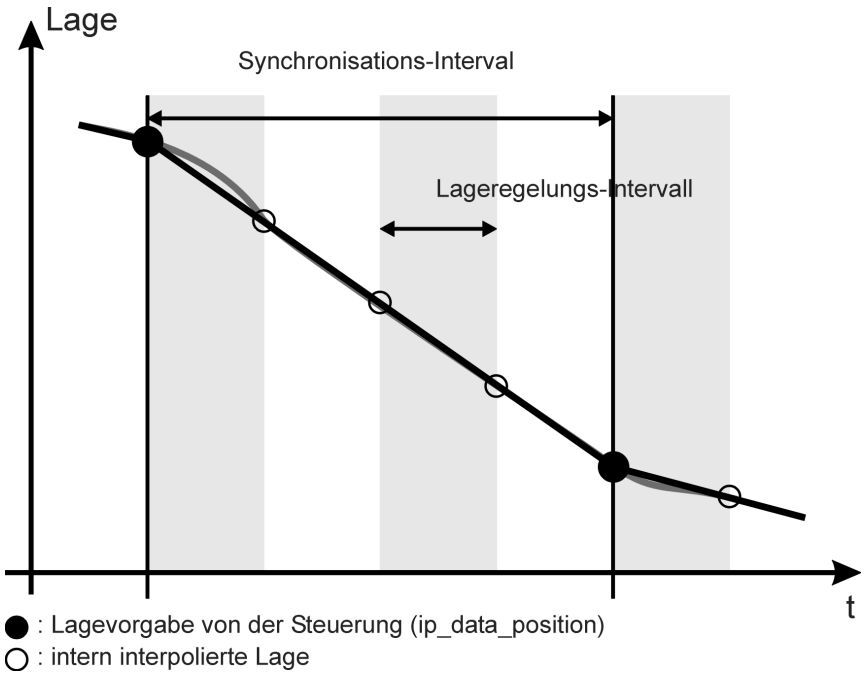


Abbildung 8.15: Fahrauftrag Lineare Interpolation zwischen zwei Datenwerten

Im Folgenden sind zunächst die für den **interpolated position mode** benötigten Objekte beschrieben. In einer anschließenden Funktionsbeschreibung wird umfassend auf die Aktivierung und die Reihenfolge der Parametrierung eingegangen.

8.4.2 Beschreibung der Objekte

8.4.2.1 In diesem Kapitel behandelte Objekte

Index	Objekt	Name	Typ	Attr.
$60C0_h$	VAR	interpolation_submode_select	INT16	rw
$60C1_h$	REC	interpolation_data_record		rw
$60C2_h$	REC	interpolation_time_period		rw
$60C4_h$	REC	interpolation_data_configuration		rw

8.4.2.2 Betroffene Objekte aus anderen Kapiteln

Index	Objekt	Name	Typ	Kapitel
6040h	VAR	controlword	INT16	7 Gerätesteuerung
6041h	VAR	statusword	UINT16	7 Gerätesteuerung
6093h	ARRAY	position_factor	UINT32	6.2 Umrechnungsfaktoren
6094h	ARRAY	velocity_encoder_factor	UINT32	6.2 Umrechnungsfaktoren
6097h	ARRAY	acceleration_factor	UINT32	6.2 Umrechnungsfaktoren

8.4.2.3 Objekt $60C0_h$: interpolation_submode_select

Über das Objekt **interpolation_submode_select** wird der Typ der Interpolation festgelegt. Es ist nur die herstellerspezifische Variante „Lineare Interpolation ohne Puffer“ verfügbar.

Index	**$60C0_h$**
Name	**interpolation_submode_select**
Object Code	VAR
Data Type	INT16

Access	rw
PDO Mapping	yes
Units	--
Value Range	-2
Default Value	-2

Wert	Interpolationstyp
-2	Lineare Interpolation ohne Puffer

8.4.2.4 Objekt $60C1_h$: interpolation_data_record

Der Objekt-Record **interpolation_data_record** repräsentiert den eigentlichen Datensatz. Er besteht aus einem Eintrag für den Lagewert (**ip_data_position**) und einem Steuerwort (**ip_data_controlword**), welches angibt, ob der Lagewert absolut oder relativ zu interpretieren ist. Die Angabe des Steuerworts ist optional. Wird er nicht angegeben, wird der Lagewert als absolut interpretiert. Soll das Steuerwort mit angegeben werden, muss aus Gründen der Datenkonsistenz zuerst Subindex 2 (**ip_data_controlword**) und anschließend Subindex 1 (**ip_data_position**) geschrieben werden, da intern die Datenübernahme mit Schreibzugriff auf **ip_data_position** ausgelöst wird.

Index	**$60C1_h$**
Name	**interpolation_data_record**
Object Code	RECORD
No. of Elements	2

Sub-Index	**01_h**
Description	**ip_data_position**
Data Type	INT32
Access	rw
PDO Mapping	yes
Units	position units
Value Range	--
Default Value	--

Objekt 60C2$_h$: interpolation_time_period

Über den Objekt-Record **interpolation_time_period** kann das Synchronisations-Intervall eingestellt werden. Über **ip_time_index** wird die Einheit (ms oder 1/10 ms) des Intervalls festgelegt, welches über **ip_time_units** parametriert wird. Das Synchronisations-Intervall muss exakt eingehalten werden.

Index	**60C2$_h$**
Name	**interpolation_time_period**
Object Code	RECORD
No. of Elements	2

Sub-Index	**01$_h$**
Description	**ip_time_units**
Data Type	UINT8
Access	rw
PDO Mapping	yes
Units	gemäß ip_time_index
Value Range	ip_time_index = -3: 8...40 ip_time_index = -4: 80...400
Default Value	--

Sub-Index	**02$_h$**
Description	**ip_time_index**
Data Type	INT8
Access	rw
PDO Mapping	yes
Units	--
Value Range	-3, -4
Default Value	-4

Wert	ip_time_units wird angegeben in
-3	10^{-3} Sekunden (ms)
-4	10^{-4} Sekunden (0.1 ms)

8.4.2.5 Objekt 60C4$_h$: interpolation_data_configuration

Über den Objekt-Record **interpolation_data_configuration** kann die Art (**buffer_organisation**) und Größe (**max_buffer_size**, **actual_buffer_size**) eines eventuell vorhandenen Puffers sowie der Zugriff auf diesen (**buffer_position**, **buffer_clear**) konfiguriert werden. Über das Objekt **size_of_data_record** kann die Größe eines Puffer-Elements ausgelesen werden. Obwohl bei der Interpolationsart „Lineare Interpolation ohne Puffer“ kein Puffer zur Verfügung steht, muss der Zugriff über das Objekt **buffer_clear** allerdings auch in diesem Fall freigegeben werden.

Index	**60C4$_h$**
Name	**interpolation_data_configuration**
Object Code	RECORD
No. of Elements	6

Sub-Index	**01$_h$**
Description	**max_buffer_size**
Data Type	UINT32
Access	ro
PDO Mapping	no
Units	--
Value Range	0
Default Value	0

Sub-Index	**02$_h$**
Description	**actual_size**
Data Type	UINT32
Access	rw
PDO Mapping	yes
Units	--
Value Range	0...max_buffer_size
Default Value	0

Sub-Index	**03$_h$**
Description	**buffer_organisation**
Data Type	UINT8
Access	rw
PDO Mapping	yes
Units	--
Value Range	0
Default Value	0

Wert	Bedeutung
0	FIFO

Sub-Index	**04_h**
Description	**buffer_position**
Data Type	UINT16
Access	rw
PDO Mapping	yes
Units	--
Value Range	0
Default Value	0

Sub-Index	**05_h**
Description	**size_of_data_record**
Data Type	UINT8
Access	wo
PDO Mapping	yes
Units	--
Value Range	2
Default Value	2

Sub-Index	**06_h**
Description	**buffer_clear**
Data Type	UINT8
Access	wo
PDO Mapping	yes
Units	--
Value Range	0, 1
Default Value	0

Wert	Bedeutung
0	Puffer löschen / Zugriff auf $60C1_h$ nicht erlaubt
1	Zugriff auf $60C1_h$ freigegeben

8.4.3 Funktionsbeschreibung

8.4.3.1 Vorbereitende Parametrierung

Bevor der Regler in die Betriebsart **interpolated position mode** geschaltet werden kann, müssen diverse Einstellungen vorgenommen werden: Dazu zählen die Einstellung des Interpolations-Intervalls (**interpolation_time_period**), also der Zeit zwischen zwei SYNC-Telegrammen und der Interpolationstyp (**interpolation_submode_select**). Zusätzlich muss der Zugriff auf den Positionspuffer über das Objekt **buffer_clear** freigegeben werden.

BEISPIEL

Aufgabe		**CAN-Objekt / COB**
Interpolationsart	-2	60C0h, interpolation_submode_select = -2
Zeiteinheit	0.1 ms	60C2h_02h, interpolation_time_index = -04
Zeitintervall	8 ms	60C2h_01h, interpolation_time_units = 80
Puffer-Freigabe	1	60C4h_06h, buffer_clear = 1
SYNC erzeugen		SYNC (Raster 8 ms)

8.4.3.2 Aktivierung des Interpolated Position Mode und Aufsynchronisation

Der **IP** wird über das Objekt **modes_of_operation (6060_h)** aktiviert. Über das Objekt **modes_of_operation_display (6061_h)** kann ausgelesen werden ob sich der Regler in der Betriebsart **interpolated position mode** befindet. Ab diesem Zeitpunkt sind die interne Selektoren so eingestellt, dass intern eine Lageregelung betrieben wird. Die Sollwerte die über CAN empfangen werden, werden zunächst interpoliert und gegebenfalls extrapoliert um auf das eingestellte Zeitintervall zu kommen.

Ist die Betriebsart eingenommen, kann die Übertragung von Positionsdaten an den Antrieb beginnen. Sinnvollerweise liest dazu die übergeordnete Steuerung zunächst die aktuelle Istposition aus dem Regler aus und schreibt diese zyklisch als neuen Sollwert (**interpolation_data_record**) in den Regler. Über Handshake- Bits des **controlword** und des **statusword** wird die Übernahme der Daten durch den Regler aktiviert. Durch Setzen des Bits **enable_ip_mode** im **controlword** zeigt der Host an, dass mit der Auswertung der Lagedaten begonnen werden soll. Erst wenn der Regler über das Statusbit **ip_mode_selected** im **statusword** dieses quittiert, werden die Datensätze ausgewertet.

Im Einzelnen ergibt sich daher folgende Zuordnung und der folgende Ablauf:

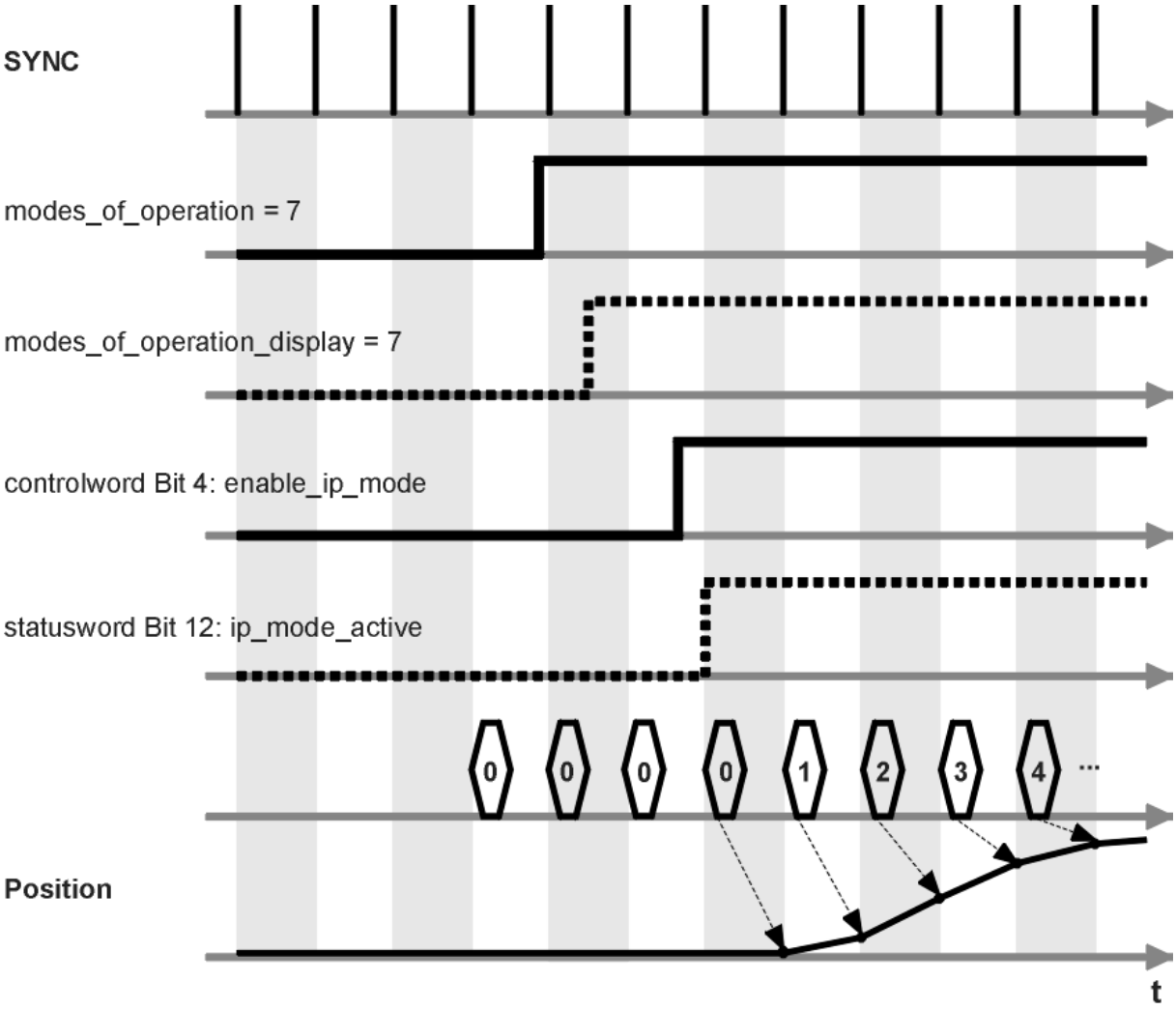


Abbildung 8.16: IP-Einschalten und Datenfreigabe

Nr.	Ereignis	CAN-Objekt
1	SYNC- Nachrichten erzeugen	
2	Anforderung der Betriebsart ip:	6060_h, modes_of_operation = 07
3	Warten bis Betriebsart eingenommen	6061_h, modes_of_operation_display = 07
4	Auslesen der akt. Istposition	6064_h, position_actual_value
5	Zurückschreiben als aktuelle Sollposition	$60C1_h\_01_h$, ip_data_position
6	Start der Interpolation	6040_h, controlword, enable_ip_mode
7	Quittierung durch Regler	6041_h, statusword, ip_mode_active
8	Ändern der aktuellen Sollposition gemäß Trajektorie	$60C1_h\_01_h$, ip_data_position

Nach Beendigung des synchronen Fahrvorgangs kann durch Löschen des Bits **enable_ip_mode** die weitere Auswertung von Lagewerten verhindert werden. Anschließend kann gegebenenfalls in eine andere Betriebsart umgeschaltet werden.

8.4.3.3 Unterbrechung der Interpolation im Fehlerfall

Wird eine laufende Interpolation (**ip_mode_active** gesetzt) durch das Auftreten eines Reglerfehlers unterbrochen, verhält sich der Antrieb zunächst so, wie für den jeweiligen Fehler spezifiziert (z.B. Wegnahme der Reglerfreigabe und Wechsel in den Zustand **SWICTH_ON_DISABLED**).

Die Interpolation kann dann nur durch eine erneute Einschaltung der IP-Paramater fortgesetzt werden, da der Regler wieder in den Zustand **OPERATION_ENABLE** gebracht werden muss, wodurch das Bit **ip_mode_active** gelöscht wird.

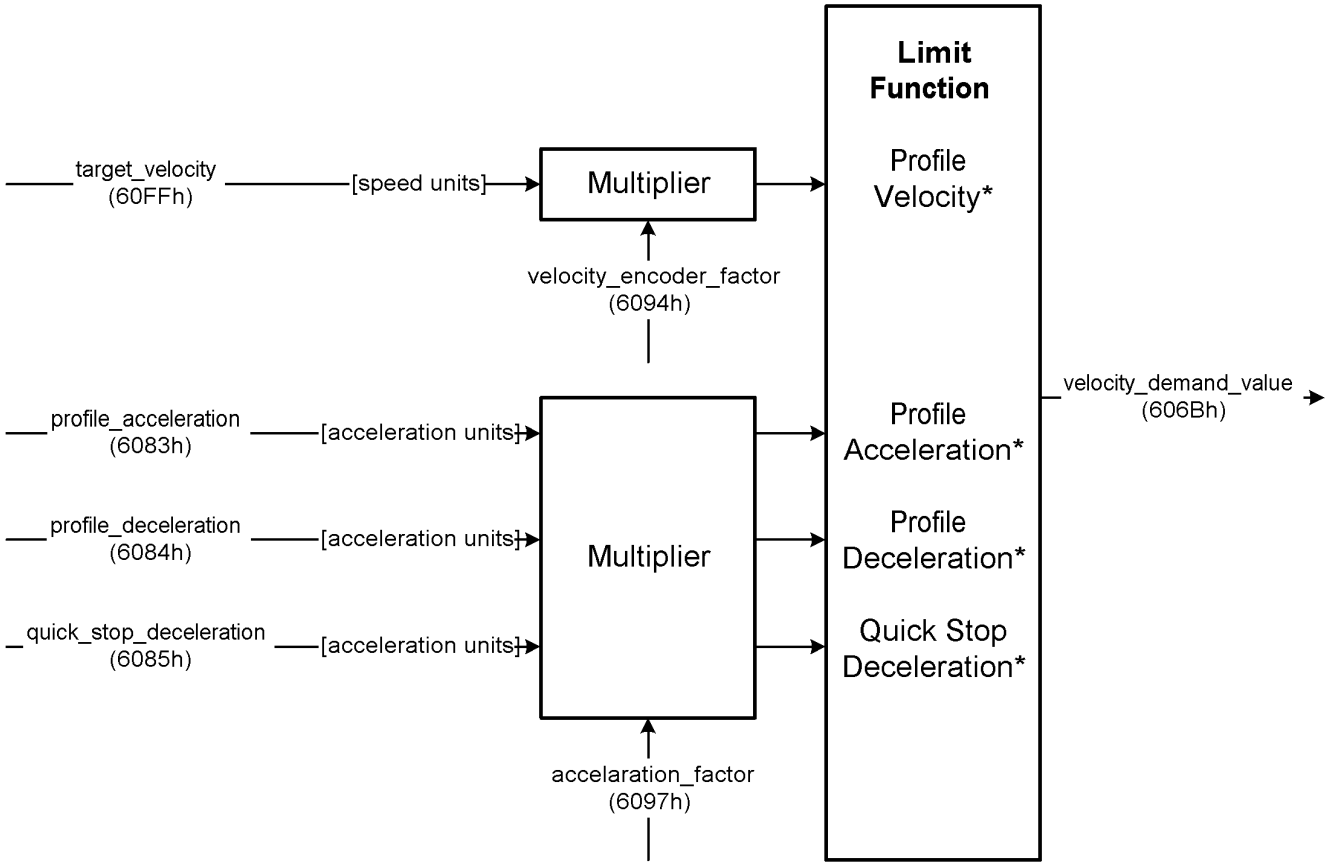
8.5 Betriebsart Drehzahlregelung (Profile Velocity Mode)

8.5.1 Übersicht

Der drehzahlgeregelte Betrieb (Profile Velocity Mode) beinhaltet die folgenden Unterfunktionen:

- Sollwert-Erzeugung durch den Rampen-Generator
- Drehzahlerfassung über den Winkelgeber durch Differentiation
- Drehzahlregelung mit geeigneten Eingabe- und Ausgabesignalen
- Begrenzung des Drehmomenten-Sollwertes (**torque_demand_value**)
- Überwachung der Ist-Geschwindigkeit (**velocity_actual_value**) mit der Fenster-Funktion/Schwelle

Die Bedeutung der folgenden Parameter ist im Kapitel Positionieren (Fehler: Referenz nicht gefunden) beschrieben: **profile_acceleration**, **profile_deceleration**, **quick_stop**.



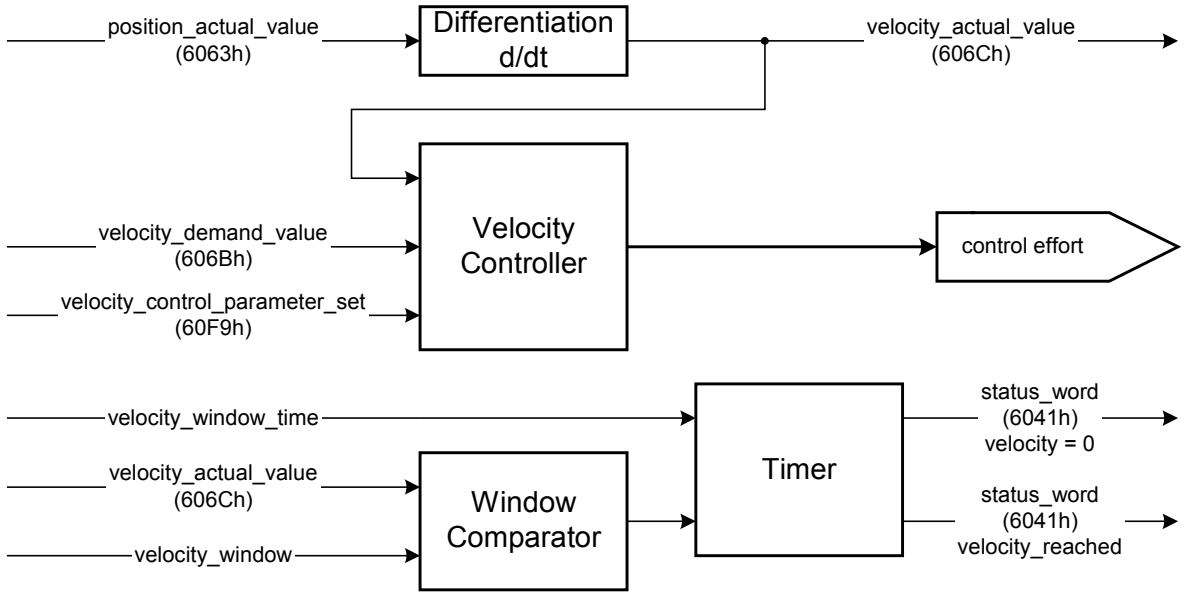


Abbildung 8.17: Struktur des drehzahlgeregelten Betriebs (Fehler: Referenz nicht gefunden)

8.5.2 Beschreibung der Objekte

8.5.2.1 In diesem Kapitel behandelte Objekte

Index	Objekt	Name	Typ	Attr.
6069_h	VAR	velocity_sensor_actual_value	INT32	ro
$606B_h$	VAR	velocity_demand_value	INT32	ro
$606C_h$	VAR	velocity_actual_value	INT32	ro
6080_h	VAR	max_motor_speed	UINT32	rw
$60FF_h$	VAR	target_velocity	INT32	rw

8.5.2.2 Betroffene Objekte aus anderen Kapiteln

Index	Objekt	Name	Typ	Kapitel
6040_h	VAR	controlword	INT16	7. Gerätesteuerung
6041_h	VAR	statusword	UINT16	7. Gerätesteuerung
6063_h	VAR	position_actual_value*	INT32	6.6 Lageregler
6069_h	VAR	velocity_sensor_actual_value	INT32	6.6 Lageregler
6071_h	VAR	target_torque	INT16	8.5.3.1 Momentenregler
6072_h	VAR	max_torque_value	UINT16	8.5.3.1 Momentenregler
$607E_h$	VAR	polarity	UINT8	6.2 Umrechnungsfaktoren
6083_h	VAR	profile_acceleration	UINT32	8.3 Positionieren
6084_h	VAR	profile_deceleration	UINT32	8.3 Positionieren
6085_h	VAR	quick_stop_deceleration	UINT32	8.3 Positionieren
6086_h	VAR	motion_profile_type	INT16	8.3 Positionieren
6094_h	ARRAY	velocity_encoder_factor	UINT32	6.2 Umrechnungsfaktoren

8.5.2.3 Objekt 6069_h: velocity_sensor_actual_value

Mit dem Objekt **velocity_sensor_actual_value** kann der Wert eines möglichen Geschwindigkeitsgebers in internen Einheiten ausgelesen werden. Bei dem DIS-2 kann kein separater Drehzahlgeber angeschlossen werden. Zur Bestimmung des Drehzahl-Istwertes sollte daher grundsätzlich das Objekt **$606C_h$** verwendet werden.

Index	**6069_h**
Name	**velocity_sensor_actual_value**
Object Code	VAR
Data Type	INT32

Access	ro
PDO Mapping	yes
Units	Inkremente/sec
Value Range	--
Default Value	--

8.5.2.4 Objekt $606B_h$: velocity_demand_value

Mit diesem Objekt kann der aktuelle Drehzahlsollwert des Drehzahlreglers ausgelesen werden. Auf diesen wirkt der Sollwert vom Rampen-Generator bzw. des Fahrkurven-Generators. Bei aktiviertem Lageregler wird außerdem dessen Korrektur-geschwindigkeit addiert.

Index	**$606B_h$**
Name	**velocity_demand_value**
Object Code	VAR
Data Type	INT32

Access	ro
PDO Mapping	yes
Units	speed units
Value Range	--
Default Value	--

8.5.2.5 Objekt $606C_h$: velocity_actual_value

Über das Objekt **velocity_actual_value** kann der Drehzahl-Istwert ausgelesen werden.

Index	**$606C_h$**
Name	**velocity_actual_value**
Object Code	VAR
Data Type	INT32

Access	ro
PDO Mapping	yes
Units	speed units
Value Range	--
Default Value	--

8.5.2.6 Objekt 6080_h: max_motor_speed

Das Objekt **max_motor_speed** gibt die höchste erlaubte Drehzahl für den Motor in min^{-1}. Das Objekt wird benutzt, um den Motor zu schützen und kann dem Motordatenblatt entnommen werden. Der Drehzahl-Sollwert wird auf diesen Wert begrenzt.

Index	**6080_h**
Name	**max_motor_speed**
Object Code	VAR
Data Type	UINT16

Access	rw
PDO Mapping	yes
Units	min^{-1}
Value Range	0... 32768 min^{-1}
Default Value	3000 min^{-1}

8.5.2.7 Objekt $60FF_h$: target_velocity

Das Objekt **target_velocity** ist die Sollwertvorgabe für den Rampen-Generator.

Index	**$60FF_h$**
Name	**target_velocity**
Object Code	VAR
Data Type	INT32

Access	rw
PDO Mapping	yes
Units	speed units
Value Range	--
Default Value	--

8.5.3 Objekte: Drehzahl-Rampen

Alle Sollwerte des Drehzahlsollwert-Selektors (der per Voreinstellung auf Analogeingang 0 steht) werden über einen Rampengenerator geführt, der sprungförmige Änderungen des Geschwindigkeits-Sollwertes in einen linear ansteigenden (trapezförmigen) Verlauf umwandelt. Die Steigung der Rampen (Beschleunigung) kann mit den nachfolgenden Objekten für die positive und negative Drehrichtung und die ansteigende und abfallende Flanke eines Sollwertsprungs parametriert werden. Wenn die Betriebsart profile_velocity_mode ausgewählt wird werden alle 4 Rampen eingeschaltet. Eine Möglichkeit über CAN die Rampen auszuschalten besteht nicht.

Die Beziehung zwischen den velocity_ramps und den profile_deceleration und profile acceleration ist in folgender Grafik dargestellt.

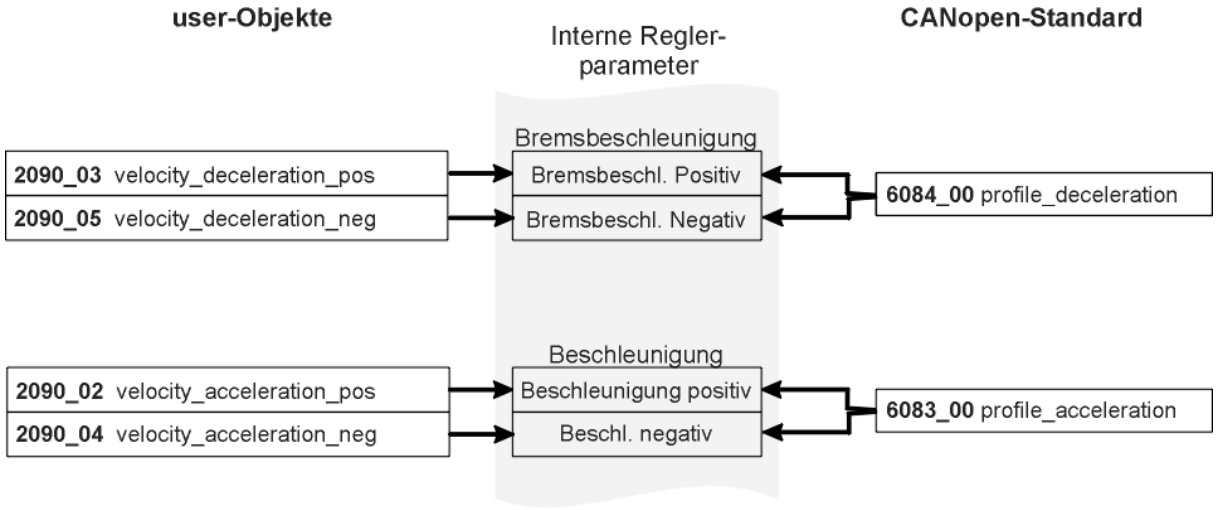


Abbildung 8.18: Zuordnung der Rampen

8.5.3.1 Objekt 2090_h: velocity_ramps

Die Bedeutung der einzelnen Objekte kann der nachfolgenden Skizze entnommen werden:

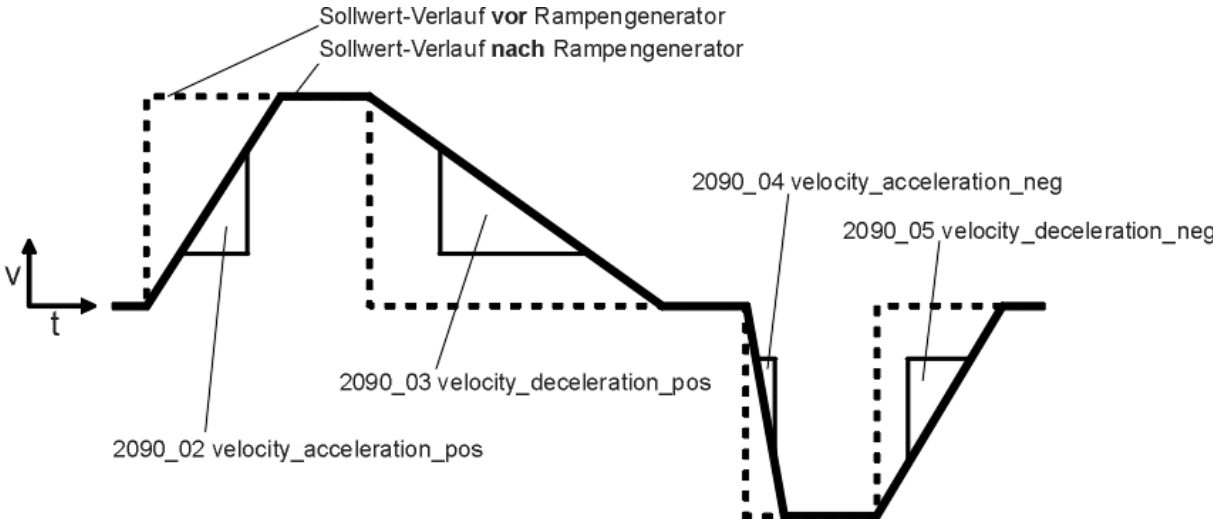


Abbildung 8.19: Bedeutung der Velocity_ramps

Index	**2090h**
Name	**velocity_ramps**
Object Code	RECORD
No. of Elements	5

Sub-Index	**02h**
Description	**velocity_acceleration_pos**
Data Type	UINT32
Access	rw
PDO Mapping	no
Units	--
Value Range	$0...2^{31}-1$
Default Value	$01E84800_h$

Sub-Index	**03h**
Description	**velocity_deceleration_pos**
Data Type	UINT32
Access	rw
PDO Mapping	no
Units	--
Value Range	$0...2^{31}-1$
Default Value	$01E84800_h$

Sub-Index	**04h**
Description	**velocity_acceleration_neg**
Data Type	UINT32
Access	rw
PDO Mapping	no
Units	--
Value Range	$0...2^{31}-1$
Default Value	$01E84800_h$

Sub-Index	**05h**
Description	**velocity_deceleration_neg**
Data Type	UINT32
Access	rw
PDO Mapping	no
Units	--
Value Range	$0...2^{31}-1$
Default Value	$01E84800_h$

Betriebsart Momentenregelung (Profile Torque Mode)

8.5.4 Übersicht

Dieses Kapitel beschreibt den drehmomentengeregelten Betrieb. Diese Betriebsart erlaubt es, dass dem Regler ein externer Momenten-Sollwert **target_torque** vorgegeben wird. Somit ist es möglich, dass dieser Regler auch für Bahnsteuerungen eingesetzt werden kann, bei denen sowohl der Lageregler als auch der Drehzahlregler auf einen externen Rechner verlagert sind.

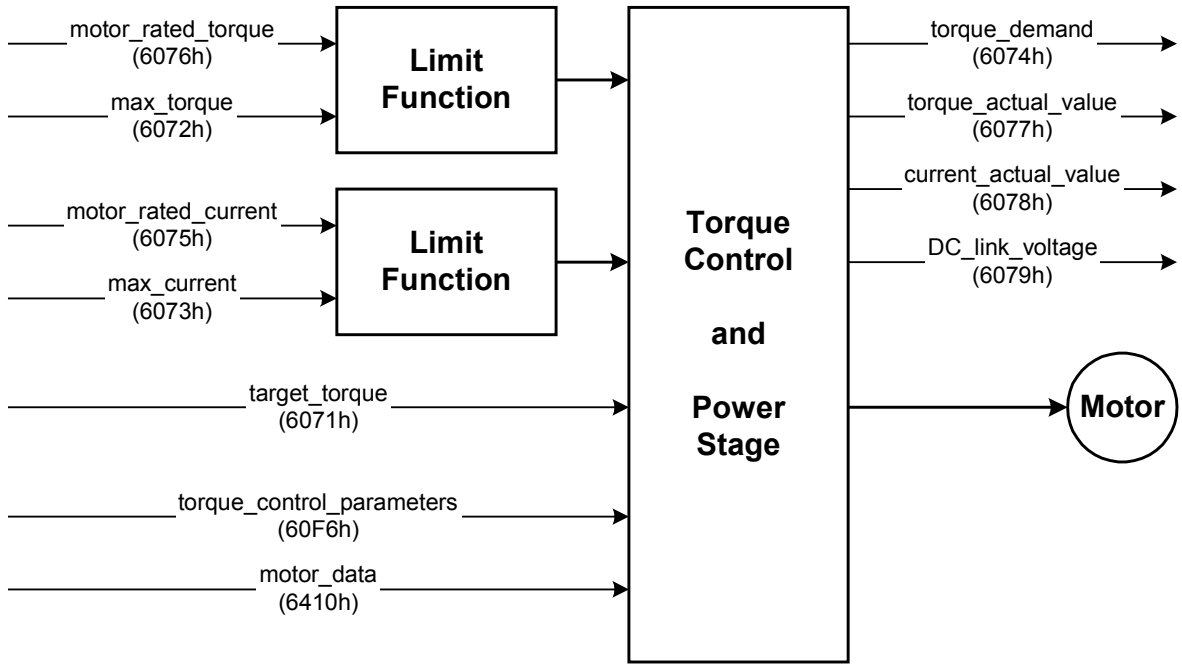


Abbildung 8.20: Struktur des Fehler: Referenz nicht gefundens

Es wird kein Rampengenerator unterstützt. Wenn im **controlword** das Bit 8 **halt** gesetzt wird, wird Stromsollwert auf Null gesetzt. Entsprechend wird der Sollwert **target_torque** auf **max_torque** begrenzt, wenn das Bit 8 wieder gelöscht wird.
Alle Definitionen innerhalb dieses Dokumentes beziehen sich auf drehbare Motoren. Wenn lineare Motoren benutzt werden, müssen sich alle „Drehmoment“-Objekte statt dessen auf eine „Kraft“ beziehen. Der Einfachheit halber sind die Objekte nicht doppelt vertreten und ihre Namen sollten nicht verändert werden.
Die Betriebsarten Positionierbetrieb (Profile Position Mode) und Drehzahlregler (Profile Velocity Mode) benötigen für ihre Funktion den Momentenregler. Deshalb ist es immer notwendig, diesen zu parametrieren.

8.5.5 Beschreibung der Objekte

8.5.5.1 In diesem Kapitel behandelte Objekte

Index	Objekt	Name	Typ	Attr.
6071_h	VAR	target_torque	INT16	rw
6072_h	VAR	max_torque	UINT16	rw
6074_h	VAR	torque_demand_value	INT16	ro
6076_h	VAR	motor_rated_torque	UINT32	rw
6077_h	VAR	torque_actual_value	INT16	ro
6078_h	VAR	current_actual_value	INT16	ro
6079_h	VAR	DC_link_circuit_voltage	UINT32	ro
$60F6_h$	RECORD	torque_control_parameters		rw

8.5.5.2 Betroffene Objekte aus anderen Kapiteln

Index	Objekt	Name	Typ	Kapitel
6040_h	VAR	controlword	INT16	6.10 Gerätesteuerung
6410_h	RECORD	motor_data	UINT32	6.4 Stromregler u. Motoranpassung
6075_h	VAR	motor_rated_current	UINT32	6.4 Stromregler u. Motoranpassung
6073_h	VAR	max_current	UINT16	6.4 Stromregler u. Motoranpassung

8.5.5.3 Objekt 6071_h: target_torque

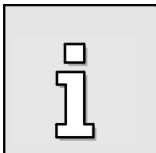
Dieser Parameter ist im drehmomentengeregelten Betrieb (Fehler: Referenz nicht gefunden) der Eingabewert für den Drehmomentenregler. Er wird in Tausendsteln des Nennmomentes (Objekt **6076_h**) angegeben.

Index	**6071_h**
Name	**target_torque**
Object Code	VAR
Data Type	INT16

Access	rw
PDO Mapping	yes
Units	motor_rated_torque / 1000
Value Range	-32768...32768
Default Value	0

8.5.5.4 Objekt 6072_h: max_torque

Dieser Wert stellt das höchstzulässige Drehmoment des Motors dar. Es wird in Tausendsteln des Nennmomentes (Objekt **6076_h**) angegeben. Wenn zum Beispiel kurzzeitig eine zweifache Überlastung des Motors zulässig ist, so ist hier der Wert 2000 einzutragen.



Das Objekt **6072_h**: max_torque korrespondiert mit dem Objekt **6073_h**: **max_current** und darf erst beschrieben werden, wenn zuvor das Objekt **6075_h**: **motor_rated_current** mit einem gültigen Wert beschrieben wurde.

Index	**6072_h**
Name	**max_torque**
Object Code	VAR
Data Type	UINT16

Access	rw
PDO Mapping	yes
Units	motor_rated_torque / 1000
Value Range	1000...65536
Default Value	1968

8.5.5.5 Objekt 6074_h: torque_demand_value

Über dieses Objekt kann das aktuelle Sollmoment in Tausendsteln des Nennmoments (**6076_h**) ausgelesen werden. Berücksichtigt sind hierbei die internen Begrenzungen des Reglers (Stromgrenzwerte und I^2T-Überwachung).

Index	**6074_h**
Name	**torque_demand_value**
Object Code	VAR
Data Type	INT16

Access	ro
PDO Mapping	yes
Units	motor_rated_torque / 1000
Value Range	--
Default Value	--

8.5.5.6 Objekt 6076_h: motor_rated_torque

Dieses Objekt gibt das Nennmoment des Motors an. Dieses kann dem Typenschild des Motors entnommen werden. Es ist in der Einheit 0.001 Nm einzugeben.

Index	**6076_h**
Name	**motor_rated_torque**
Object Code	VAR
Data Type	UINT32

Access	rw
PDO Mapping	yes
Units	0.001 Nm
Value Range	--
Default Value	2870

8.5.5.7 Objekt 6077_h: torque_actual_value

Über dieses Objekt kann der Drehmomenten-Istwert des Motors in Tausendsteln des Nennmomentes (Objekt **6076_h**) ausgelesen werden.

Index	**6077_h**
Name	**torque_actual_value**
Object Code	VAR
Data Type	INT16

Access	ro
PDO Mapping	yes
Units	motor_rated_torque / 1000
Value Range	--
Default Value	--

8.5.5.8 Objekt 6078_h: current_actual_value

Über dieses Objekt kann der Strom-Istwert des Motors in Tausendsteln des Nennstromes (Objekt **6075_h**) ausgelesen werden.

Index	**6078_h**
Name	**current_actual_value**
Object Code	VAR
Data Type	INT16

Access	ro
PDO Mapping	yes
Units	motor_rated_current / 1000
Value Range	--
Default Value	--

8.5.5.9 Objekt 6079_h: dc_link_circuit_voltage

Über dieses Objekt kann die Zwischenkreisspannung des Reglers ausgelesen werden. Die Spannung wird in der Einheit Millivolt angegeben.

Index	**6079_h**
Name	**dc_link_circuit_voltage**
Object Code	VAR
Data Type	UINT32

Access	ro
PDO Mapping	yes
Units	mV
Value Range	--
Default Value	--

Stichwortverzeichnis

A

B

C

D

E

F

G

I

K