

Wichtig!

Vor Gebrauch sorgfältig lesen!

Aufbewahren für späteres Nachschlagen!

metronix
servo drives



EtherCAT  

CANopen 

EtherCAT- und CANopen-Handbuch

Original-EtherCAT und CANopen-Handbuch

› Urheberrechte

© 2020 Metronix Meßgeräte und Elektronik GmbH. Alle Rechte vorbehalten.

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.

Metronix ServoCommander® ist ein eingetragenes Warenzeichen der Metronix Meßgeräte und Elektronik GmbH.

› **Kontakt**daten

Metronix Meßgeräte und Elektronik GmbH

Kocherstraße 3

38120 Braunschweig

Germany

Telefon: +49 (0)531 8668 0

Telefax: +49 (0)531 8668 555

E-mail: vertrieb@metronix.de

<https://www.metronix.de>

› **Revisions**information

Handbuchname	EtherCAT und CANopen-Handbuch
Dateiname	EC_CO-HB_1p0_DE.pdf
Version	1.0
Jahr	2020

Inhaltsverzeichnis

1 Zu diesem Handbuch	14
1.1 Aufbau der Warnhinweise	14
1.2 Schreibweisen in diesem Handbuch	15
2 Schnellstart	16
2.1 CANopen	16
2.1.1 Grundlagen	16
2.1.2 Verkabelung und Steckerbelegung	17
2.1.3 Verkabelungs-Hinweise	18
2.1.4 Status LEDs	19
2.1.5 CANopen aktivieren	20
2.1.6 Integration des Servoreglers in ein Masterprojekt	22
2.2 EtherCAT	28
2.2.1 Grundlagen	28
2.2.2 Verkabelung und Steckerbelegung	29
2.2.3 Verkabelungs-Hinweise	30
2.2.4 Status LEDs	30
2.2.5 EtherCAT aktivieren	31
2.2.6 Integration des Servoreglers in ein Masterprojekt	32
2.2.7 EoE (Ethernet over EtherCAT®)	36
2.2.7.1 EoE im Master aktivieren	37
2.2.7.2 Bridge konfigurieren	39
3 Parametrierung	40
3.1 Parametersätze laden und speichern	42
3.1.1 Übersicht	42
3.1.2 Beschreibung der Objekte	44
3.1.2.1 Objekt 1011h: restore_default_parameters	44
3.1.2.2 Objekt 1010h: store_parameters	44
3.2 Kompatibilitäts- Einstellungen	45

3.2.1 Übersicht	45
3.2.2 Beschreibung der Objekte	45
3.2.2.1 Objekt 6510h_F0h: compatibility_control	45
3.3 Umrechnungsfaktoren (Factor Group)	47
3.3.1 Übersicht	47
3.3.2 Parametrierung der Factor Group	48
3.3.3 Beschreibung der Objekte	49
3.3.3.1 Objekt 6093h: position_factor	49
3.3.3.2 Objekt 6094h: velocity_encoder_factor	49
3.3.3.3 Objekt 6097h: acceleration_factor	50
3.3.3.4 Objekt 607Eh: polarity	50
3.4 Endstufenparameter	51
3.4.1 Übersicht	51
3.4.2 Beschreibung der Objekte	51
3.4.2.1 Objekt 6510h_10h: enable_logic	51
3.4.2.2 Objekt 6510h_30h: pwm_frequency	53
3.4.2.3 Objekt 6510h_3Ah: enable_enhanced_modulation	53
3.4.2.4 Objekt 6510h_31h: power_stage_temperature	54
3.4.2.5 Objekt 6510h_32h: max_power_stage_temperature	54
3.4.2.6 Objekt 6510h_33h: nominal_dc_link_circuit_voltage	55
3.4.2.7 Objekt 6510h_34h: actual_dc_link_circuit_voltage	55
3.4.2.8 Objekt 6510h_35h: max_dc_link_circuit_voltage	56
3.4.2.9 Objekt 6510h_36h: min_dc_link_circuit_voltage	56
3.4.2.10 Objekt 6510h_37h: enable_dc_link_undervoltage_error	57
3.4.2.11 Objekt 6510h_40h: nominal_current	58
3.4.2.12 Objekt 6510h_41h: peak_current	59
3.5 Stromregler und Motoranpassung	60
3.5.1 Übersicht	60
3.5.2 Beschreibung der Objekte	61
3.5.2.1 Objekt 6075h: motorRatedCurrent	61
3.5.2.2 Objekt 6073h: maxCurrent	61

3.5.2.3	Objekt 604Dh: pole_number	62
3.5.2.4	Objekt 6410h_11h: encoder_offset_angle	62
3.5.2.5	Objekt 6410h_10h: phase_order	63
3.5.2.6	Objekt 6410h_03h: iit_time_motor	63
3.5.2.7	Objekt 6410h_04h: iit_ratio_motor	64
3.5.2.8	Objekt 6510h_3Dh: iit_ratio_servo	64
3.5.2.9	Objekt 6510h_38h: iit_error_enable	65
3.5.2.10	Objekt 6510h_2Eh: motor_temperature	65
3.5.2.11	Objekt 6410h_14h: motor_temperature_sensor_polarity	66
3.5.2.12	Objekt 6510h_2Fh: max_motor_temperature	66
3.5.2.13	Objekt 60F6h: torque_control_parameters	67
3.5.2.14	Objekt 203Ah: torque_feed_forward	67
3.6	Drehzahlregler	68
3.6.1	Übersicht	68
3.6.2	Beschreibung der Objekte	68
3.6.2.1	Objekt 60F9h: velocity_control_parameters	68
3.6.2.2	Objekt 2073h: velocity_display_filter_time	69
3.7	Lageregler (Position Control Function)	70
3.7.1	Übersicht	70
3.7.2	Beschreibung der Objekte	71
3.7.2.1	Objekt 60FBh: position_control_parameter_set	71
3.7.2.2	Objekt 6062h: position_demand_value	73
3.7.2.3	Objekt 202Dh: position_demand_sync_value	73
3.7.2.4	Objekt 6064h: position_actual_value	73
3.7.2.5	Objekt 6066h: following_error_time_out	73
3.7.2.6	Objekt 6065h: following_error_window	74
3.7.2.7	Objekt 60F4h: following_error_actual_value	74
3.7.2.8	Objekt 60FAh: control_effort	74
3.7.2.9	Objekt 6410h_0Fh: rotor_position	75
3.7.2.10	Objekt 6067h: position_window	75
3.7.2.11	Objekt 6068h: position_window_time	75

3.7.2.12	Objekt 6510h_22h: position_error_switch_off_limit	76
3.7.2.13	Objekt 2030h: set_position_absolute	76
3.7.2.14	Objekt 607Dh: software_position_limit	77
3.7.2.15	Objekt 607Bh: position_range_limit	77
3.7.2.16	Objekt 6510h_20h: position_range_limit_enable	78
3.8	Sollwert- Begrenzung	79
3.8.1	Objekt 2415h: current_limitation	79
3.8.2	Objekt 2416h: speed_limitation	80
3.9	Geberanpassungen	81
3.9.1	Übersicht	81
3.9.2	Beschreibung der Objekte	81
3.9.2.1	Objekt 2024h: encoder_x2a_data_field	81
3.9.2.2	Objekt 2026h: encoder_x2b_data_field	82
3.9.2.3	Objekt 2025h: encoder_x10_data_field	83
3.9.2.4	Objekt 202Ch: max_comm_enc_pos_enc_difference	84
3.10	Leitfrequenzausgang	85
3.10.1	Übersicht	85
3.10.2	Beschreibung der Objekte	85
3.10.2.1	Objekt 201Ah: encoder_emulation_data	85
3.10.2.2	Objekt 2028h: encoder_emulation_resolution	85
3.11	Soll-/ Istwertaufschaltung	86
3.11.1	Übersicht	86
3.11.2	Beschreibung der Objekte	86
3.11.2.1	Objekt 201Fh: commutation_encoder_select	86
3.11.2.2	Objekt 2021h: position_encoder_selection	87
3.11.2.3	Objekt 2022h: synchronisation_encoder_selection	87
3.11.2.4	Objekt 202Fh: synchronisation_selector_data	88
3.11.2.5	Objekt 2023h: synchronisation_filter_time	88
3.12	Analoge Eingänge	89
3.12.1	Übersicht	89
3.12.2	Beschreibung der Objekte	89

3.12.2.1	Objekt 2400h: analog_input_voltage (Eingangsspannung)	89
3.12.2.2	Objekt 2401h: analog_input_offset (Offset Analogeingänge)	90
3.13	Digitale Ein- und Ausgänge	91
3.13.1	Übersicht	91
3.13.2	Beschreibung der Objekte	91
3.13.2.1	Objekt 60FDh: digital_inputs	91
3.13.2.2	Objekt 60FEh: digital_outputs	91
3.13.2.3	Objekt 2420h: digital_output_state_mapping	92
3.14	Endschalter / Referenzschalter	95
3.14.1	Übersicht	95
3.14.2	Beschreibung der Objekte	95
3.14.2.1	Objekt 6510h_11h: limit_switch_polarity	95
3.14.2.2	Objekt 6510h_12h: limit_switch_selector	96
3.14.2.3	Objekt 6510h_15h: limit_switch_deceleration	96
3.14.2.4	Objekt 6510h_14h: homing_switch_polarity	97
3.14.2.5	Objekt 6510h_13h: homing_switch_selector	97
3.15	Erfassen von Positionen	98
3.15.1	Übersicht	98
3.15.2	Beschreibung der Objekte	98
3.15.2.1	Objekt 204Ah: sample_data	98
3.16	Bremsen-Ansteuerung	101
3.16.1	Übersicht	101
3.16.2	Beschreibung der Objekte	101
3.16.2.1	Objekt 6510h_18h: brake_delay_time	101
3.17	Geräteinformationen	102
3.17.1	Beschreibung der Objekte	102
3.17.1.1	Objekt 1000h: device_type	102
3.17.1.2	Objekt 1008h: manufacturer_device_name	102
3.17.1.3	Objekt 1009h: manufacturer_hardware_version	102
3.17.1.4	Objekt 100Ah: manufacturer_software_version	103
3.17.1.5	Objekt 1018h: identity_object	103

3.17.1.6	Objekt 6510h_A0h: drive_serial_number	104
3.17.1.7	Objekt 6510h_A1h: drive_type	105
3.17.1.8	Objekt 6510h_A9h: firmware_main_version	105
3.17.1.9	Objekt 6510h_AAh: firmware_custom_version	106
3.17.1.10	Objekt 6510h_ADh: km_release	106
3.17.1.11	Objekt 6510h_ACh: firmware_type	107
3.17.1.12	Objekt 6510h_B0h: cycletime_current_controller	107
3.17.1.13	Objekt 6510h_B1h: cycletime_velocity_controller	108
3.17.1.14	Objekt 6510h_B2h: cycletime_position_controller	108
3.17.1.15	Objekt 6510h_B3h: cycletime_trajectory_generator	108
3.17.1.16	Objekt 6510h_C0h: commissioning_state	109
3.17.1.17	Objekt 20FDh: user_device_name	109
3.18	Fehlermanagement	110
3.18.1	Übersicht	110
3.18.2	Beschreibung der Objekte	110
3.18.2.1	Objekt 2100h: error_management	110
3.18.2.2	Objekt 200Fh: last_warning_code	111
4	Gerätesteuerung (Device Control)	112
4.1	Übersicht	112
4.2	Das Zustandsdiagramm des Servoreglers (State Machine)	113
4.2.1	Zustandsdiagramm: Zustände	115
4.2.2	Zustandsdiagramm: Zustandsübergänge	116
4.3	controlword (Steuerwort)	118
4.4	Auslesen des Servoreglerzustands	121
4.5	statuswords (Statusworte)	122
4.5.1	Objekt 6041h: statusword	122
4.5.2	Objekt 2000h: manufacturer_statuswords	126
4.5.3	Objekt 2005h: manufacturer_status_masks	129
4.5.4	Objekt 200Ah: manufacturer_status_invert	129
4.5.5	Objekt 2001h: manufacturer_warnings	129
4.5.6	Objekt 2006h: manufacturer_warning_masks	130

4.6	Beschreibung der weiteren Objekte	131
4.6.1	Objekt 605Bh: shutdown_option_code	131
4.6.2	Objekt 605Ch: disable_operation_option_code	131
4.6.3	Objekt 605Ah: quick_stop_option_code	132
4.6.4	Objekt 605Eh: fault_reaction_option_code	132
5	Betriebsarten	133
5.1	Einstellen der Betriebsart	133
5.1.1	Übersicht	133
5.1.2	Beschreibung der Objekte	133
5.1.2.1	Objekt 6060h: modes_of_operation	133
5.1.2.2	Objekt 6061h: modes_of_operation_display	134
5.2	Betriebsart Referenzfahrt (Homing Mode)	135
5.2.1	Übersicht	135
5.2.2	Beschreibung der Objekte	136
5.2.2.1	Wichtige Objekte in anderen Kapiteln	136
5.2.2.2	Objekt 607Ch: home_offset	136
5.2.2.3	Objekt 6098h: homing_method	136
5.2.2.4	Objekt 6099h: homing_speeds	137
5.2.2.5	Objekt 609Ah: homing_acceleration	138
5.2.2.6	Objekt 2045h: homing_timeout	138
5.2.3	Referenzfahrt-Abläufe	139
5.2.3.1	Methode -17 und -18: Anschlag	139
5.2.3.2	Methoden -1 und -2: Anschlag mit Nullimpulsauswertung	139
5.2.3.3	Methoden 17 und 18: Positiver und negativer Endschalter	140
5.2.3.4	Methoden 1 und 2: Positiver und negativer Endschalter mit Nullimpulsauswertung	140
5.2.3.5	Methoden 23 und 27: Referenzschalter	141
5.2.3.6	Methoden 7 und 11: Referenzschalter und Nullimpulsauswertung	142
5.2.3.7	Methoden -23 und -27: Referenzfahrt (pos/neg) auf den Referenzschalter	143
5.2.3.8	Methoden 32 und 33: Referenzfahrt auf den Nullimpuls	143
5.2.3.9	Methode 34: Referenzfahrt auf die aktuelle Position	143
5.2.4	Steuerung der Referenzfahrt	144

5.3 Betriebsart Positionieren (Profile Position Mode)	145
5.3.1 Übersicht	145
5.3.2 Funktionsbeschreibung	145
5.3.3 Beschreibung der Objekte	147
5.3.3.1 Wichtige Objekte in anderen Kapiteln	147
5.3.3.2 Objekt 607Ah: target_position	147
5.3.3.3 Objekt 6081h: profile_velocity	147
5.3.3.4 Objekt 6082h: end_velocity	148
5.3.3.5 Objekt 6083h: profile_acceleration	148
5.3.3.6 Objekt 6084h: profile_deceleration	148
5.3.3.7 Objekt 6085h: quick_stop_deceleration	149
5.3.3.8 Objekt 6086h: motion_profile_type	149
5.4 Interpolated Position Mode	150
5.4.1 Übersicht	150
5.4.2 Funktionsbeschreibung	150
5.4.3 Beschreibung der Objekte	153
5.4.3.1 Wichtige Objekte in anderen Kapiteln	153
5.4.3.2 Objekt 60C0h: interpolation_submode_select	153
5.4.3.3 Objekt 60C1h: interpolation_data_record	154
5.4.3.4 Objekt 60C2h: interpolation_time_period	154
5.4.3.5 Objekt 60C3h: interpolation_sync_definition	155
5.4.3.6 Objekt 60C4h: interpolation_data_configuration	156
5.4.3.7 Objekt 1006h: communication_cycle_period	157
5.5 Cyclic Synchronous Position Mode	158
5.5.1 Übersicht	158
5.5.2 Beschreibung der Objekte	158
5.5.2.1 Wichtige Objekte in anderen Kapiteln	158
5.6 Betriebsart Drehzahlregelung (Profile Velocity Mode)	159
5.6.1 Übersicht	159
5.6.2 Beschreibung der Objekte	160
5.6.2.1 Wichtige Objekte in anderen Kapiteln	160

5.6.2.2	Objekt 6069h: velocity_sensor_actual_value	161
5.6.2.3	Objekt 606Ah: sensor_selection_code	161
5.6.2.4	Objekt 606Bh: velocity_demand_value	161
5.6.2.5	Objekt 202Eh: velocity_demand_sync_value	161
5.6.2.6	Objekt 606Ch: velocity_actual_value	162
5.6.2.7	Objekt 2074h: velocity_actual_value_filtered	162
5.6.2.8	Objekt 606Dh: velocity_window	163
5.6.2.9	Objekt 606Eh: velocity_window_time	163
5.6.2.10	Objekt 606Fh: velocity_threshold	163
5.6.2.11	Objekt 6070h: velocity_threshold_time	164
5.6.2.12	Objekt 6080h: max_motor_speed	164
5.6.2.13	Objekt 60FFh: target_velocity	164
5.6.2.14	Drehzahl- Rampen	165
5.7	Betriebsart Momentenregelung (Profile Torque Mode)	167
5.7.1	Übersicht	167
5.7.2	Beschreibung der Objekte	168
5.7.2.1	Wichtige Objekte in anderen Kapiteln	168
5.7.2.2	Objekt 6071h: target_torque	168
5.7.2.3	Objekt 6072h: max_torque	168
5.7.2.4	Objekt 6074h: torque_demand_value	169
5.7.2.5	Objekt 6076h: motorRated_torque	169
5.7.2.6	Objekt 6077h: torque_actual_value	169
5.7.2.7	Objekt 6078h: current_actual_value	170
5.7.2.8	Objekt 6079h: dc_link_circuit_voltage	170
5.7.2.9	Objekt 6087h: torque_slope	170
5.7.2.10	Objekt 6088h: torque_profile_type	171
6	Detaillierte Beschreibung des CANopen-Protokolls	172
6.1	Einleitung	172
6.2	SDO-Zugriff	173
6.2.1	SDO-Sequenzen zum Lesen und Schreiben	174
6.2.2	SDO-Fehlermeldungen (abort codes)	175

6.2.3	Simulation von SDO-Zugriffen	176
6.3	PDO-Message	177
6.3.1	Beschreibung der Objekte	178
6.3.2	Objekte zur PDO-Parametrierung	181
6.3.3	PDOs aktivieren	186
6.4	EMERGENCY-Message	187
6.4.1	Übersicht	187
6.4.2	Aufbau der EMERGENCY-Message	188
6.4.3	Beschreibung der Objekte	188
6.5	SYNC-Message	189
6.6	Netzwerkmanagement (NMT-Service)	190
6.7	Bootup	193
6.7.1	Übersicht	193
6.7.2	Aufbau der Bootup- Nachricht	193
6.8	Heartbeat (Error Control Protocol)	193
6.8.1	Übersicht	193
6.8.2	Aufbau der Heartbeat- Nachricht	194
6.8.3	Beschreibung der Objekte	194
6.9	Nodeguarding (Error Control Protocol)	195
6.9.1	Übersicht	195
6.9.2	Aufbau der Nodeguarding-Nachrichten	195
6.9.3	Beschreibung der Objekte	196
6.9.3.1	Objekt 100Ch: guard_time	196
6.9.3.2	Objekt 100Dh: life_time_factor	196
6.10	Tabelle der Identifier	197
7	Anhang	198
7.1	CANopen	198
7.2	Kenndaten des CAN-Interface	198
7.3	Fehlercodes der EMERGENCY-Message	199

1 Zu diesem Handbuch

Dieses Handbuch beschreibt, wie die Servoregler der Gerätefamilie ARS 2000 FS und smartServo BL 4000-C in ein CANopen- bzw. Ethercat-Netzwerk einbezogen werden können. Es werden der physikalische Anschluss, die Aktivierung des Feldbus-Protokolls, die Einbindung in das Netzwerk und die Parameter zur Anpassung an die jeweilige Applikation beschrieben.

Es richtet sich an Personen, die bereits mit der jeweiligen Servoregler-Familie vertraut sind und das entsprechende Produkthandbuch gelesen und verstanden haben.

Im Produkthandbuch sind Hinweise für den sachgemäßen und fachgerechten Transport, die Lagerung, die Montage, die Installation, die Projektierung und den einwandfreien und sicheren Betrieb des Servoreglers enthalten.

Das Produkthandbuch enthält Sicherheitshinweise, die unbedingt beachtet werden müssen.

Die Produkthandbücher stehen auf unserer Homepage (<https://www.metronix.de>) zum Download zur Verfügung.

1.1 Aufbau der Warnhinweise

Warnhinweise sind folgendermaßen aufgebaut:

- Signalwort
- Art der Gefährdung
- Maßnahmen zur Abwehr der Gefährdung

> Verwendete Signalwörter

▲ GEFAHR Bezeichnet eine unmittelbar drohende Gefahr.

Wenn die Situation nicht gemieden wird, sind Tod oder schwerste Verletzungen die Folge.

▲ WARNUNG Bezeichnet eine möglicherweise gefährliche Situation.

Wenn die Situation nicht gemieden wird, können Tod oder schwerste Verletzungen die Folge sein.

▲ VORSICHT Bezeichnet eine möglicherweise gefährliche Situation.

Wenn die Situation nicht gemieden wird, können leichte oder geringfügige Verletzungen die Folge sein.

ACHTUNG Bezeichnet eine Warnung vor Sachschäden.

> Verwendete Warnzeichen gemäß ISO 7010

Warnzeichen	Erklärung
	Warnung vor lebensgefährlicher elektrischer Spannung.

1.2 Schreibweisen in diesem Handbuch

› Aufbau der Hinweise

Hinweise in diesem Handbuch sind folgendermaßen aufgebaut:

- Signalwort "HINWEIS"
- Einleitender Satz
- Erklärungen, spezielle Hinweise und Tipps

› Bedienelemente, Menüs

Bedienelemente, Menüs und Menüpfade werden in Orange geschrieben.

Beispiel: Doppelklick auf das gewünschte Gerät oder Anklicken der Schaltfläche **Verbindung neu aufbauen** stellt eine Online-Verbindung her.

› CAN-Objekte, Bitkonstanten

Begriffe aus den CANopen-Standards wie zum Beispiel Parameternamen (CAN-Objekte) werden in Blau geschrieben. Bitkonstanten werden durch eine andere Schriftart hervorgehoben.

Beispiel: Wenn dieses Bit gesetzt ist, wird Bit 4 des **statusword** (`voltage_enabled`) gemäß DSP 402 v2.0 ausgegeben.

› Zustände, Kommandos

Reglerzustände (siehe Abschnitt 4 *Gerätesteuerung (Device Control)* auf Seite 112) werden in einer anderen Schriftart gesetzt und großgeschrieben. Kommandos werden mit einem weißen Kasten hinterlegt.

Beispiel:

NOT_READY_TO_SWITCH_ON					Der Servoregler führt einen Selbsttest durch.	
4	Enable Operation	1	1	1	1	Regelung gemäß eingestellter Betriebsart

2 Schnellstart

Dieses Kapitel beschreibt, wie die Servoregler ARS 2000 FS oder BL 4000-C mit einer handelsüblichen CANopen- bzw. Ethercat-Steuerung verbunden und in Betrieb genommen werden, um eine schnelle Einrichtung zum Starten der Anwendungs-Entwicklung zu erhalten. Je nachdem, welche Feldbus-Schnittstelle verwendet wird, kann das jeweils andere Kapitel übersprungen werden.

Im Kapitel 3 *Parametrierung* auf Seite 40 werden dann alle verfügbaren Parameter beschrieben, die in der Regel sowohl unter CANopen als auch unter EtherCAT gleichermaßen nutzbar sind, um den Servoregler an die jeweilige Applikation anzupassen. Dieses Kapitel richtet sich an Anwender, die bereits eine Industriesteuerung besitzen.

2.1 CANopen

CANopen ist ein von der Vereinigung "CAN in Automation" gepflegter Standard, der die Verwendung von CAN in der Automatisierungstechnik herstellerunabhängig definiert. Die CANopen-Schnittstelle in den Servoreglern ARS 2000 FS und BL 4000-C ist gemäß CiA 301 (Übertragungsschicht) und CiA 402 (Antriebsreglerprofil) ausgeführt.

2.1.1 Grundlagen

Das Feldbusprotokoll CANopen legt fest, wie in der industriellen Automatisierung Daten über den CAN-Feldbus ausgetauscht werden.

Im Allgemeinen gibt es zwei Arten von Nachrichten (Kommunikationsobjekte), die zwischen dem Master (z.B. CoDeSys-Steuerung) und dem Slave ausgetauscht werden.

- **SDO (Servicedatenobjekte)**
Diese Art von Nachrichten wird für die azyklische Kommunikation zwischen Master und Slave verwendet, z.B. während der Initialisierungsphase der Anwendung oder in einer sehr einfachen Anwendung, bei der kein zyklischer Datenaustausch erforderlich ist.
- **PDO (Prozessdatenobjekte)**
Diese Art von Nachrichten wird zyklisch/automatisch zwischen Master und Slave ausgetauscht, um Prozessdaten zu übertragen. Prozessdaten sind alle Daten, die von Master oder Slave benötigt werden, um die Anwendung auszuführen. In unserem Beispiel enthalten diese Prozessdaten z.B. Positionssoll-/Istwerte, Steuer- und Statusworte und andere wichtige Informationen, um den Servoregler als SoftMotion-Achse nutzen zu können.

Es gibt weitere Nachrichtentypen, wie [Emergency Messages](#), [Heartbeat Messages](#) oder [Node Guarding Messages](#), die ebenfalls zwischen Master und Slave ausgetauscht werden, aber nur bei einem speziellen Ereignis oder in speziellen Anwendungen. Beispielsweise wird vom Slave eine [Emergency Message](#) an den Master gesendet, wenn ein schwerer Fehler im Servoantrieb aufgetreten ist. Eine detaillierte Beschreibung dieser Nachrichtentypen finden Sie in Kapitel 6 *Detaillierte Beschreibung des CANopen-Protokolls* auf Seite 172.

2.1.2 Verkabelung und Steckerbelegung

Das CAN-Interface ist in den Servoreglern ARS 2000 FS und BL 4000-C integriert und somit immer verfügbar. Der CAN-Bus-Anschluss ist normgemäß als 9-poliger DSUB-Stecker (reglerseitig) ausgeführt.

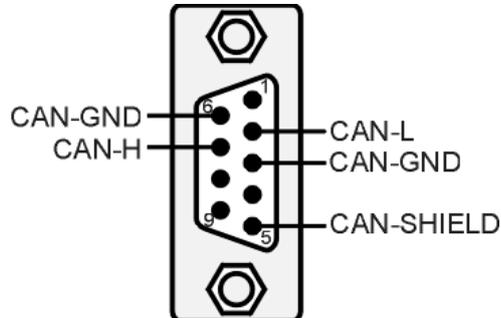


Abbildung 1: CAN-Steckverbinder

HINWEIS CAN-Bus-Verkabelung

Bei der Verkabelung der Servoregler 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 Servoregler aus Sicherheitsgründen mit einem Fehler abschaltet.

HINWEIS 120Ω Abschlusswiderstand

In den Servoreglern BL 4000-C ist kein Abschlusswiderstand integriert.

In den Servoreglern ARS 2000 FS ist ein Abschlusswiderstand über den DIP-Schalter CAN TERM zuschaltbar.

2.1.3 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.

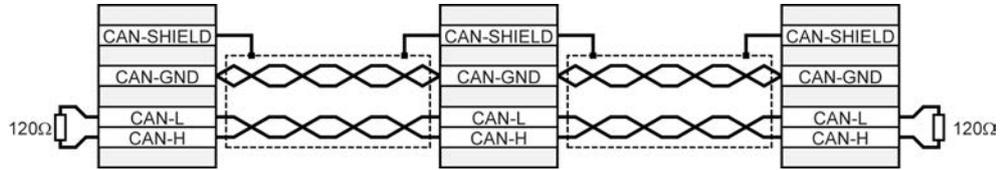


Abbildung 2: Verkabelungsbeispiel

Die einzelnen Knoten des Netzwerkes werden grundsätzlich linienförmig miteinander verbunden, so dass das CAN-Kabel von Servoregler zu Servoregler durchgeschleift wird.

An beiden Enden des CAN-Kabels muss jeweils genau ein Abschlusswiderstand von $120\Omega \pm 5\%$ vorhanden sein. Häufig ist in CAN-Karten oder in einer SPS bereits ein solcher Abschlusswiderstand eingebaut, der entsprechend berücksichtigt werden muss.

Für die Verkabelung muss **geschirmtes** Kabel mit genau zwei **verdrillten** Adernpaaren verwendet werden.

- Ein verdrilltes Aderpaar wird für den Anschluss von CAN-H und CAN-L verwendet.
- Die Adern des anderen Paares werden **gemeinsam** für CAN-GND verwendet.
- Der Schirm des Kabels wird bei allen Knoten an die CAN-Shield-Anschlüsse geführt.

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.

› Technische Daten CAN-Kabel:

- 2 Paare à 2 verdrillten Adern, $d \geq 0,22 \text{ mm}^2$, geschirmt
- Schleifenwiderstand $< 0,2 \Omega/\text{m}$
- Wellenwiderstand $100\text{-}120 \Omega$

2.1.4 Status LEDs

› BL 4000-C

Zur einfachen Anzeige des CAN-Bus-Status ist der Servoregler mit zwei Feldbusstatus-LEDs ausgestattet:

Die LEDs zeigen die folgenden Zustände an:

Name	Farbe	Beschreibung
RUN/SF/MS	Grün	Diese LED zeigt eine laufende Kommunikation zwischen dem Master und dem Servoregler an. Sie wird ausgelöst, wenn eine Nachricht vom Master empfangen wurde. Wenn diese LED ständig AUS ist, wird keine Kommunikation mit dem Servoregler durchgeführt.
ERR/BF/NS	Rot	Diese LED zeigt den Feldbusfehler bezogen auf den CAN-Feldbus an. Die LED blinkt, wenn ein CAN-bezogener Feldbusfehler auftritt und noch nicht quittiert wurde.

Im Normalbetrieb leuchtet die RUN-LED, da die Kommunikation mit dem Servoregler erfolgt und die ERR-LED ausgeschaltet ist.

Wenn die ERR-LED blinkt, ist einer der folgenden CAN-Feldbusfehler aufgetreten:

Gruppe 12: CAN-Kommunikation		
12-1	CAN: Kommunikationsfehler, Bus AUS	Verkabelung überprüfen: Kabelspezifikation eingehalten, Kabelbruch, maximale Kabellänge überschritten, Abschlusswiderstände korrekt, Kabelschirm geerdet, alle Signale aufgelegt?
12-2	CAN: Kommunikationsfehler beim Senden	
12-3	CAN: Kommunikationsfehler beim Empfangen	
12-4	CAN: Node Guarding	Die Steuerung ist ausgefallen oder die Zykluszeit der Remoteframes von Servoregler und Steuerung stimmen nicht überein.
12-5	CAN: RPDO zu kurz	Ein empfangenes RPDO enthält weniger Bytes als im Servoregler parametrierter.
12-9	CAN: Protokollfehler	Bitte Kontakt zum Technischen Support aufnehmen.

› ARS 2000 FS

Die LED CAN-ON blinkt kurz auf, wenn der Servoregler ein Telegramm empfangen hat.

2.1.5 CANopen aktivieren

Die CANopen-Feldbuskommunikation muss einmalig über das CANopen-Fenster des Metronix ServoCommander® aktiviert werden (**Parameter / Feldbus / CANopen / Betriebsparameter**). Je nach Gerätefamilie sind möglicherweise nicht alle Optionen verfügbar, so dass das Aussehen des Fensters unterschiedlich sein kann.

Es müssen insgesamt 3 verschiedene Parameter eingestellt werden:

Parameter	Beschreibung
Bitrate	Dieser Parameter bestimmt die auf dem CAN-Bus verwendete Bitrate in kBit/s. Sie muss mit der Bitrate in der Steuerung übereinstimmen. Beachten Sie, dass bei hohen Bitraten die maximal zulässige Kabellänge reduziert ist.
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 Servoreglers von der äußeren Beschaltung abhängig zu machen. Zur Basis-Knotennummer wird einmalig nach dem Reset die Eingangskombination der digitalen Eingänge DIN0...DIN3 addiert. Beim ARS 2000 FS können auch die analogen Eingänge mit einer Wertigkeit von 32 bzw. 64 hinzuaddiert werden, wenn der jeweilige Eingang auf $V_{ref} = 10V$ gebrückt ist.

Parameter	Beschreibung
Optionen	<p>Test auf doppelte Knotennummer (Nur ARS 2000 FS): Alle in einem CANopen-Netzwerk vorhandenen Geräte senden eine Einschaltmeldung (Bootup-Message) über den Bus, die die Knotennummer des Senders enthält. Empfängt der Servoregler eine solche Einschaltmeldung, die seiner eigenen Knotennummer entspricht, wird der Fehler 12-0 ausgelöst.</p> <p>Knotennummer zu COB-IDs der PDOs addieren: Durch Setzen dieser Option, müssen die COB-IDs der PDOs nicht manuell an die Knotennummer angepasst werden (siehe Kapitel 6.3.2 <i>Objekte zur PDO-Parametrierung</i> auf Seite 181).</p>

Abschließend kann das CANopen-Protokoll aktiviert werden. Die oben genannten Parameter können nur bei deaktiviertem Protokoll geändert werden.

HINWEIS DIP-Schalter beim ARS 2000 FS

Beim Servoregler ARS 2000 FS lassen sich Feldbus-Einstellungen auch über die DIP-Schalter der einsteckbaren Sicherheitsmodule einstellen. Die möglichen Optionen sind im Produkthandbuch ARS 2000 FS aufgeführt.

HINWEIS Parametrierung der CANopen-Funktionalität

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

HINWEIS Identische Knotennummern

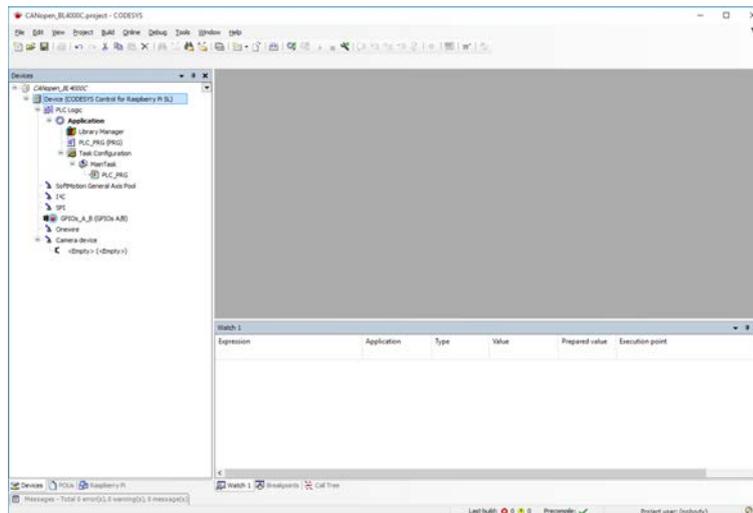
Es ist nicht erlaubt, mehrere Servoregler am CANopen-Feldbus mit derselben Knotennummer zu betreiben. Stellen Sie daher sicher, dass jeder Servoregler am CANopen-Feldbus eine eindeutige Knotennummer hat, bevor Sie die Kommunikation aktivieren.

2.1.6 Integration des Servoreglers in ein Masterprojekt

Als Beispiel wird in diesem Kapitel gezeigt, wie man die Servoregler BL 4000-C oder ARS 2000 FS in ein CoDeSys V3.5 Projekt integriert und als SoftMotion Antrieb betreibt.

Als Voraussetzung müssen Sie die CANopen EDS-Datei (Electronic Data Sheet) für den entsprechenden Servoregler von der Metronix Website (<https://www.metronix.de>) herunterladen. Diese Datei enthält eine vollständige Beschreibung der Antriebsmerkmale und des Objektverzeichnisses und wird vom CoDeSys (oder einem anderen CANopen-Master) zur automatischen Konfiguration des Servoreglers verwendet. Das folgende Beispiel zeigt die Installation eines **BL 4104-C**.

Starten Sie CoDeSys, verbinden Sie sich mit Ihrem CANopen-Master und erstellen Sie ein leeres Projekt.



› EDS-Datei in das CoDeSys-Geräteverzeichnis installieren

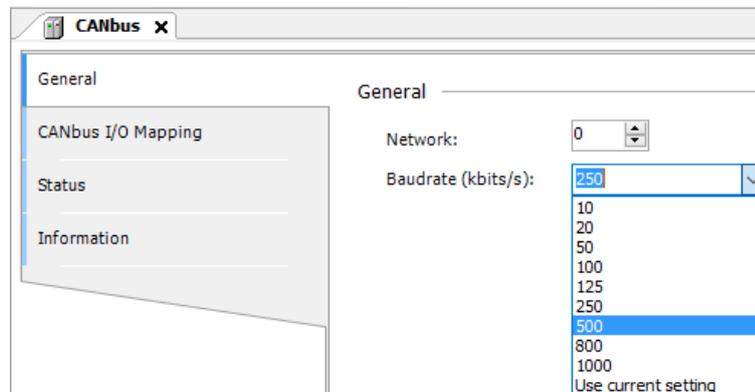
- Rufen Sie das CoDeSys-Geräteverzeichnis auf.
Pfad: **Tools / Device Repository**
- Klicken Sie die Schaltfläche **Installieren**
- Wählen Sie die heruntergeladene EDS-Datei von Ihrem Speicherort aus.
- Bestätigen Sie durch Klick auf die Schaltfläche **Öffnen**.

Nun ist der CoDeSys-Software der Servoregler BL 4000-C oder ARS 2000 FS bekannt und kann verwendet werden.

› CANopen-Master hinzufügen

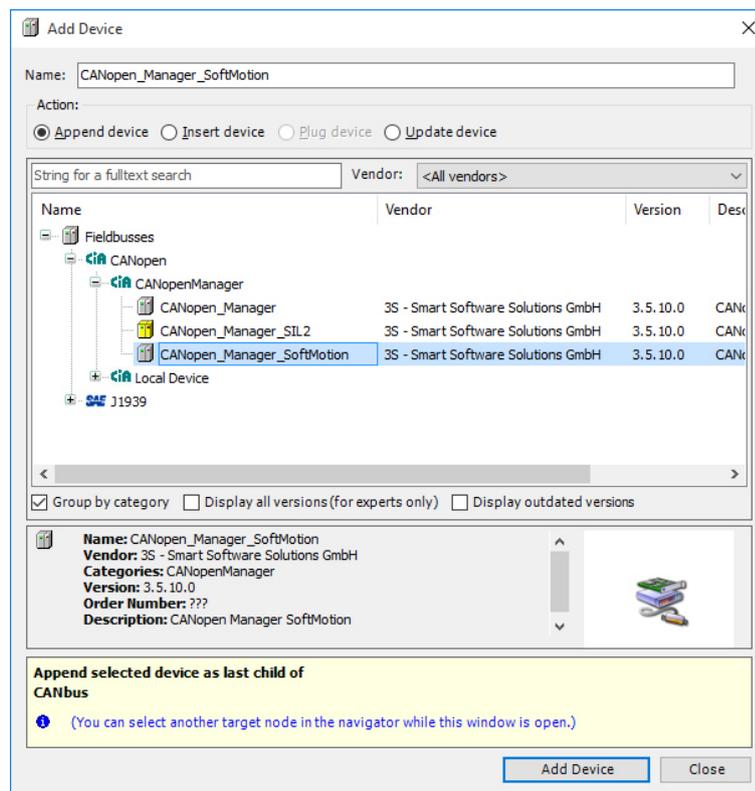
Als nächstes muss ein CANopen-Master hinzugefügt werden. Klicken Sie dazu mit der rechten Maustaste auf das Master-Gerät und wählen Sie **Add Device**.

Der CAN-Master muss auf die gleiche Bitrate konfiguriert werden, die für den Servoregler über den Metronix ServoCommander[®] ausgewählt wurde.



Um den Servoregler mit dem CAN-Master verbinden zu können, muss ein zusätzlicher CANopen SoftMotion Manager an den CAN-Master angehängt werden.

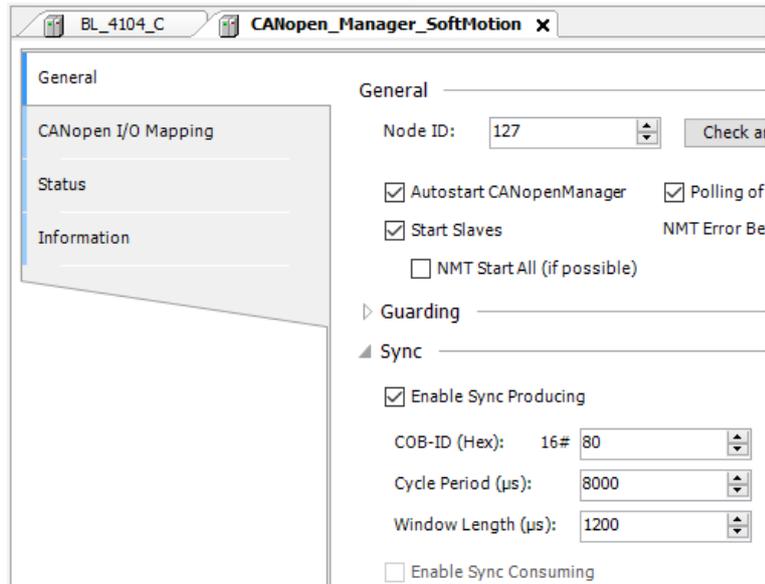
Dazu klicken Sie erneut mit der rechten Maustaste auf den CAN-Master und wählen **Add Device** aus.



› Zykluszeit einstellen

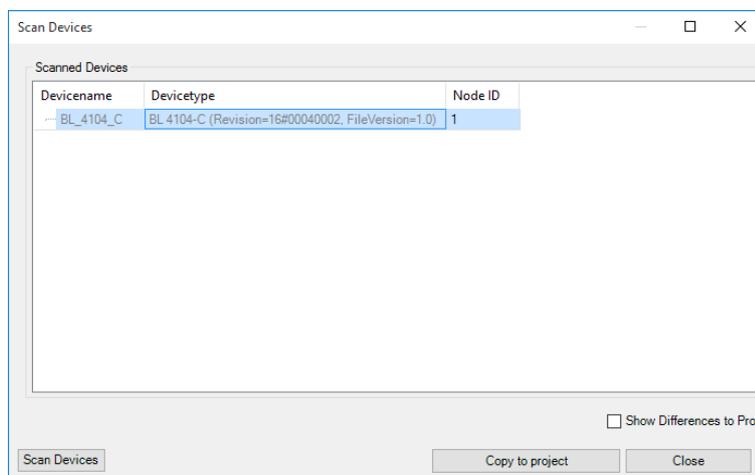
Der SoftMotion Manager läuft mit einer bestimmten Zykluszeit. Da in unserer Anwendung der zyklische PDO-Datenaustausch verwendet wird, synchronisiert der Master den Servoregler auf diese Zykluszeit. Dazu muss die Zykluszeit des Masters (**Cycle Period**) mit der im Servoregler konfigurierten Zykluszeit übereinstimmen.

Im Metronix ServoCommander® finden Sie den Dialog zur Konfiguration der Zykluszeit im Menü **Parameter\Reglerparameter\Zykluszeiten**. Nähere Informationen zum Einstellen der Zykluszeiten finden Sie im Abschnitt *Zykluszeiten der Regelkreise* im Produkthandbuch BL 4000-C oder im Produkthandbuch ARS 2000 FS.



› Geräte dem Projekt hinzufügen

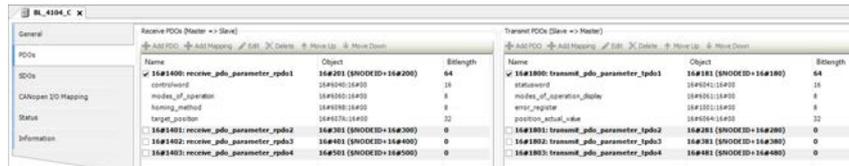
Die Generierung von Synchronisationstelegrammen muss abschließend im SoftMotion Manager aktiviert werden (**Enable Sync Producing**). Melden Sie sich durch Klicken auf die Schaltfläche **Online Config Mode** beim Master an. Suchen Sie nach Servoreglern am CANopen-Feldbus, indem Sie mit der rechten Maustaste auf den CANopen SoftMotion Manager klicken und **Nach Geräten suchen** wählen.



Alle am Feldbus angeschlossenen Servoregler werden erkannt und können durch Anklicken der Schaltfläche **Copy To Project** dem Projekt hinzugefügt werden. Dadurch werden die ausgewählten Servoregler als mit dem SoftMotion Manager verbundene Geräte dargestellt.

› PDO-Konfiguration einstellen

Sobald der Servoregler gefunden wurde, müssen die zyklischen Daten angegeben werden, die zwischen Servoregler und Master ausgetauscht werden sollen. Dies wird als PDO-Konfiguration bezeichnet und ist auf der Registerkarte **BL 4104-C** bzw. **ARS 2000 FS** zu finden.



Das standardmäßige PDO-Mapping verwendet nur die PDOs **1400_h** (TPDO0-Master►Slave) und **1800_h** (RPDO0-Master◄Slave).

Diese PDOs enthalten die folgenden Parameter, um den Servoregler als SoftMotion-Achse zu betreiben:

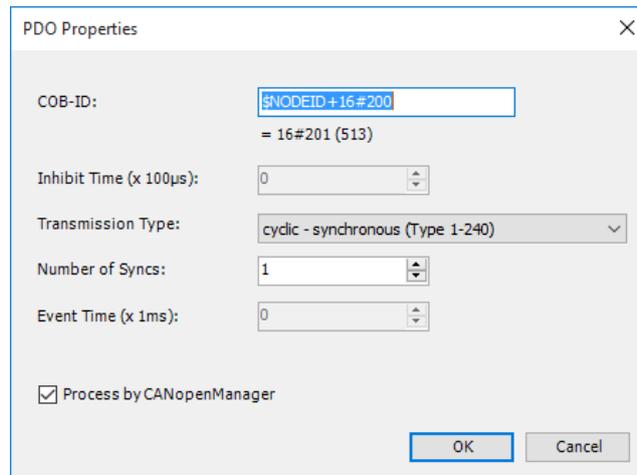
TPDO 0

Name	ID	Beschreibung	Siehe
controlword	6040 _h	Steuerwort zum Aktivieren / Deaktivieren des Servoreglers	Seite 112
modes_of_operation	6060 _h	Konfiguration des Betriebsmodus des Antriebs	Seite 133
homing_method	6098 _h	Konfigurieren der zu verwendenden Referenzfahrtmethode	Seite 135
target_position	607A _h	Neue Positions-Sollwerte	Seite 158

RPDO 0

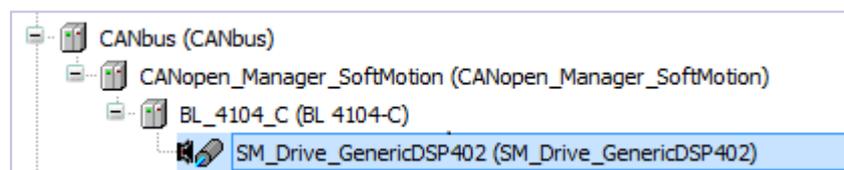
Name	ID	Beschreibung	Seite
statusword	6041 _h	Aktueller Status des Antriebs	Seite 112
modes_of_operation_display	6061 _h	Aktuelle Betriebsart des Antriebs	Seite 133
error_register	1001 _h	Aktueller Fehlercodes des Antriebs	Seite 187
position_actual_value	6064 _h	Positions-Istwert	Seite 73

Beide PDOs müssen auf "Zyklisches Senden bei 1 Sync" eingestellt sein. Dies geschieht durch die Bearbeitung des PDOs selbst.



Nach erfolgreicher PDO-Konfiguration kann dem Servoregler eine SoftMotion-Achse hinzugefügt werden.

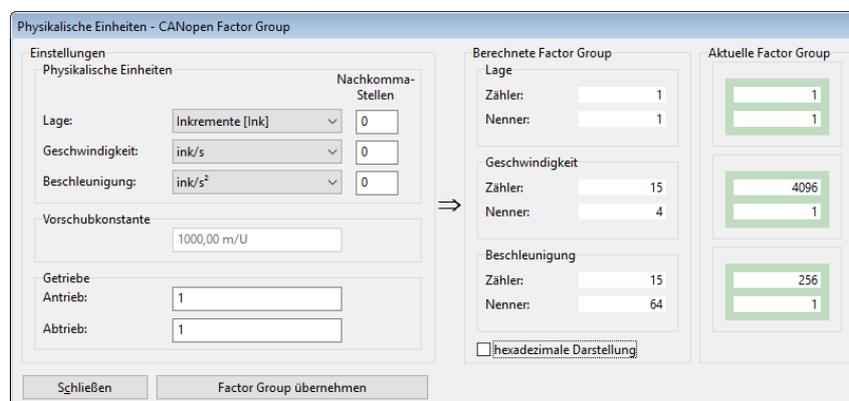
Wählen Sie dazu im SoftMotion Manager den Servoregler aus. Durch Rechtsklick auf den Listeneintrag ("BL_4104_C") öffnet sich ein Kontextmenü. Klicken Sie hier auf den Menüpunkt **Add SoftMotion CiA402 Axis**.



› Skalierung anpassen

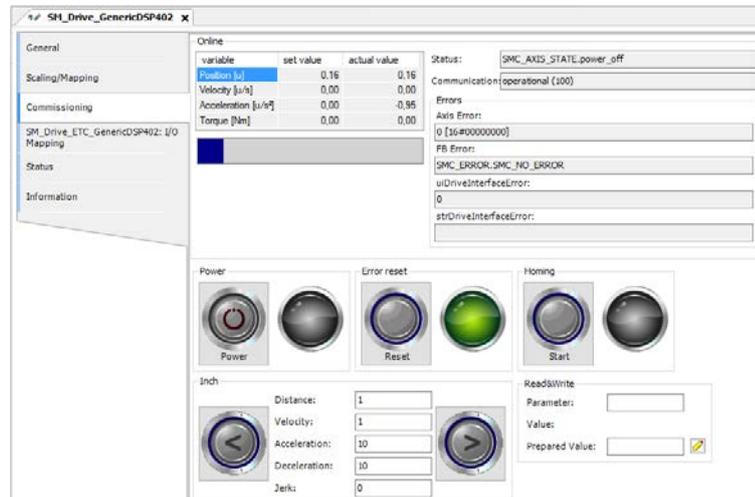
Damit die Einheiten der zyklisch ausgetauschten Werte (z.B. für Position und Drehzahl) zwischen Master und Servoregler passen, muss die Skalierung im Servoregler mit der Skalierung im Master übereinstimmen.

Unter **Parameter/Feldbus/CANopen/Anzeigeeinheiten** sollte daher folgende Skalierung eingestellt werden:



Mit dieser Skalierung sind maximal ± 32768 Umdrehungen mit 16 Bit Auflösung auf dem Bus darstellbar. Falls dies nicht ausreicht, kann die Skalierung der auf dem Bus übertragenen Sollwerte angepasst werden. Dies ist im Abschnitt 3.3 *Umrechnungsfaktoren (Factor Group)* auf Seite 47 beschrieben.

Bei korrekter Konfiguration der Skalierung sollten die Positionswerte nun im Fenster **Inbetriebnahme** der CoDeSys SoftMotion-Achse sichtbar sein:



Die Achse kann jetzt aus der Registerkarte **Inbetriebnahme** zum Testen verschoben werden. Zusätzlich ist die Achse nun bereit für die Implementierung in das SPS-Projekt. Eine detaillierte Beschreibung aller Parameter des Servoreglers und der implementierten Betriebsarten findet sich ab Abschnitt 3 *Parametrierung* auf Seite 40.

2.2 EtherCAT

EtherCAT ist ein von der Firma Beckhoff Automation entwickeltes Echtzeit-Ethernet. Um einen einfachen Umstieg von CAN auf EtherCAT zu ermöglichen, wurde das *CAN application protocol over EtherCAT* (CoE) definiert. Dadurch kann das Antriebsreglerprofil CiA 402 über EtherCAT verwendet werden.

2.2.1 Grundlagen

CoE basiert auf dem Feldbusprotokoll CANopen und verwendet daher das gleiche Objektverzeichnis und die gleichen Nachrichtentypen:

- **SDO (Servicedatenobjekte)**
Diese Art von Nachrichten wird für die azyklische Kommunikation zwischen Master und Slave verwendet, z.B. während der Initialisierungsphase der Anwendung oder in einer sehr einfachen Anwendung, bei der kein zyklischer Datenaustausch erforderlich ist.
- **PDO (Prozessdatenobjekte)**
Diese Art von Nachrichten wird zyklisch/automatisch zwischen Master und Slave ausgetauscht, um Prozessdaten auszutauschen. Prozessdaten sind alle Daten, die von Master oder Slave benötigt werden, um die Anwendung auszuführen. In unserem Beispiel enthalten diese Prozessdaten z.B. Positionssoll-/Istwerte, Steuer- und Statusworte und andere wichtige Informationen, um den Servoregler als SoftMotion-Achse nutzen zu können.

Zusätzlich ist der Nachrichtentyp **Emergency** Message verfügbar. Diese Nachricht wird vom Slave an den Master gesendet, wenn ein schwerer Fehler im Servoantrieb aufgetreten ist.

Andere Nachrichtentypen, wie z.B. **Sync**-Nachrichten, werden von EtherCAT CoE nicht unterstützt, da es andere Mechanismen gibt, um mehrere Slaves am Feldbus auf eine gemeinsame Uhr zu synchronisieren. Die wichtigsten sind **Distributed Clocks** (DC), die von der Gerätefamilie BL 4000-C vollständig unterstützt werden.

Die Synchronisation ist beispielsweise für Bewegungsanwendungen wichtig, bei denen mehrere Antriebe interpolierte Bewegungen ausführen sollen.

2.2.2 Verkabelung und Steckerbelegung

Die EtherCAT-Schnittstelle ist bei den Servoreglern ARS 2000 FS als steckbares Technologiemodul ausgeführt, wohingegen sie in den Servoreglern BL 4000-C bereits integriert ist.

Gemäß der EtherCAT-Spezifikation sind zwei RJ45-Stecker als RTE0 und RTE1 vorhanden [X21]. Einer für Uplink (Verbindung vom vorherigen Antrieb) und einer als Downlink (Verbindung zum nächsten Servoregler in der Leitung).

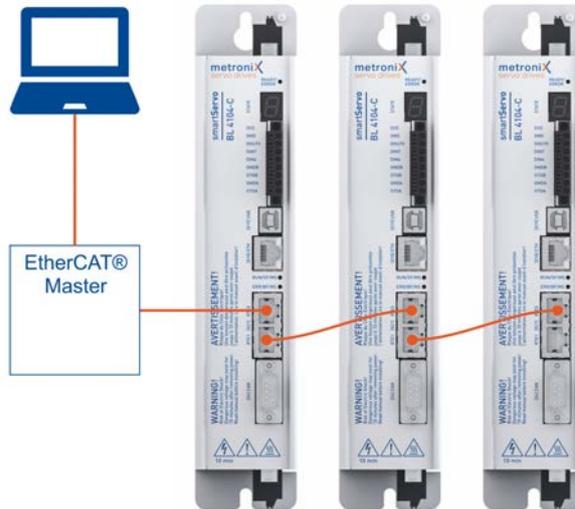


Die beiden Anschlüsse RTE0 und RTE1 sind RJ 45-Buchsen, Cat. 6

Pin	Bezeichnung	Beschreibung
1	RX-	Empfängersignal -
2	RX+	Empfängersignal +
3	TX-	Sendesignal -
4	-	-
5	-	-
6	TX+	Sendesignal +
7	-	-
8	-	-

2.2.3 Verkabelungs-Hinweise

Für die Verkabelung werden beim EtherCAT-Bus geschirmte Twisted Pair Ethernet-Kabel, welche STP, Cat.5 erfüllen, verwendet. Alle Knoten eines Netzwerks werden linienförmig miteinander verbunden.



2.2.4 Status LEDs

Zur einfachen Anzeige des EtherCAT-Bus-Status ist die Servoreglerfamilie BL 4000-C mit zwei Feldbusstatus-LEDs ausgestattet. Das Verhalten der LEDs ist durch die EtherCAT User Group (ETG) vordefiniert.

Die grüne RUN-LED zeigt den aktuellen EtherCAT® CoE-Zustand an:

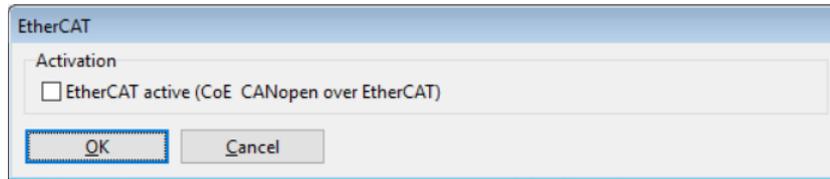
Blinkcode	Status der State Machine
LED ist aus	Noch keine Kommunikation.
LED blinkt	Vorbereitende Maßnahmen (PreOp) Der Master richtet den Slave für die zyklische Kommunikation ein. Es ist nur asynchrone Kommunikation über SDOs aktiv.
LED blinkt einmal auf	Sicherer Betrieb (SafeOp) Die zyklische Kommunikation über PDOs läuft. Der Slave ignoriert die Sollwertdaten, sendet aber Istwerte an den Master.
LED leuchtet	Betriebsbereit (OP) Der Slave übernimmt Sollwerte vom Master und folgt diesen.

Die rote ERR-LED zeigt die möglichen Feldbusfehler an:

Blinkcode	Status der State Machine
LED ist aus	Keine Fehler.
LED blinkt zweimal auf	Zyklischer Prozessdaten-Watchdog-Fehler Die Feldbuskommunikation ist unterbrochen. Der Slave hat vom Master keine Sollwerte empfangen.

2.2.5 EtherCAT aktivieren

Die EtherCAT-Feldbuskommunikation muss einmalig über das EtherCAT-Fenster des Metronix ServoCommander® aktiviert werden ([Parameter / Feldbus / EtherCAT / Betriebsparameter](#)).



HINWEIS Servoregler blockiert die Kommunikation zu nachfolgenden Slaves

Beachten Sie, dass ein Servoregler mit deaktivierter Ethercat-Schnittstelle die Kommunikation zu allen nachfolgenden Slaves am Feldbus blockiert. Daher sollte ein deaktivierter Servoregler von der Feldbusleitung abgeklemmt werden.

2.2.6 Integration des Servoreglers in ein Masterprojekt

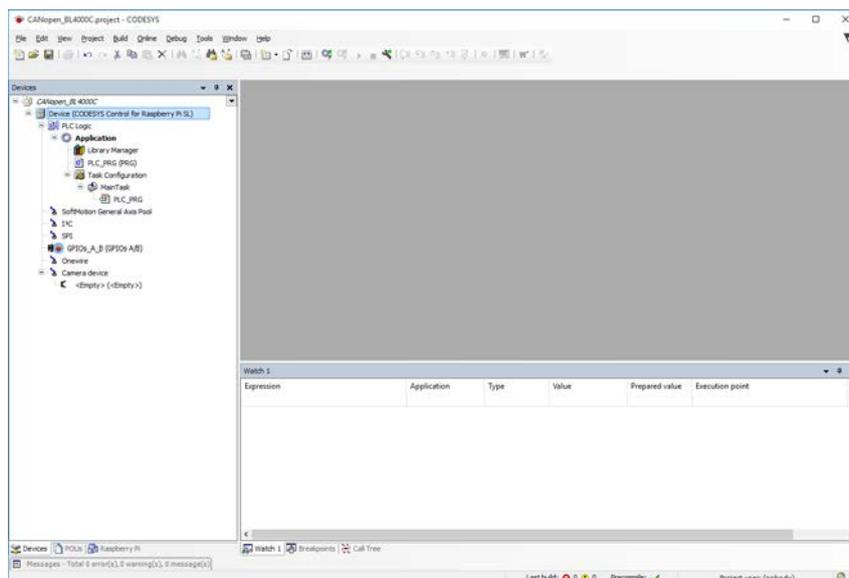
Als Beispiel soll der Servoregler BL 4000-C als SoftMotion-Achse in ein PLC-Projekt auf Basis von CoDeSys V3.5 und Beckhoff TwinCAT eingefügt werden. Die Integration eines Servoreglers ARS 2000 FS erfolgt in gleicher Weise.

› Integration in das CoDeSys V3.5 Projekt

Als Voraussetzung müssen Sie die EtherCAT-ESI-Datei für den entsprechenden Servoregler von der Metronix-Website (<https://www.metronix.de>) herunterladen. Diese Datei enthält eine vollständige Beschreibung der Antriebsmerkmale und des Objektverzeichnisses und wird vom CoDeSys (oder einem anderen EtherCAT-Master) zur automatischen Konfiguration des Servoantriebs verwendet.

Im Gegensatz zur CANopen EDS-Datei beinhaltet diese Datei nicht nur das Objektverzeichnis, sondern auch die komplette Konfiguration des Servoreglers, einschließlich der Auswahl von zyklisch ausgetauschten Soll- und Istwerten über PDOs, der Konfiguration der Feldbuszykluszeit und aller notwendigen Initialisierungsbefehle, die beim Hochfahren des Feldbusses an den Servoregler gesendet werden sollen.

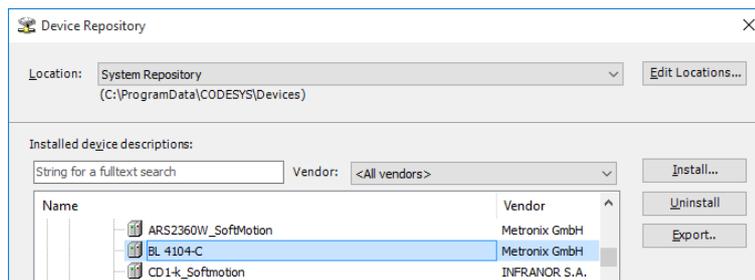
- Nach dem Herunterladen der ESI-XML-Datei verbinden Sie den Servoregler über ein Ethernet-Kabel mit dem CoDeSys-Master.
- Starten Sie anschließend CoDeSys, verbinden Sie sich mit Ihrem EtherCAT-Master und erstellen Sie ein leeres Projekt.



› ESI XML-Datei in das CoDeSys-Geräteverzeichnis installieren

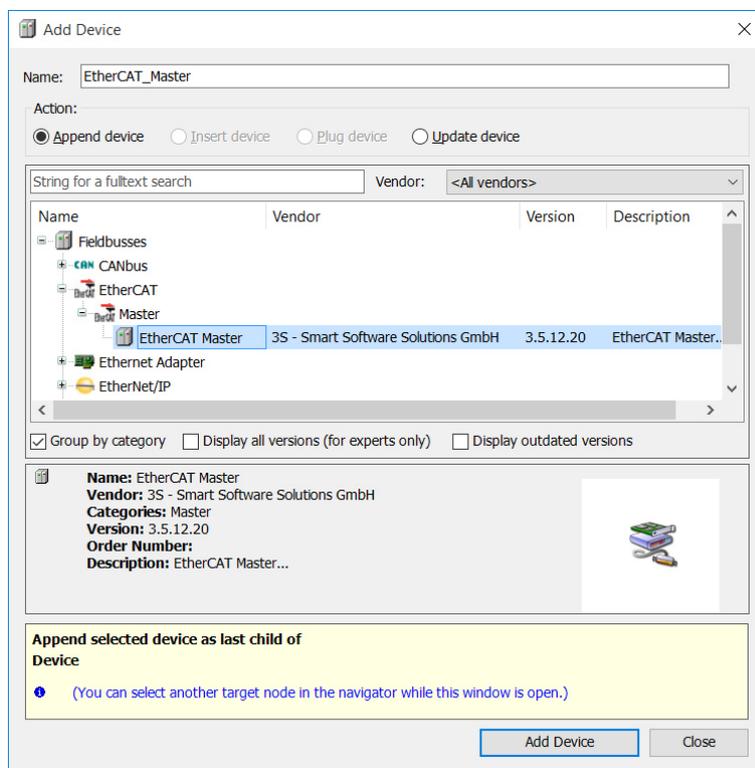
- Rufen Sie das CoDeSys-Geräteverzeichnis auf.
Pfad: Tools / Device Repository
- Klicken Sie die Schaltfläche **Installieren**.
- Wählen Sie die heruntergeladene EDS-Datei von Ihrem Speicherort aus.
- Bestätigen Sie durch Klick auf die Schaltfläche **Öffnen**.

Nun ist der CoDeSys-Software der Servoregler BL 4000-C bekannt und kann verwendet werden.



› EtherCAT-Master hinzufügen

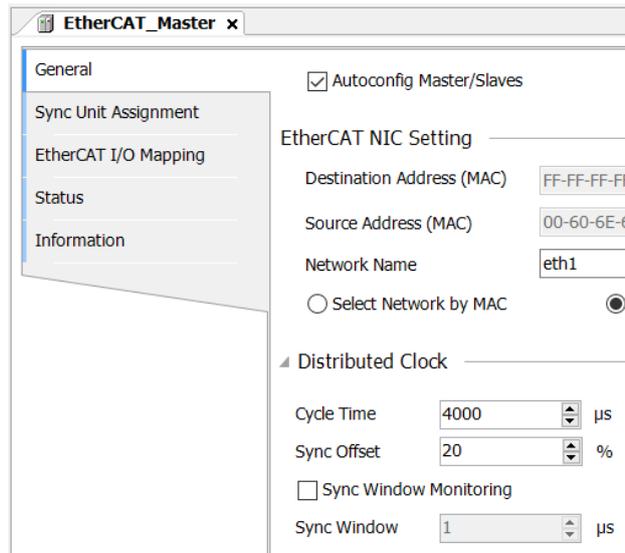
Als nächstes muss ein EtherCAT-Master hinzugefügt werden. Klicken Sie dazu mit der rechten Maustaste auf das Master-Gerät und wählen Sie **Add Device**.



› Zykluszeit einstellen

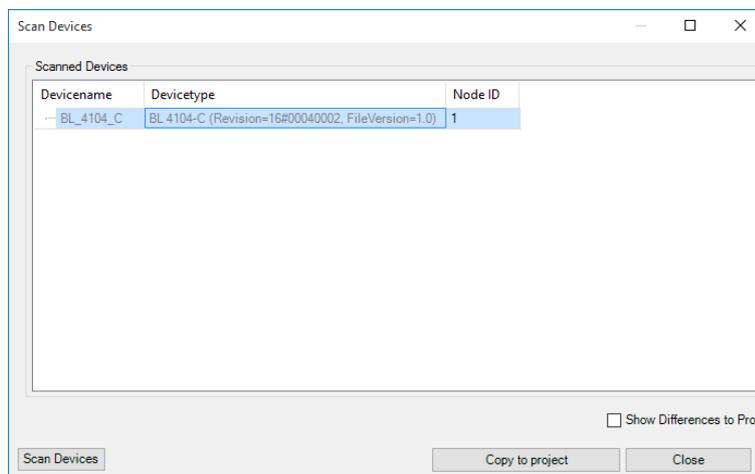
Der EtherCAT-Master tauscht mit einer bestimmten Zykluszeit PDOs mit dem Servoregler aus. Dazu wird der Servoregler vom Master auf diese Zykluszeit synchronisiert. Die Zykluszeit des Servoreglers muss daher mit der im EtherCAT-Master konfigurierten Zykluszeit (**Cycle Time**) übereinstimmen und Distributed Clock (DC) muss im Master aktiviert werden.

Im Metronix ServoCommander® finden Sie den Dialog zur Konfiguration der Zykluszeit im Menü **Parameter\Reglerparameter\Zykluszeiten**. Nähere Informationen zum Einstellen der Zykluszeiten finden Sie im Abschnitt *Zykluszeiten der Regelkreise* im Produkthandbuch BL 4000-C oder im Produkthandbuch ARS 2000 FS.



› Geräte dem Projekt hinzufügen

Die Generierung von Synchronisationstelegrammen muss abschließend im SoftMotion Manager aktiviert werden (**Enable Sync Producing**). Melden Sie sich durch Klicken auf die Schaltfläche **Online Config Mode** beim Master an. Suchen Sie nach Servoreglern an der EtherCAT-Schnittstelle, indem Sie mit der rechten Maustaste auf den EtherCAT-Master SoftMotion klicken und **Nach Geräten suchen** wählen.

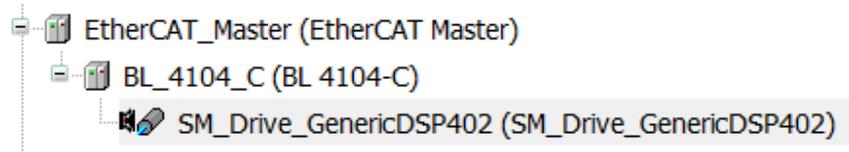


Alle am Feldbus angeschlossenen Servoregler werden erkannt und können durch Anklicken der Schaltfläche **Copy To Project** dem Projekt hinzugefügt werden. Dadurch werden die ausgewählten Servoregler als mit dem SoftMotion Manager verbundene Geräte dargestellt.

› PDO-Konfiguration einstellen

Im Gegensatz zu CANopen erfolgt die komplette PDO-Konfiguration der zyklischen Daten automatisch über die ESI-XML-Datei, so dass nun direkt eine SoftMotion-Achse zum Servoantrieb hinzugefügt werden kann.

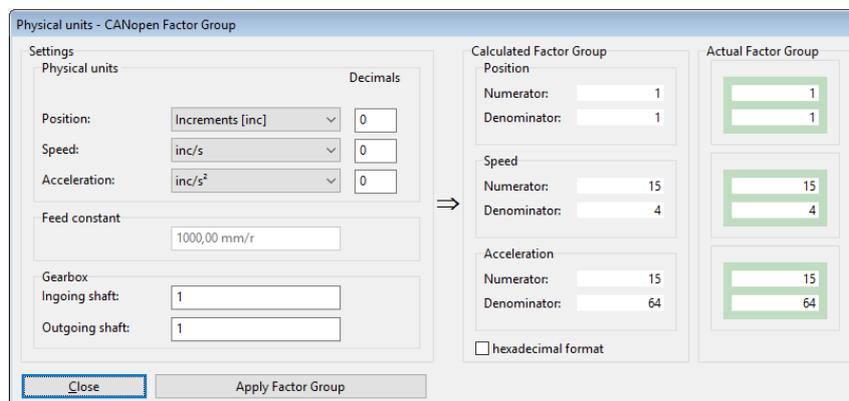
Rechtsklick auf den **BL 4104-C**, um eine DSP402-kompatible SoftMotion-Achse hinzuzufügen:



› Skalierung anpassen

Damit die Einheiten der zyklisch ausgetauschten Werte (z.B. für Position und Drehzahl) zwischen Master und Servoregler passen, muss die Skalierung angepasst werden.

Unter **Parameter/Feldbus/EtherCAT/Anzeigeeinheiten** sollte daher folgende Skalierung eingestellt werden:



The screenshot shows the 'Physical units - CANopen Factor Group' dialog box. It is divided into several sections:

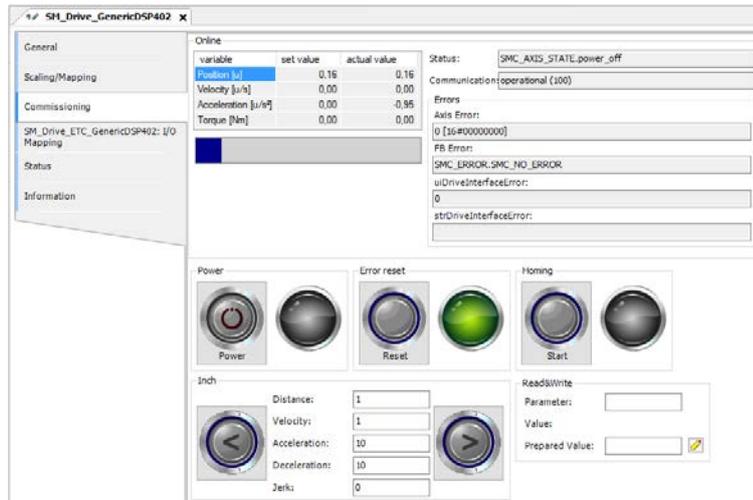
- Settings:**
 - Physical units:** A table with columns for 'Physical units' and 'Decimals'.

Physical units	Decimals
Position: Increments [inc]	0
Speed: inc/s	0
Acceleration: inc/s ²	0
 - Feed constant:** 1000,00 mm/r
 - Gearbox:** Ingoing shaft: 1, Outgoing shaft: 1
- Calculated Factor Group:**
 - Position:** Numerator: 1, Denominator: 1
 - Speed:** Numerator: 15, Denominator: 4
 - Acceleration:** Numerator: 15, Denominator: 64
- Actual Factor Group:** Three input fields with values 1, 15, and 64, each highlighted with a green border.
- hexadecimal format

Buttons at the bottom: 'Close' and 'Apply Factor Group'.

Mit dieser Skalierung sind maximal +-32768 Umdrehungen mit 16 Bit Auflösung auf dem Bus darstellbar. Falls dies nicht ausreicht, kann die Skalierung der auf dem Bus übertragenen Sollwerte angepasst werden. Dies ist im Abschnitt 3.3 *Umrechnungsfaktoren (Factor Group)* auf Seite 47 beschrieben.

Bei korrekter Konfiguration der Skalierung sollten die Positionswerte nun im Fenster **Inbetriebnahme** der CoDeSys SoftMotion-Achse sichtbar sein:



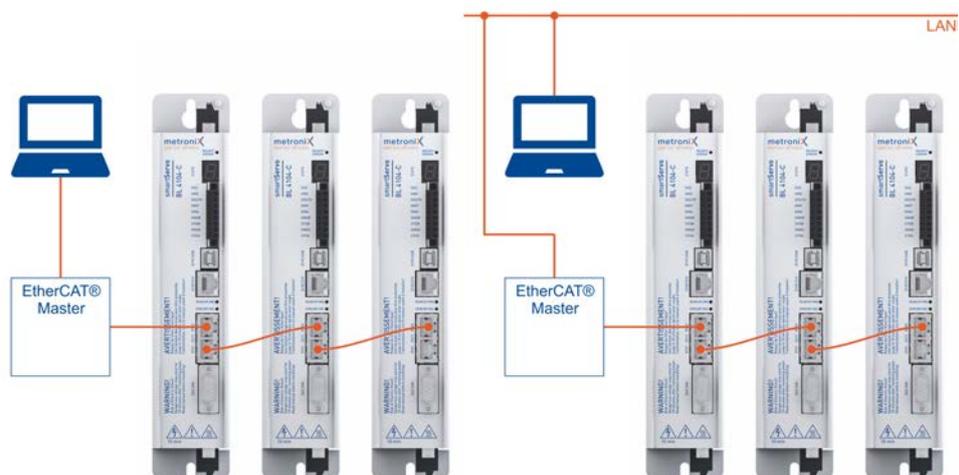
Die Achse kann jetzt aus der Registerkarte **Inbetriebnahme** zum Testen verschoben werden und ist nun bereit für die Implementierung in das SPS-Projekt.

2.2.7 EoE (Ethernet over EtherCAT®)

Servoregler der Baureihe BL 4000-C unterstützen das Profil EoE (Ethernet over EtherCAT®). Dabei werden normale Ethernet-Pakete vom Ethernet-Master mit über das EtherCAT®-Netzwerk geleitet (getunnelt). Dadurch kann der Metronix ServoCommander® Ethernet-Kommunikation mit den Servoreglern im EtherCAT®-Netzwerk aufbauen, ohne dass eine zusätzliche Verkabelung der LAN-Schnittstellen nötig ist.

EoE muss im Servoregler nicht gesondert aktiviert, sondern lediglich im EtherCAT®-Master konfiguriert werden.

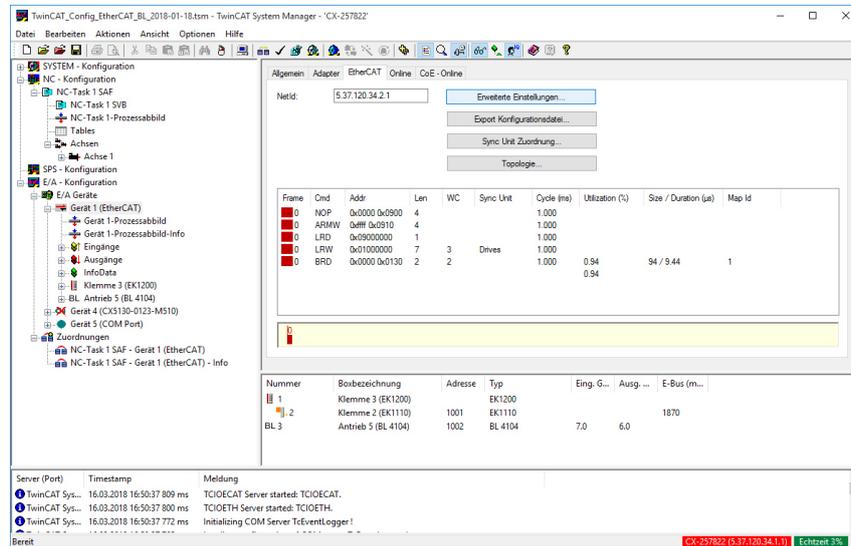
Für das Profil EoE gibt es zwei verschiedene Anschlussmöglichkeiten. Im ersten Fall wird der Laptop/PC, auf dem der Metronix ServoCommander® läuft, direkt mit der Steuerung verbunden, im zweiten Fall werden beide an einem gemeinsamen LAN betrieben.



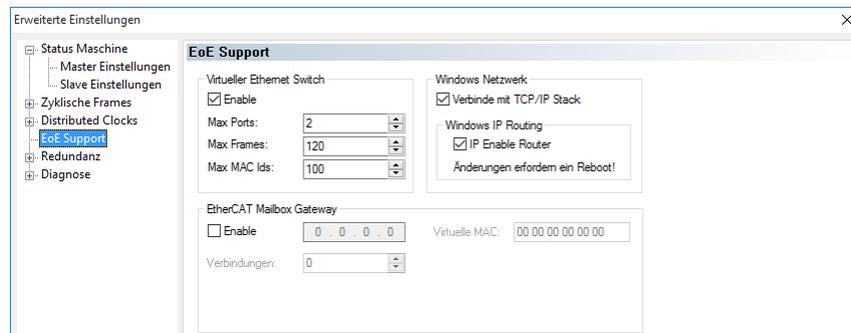
2.2.7.1 EoE im Master aktivieren

Die Aktivierung der EoE-Funktion ist im Folgenden am Beispiel einer Beckhoff Steuerung erläutert. Das Beispiel setzt voraus, dass bereits ein EtherCAT®-Netzwerk vorhanden ist und zyklische Kommunikation zu den Antrieben möglich ist.

Wählen Sie im TwinCAT System Manager **Gerät 1 (EtherCAT®)** aus und klicken in der Registerkarte **EtherCAT** auf **Erweiterte Einstellungen**.

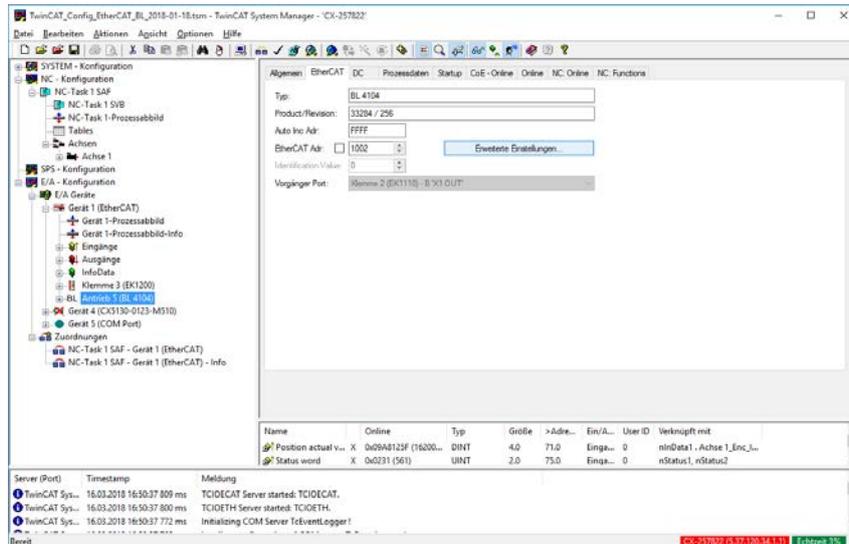


Wählen Sie den Eintrag **EoE Support** aus und aktivieren Sie **Virtueller Ethernet Switch** und **Verbinde mit TCP/IP Stack**. Im Abschnitt **Windows IP Routing** muss das Feld **IP Enable Router** ausgewählt sein. Dadurch wird in der Steuerung die Weiterleitung von Standard-Ethernet-Paketen aktiviert.

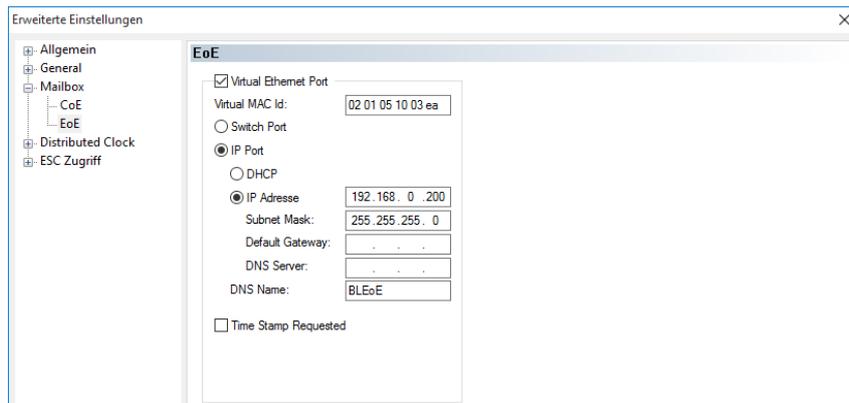


Abschließend muss für jeden Servoregler die EoE-Unterstützung im Servoregler aktiviert werden.

Wählen Sie jeweils den entsprechenden Antrieb aus, in diesem Beispiel **Antrieb 5 (BL 4104)** und klicken im Reiter **EtherCAT** auf **Erweiterte Einstellungen**.



Wählen Sie den Eintrag **Mailbox / EoE** aus, aktivieren Sie **Virtual Ethernet Port** und wählen Sie **IP Port** aus. An dieser Stelle haben Sie die Wahl, ob sie dem Gerät eine feste IP-Adresse zuweisen wollen oder diese dynamisch über DHCP bezogen werden soll. Dies setzt voraus, dass sich im Netzwerk ein entsprechender DHCP-Server befindet.



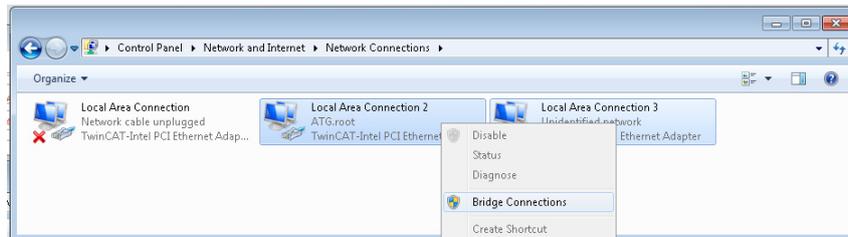
Abschließend muss die neue Konfiguration auf die Steuerung geladen und aktiviert werden. Der Servoregler wird nun in der Gerätesuche des Metronix ServoCommander[®] genauso angezeigt, als ob der Servoregler direkt über die Ethernet-Parametrierschnittstelle (X18) angeschlossen ist. Falls dies nicht der Fall ist, muss zusätzlich innerhalb der Beckhoff-Steuerung eine „Bridge“ aktiviert werden. Dies ist im folgenden Kapitel beschrieben.

2.2.7.2 Bridge konfigurieren

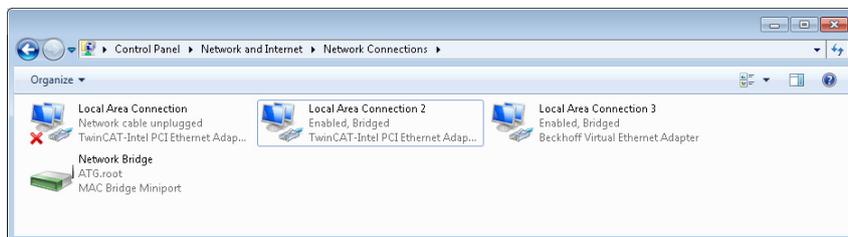
Um diese Einstellung vorzunehmen, müssen Sie sich direkt auf dem Betriebssystem der Beckhoff-Steuerung einloggen.

Wählen Sie in der Systemsteuerung Netzwerk und Internet aus. Markieren sie hier die passenden Ethernet-Verbindungen (in unserem Fall TwinCAT-Intel PCI Ethernet Adapter und Beckhoff Virtual Ethernet Adapter).

Drücken Sie die rechte Maustaste und wählen Sie **Bridge Connections** aus.



Anschließend wird eine Network Bridge angezeigt.



3 Parametrierung

Bevor der Servoregler die gewünschte Aufgabe (Momentenregelung, Drehzahlregelung, Positionierung) ausführen kann, müssen zahlreiche Parameter des Servoreglers an den verwendeten Motor und die spezifische Applikation angepasst werden. Dies kann entweder über den Metronix ServoCommander® erfolgen oder über CANopen.

Bei der Reihenfolge der Parametrierung kann sich an der Reihenfolge der anschließenden Kapitel orientiert werden. Ist der Servoregler bereits vollständig parametrierung, kann direkt mit dem Abschnitt 4 *Gerätesteuerung (Device Control)* auf Seite 112 bzw. Abschnitt 5 *Betriebsarten* auf Seite 133 fortgefahren werden.

HINWEIS Display des Servoreglers zeigt ein „A“ (Attention) an

Das Display des Servoreglers zeigt ein „A“ (Attention) an, wenn er noch nicht geeignet parametrierung wurde. Soll der Servoregler komplett über CANopen parametrierung werden, müssen Sie das Objekt 6510_h_C0_h beschreiben, um diese Anzeige zu unterdrücken. (Siehe Abschnitt 3.17.1.16 *Objekt 6510_h_C0_h: commissioning_state* auf Seite 109).

Neben den hier ausführlich beschriebenen Parametern sind im Objektverzeichnis des Servoreglers weitere Parameter vorhanden, die gemäß CANopen implementiert werden müssen. Sie enthalten aber in der Regel keine Information, die beim Aufbau einer Applikation mit Metronix Servoreglern sinnvoll verwendet werden kann. Bei Bedarf ist die Spezifikation solcher Objekte in den entsprechenden Standards (siehe Abschnitt 7.1 *CANopen* auf Seite 198) nachzulesen.

› Darstellung der Parameter

Alle Parameter des Antriebs werden in einer einheitlichen Darstellungsweise beschrieben. Handelt es sich beim Parameter um einen einfachen Datentyp (VAR), wird dieser folgendermaßen dargestellt:

Index	Index (hexadezimal)			
Name	Name des Parameters			
Info	Einheit	rw	PDO	Datentyp
Value	Wertebereich		Defaultwert	

Handelt es sich beim Parameter um einen zusammengesetzten Datentyp (ARRAY/RECORD), wird dieser folgendermaßen dargestellt:

Index	Index (hexadezimal)			
Name	Name der Parametergruppe			
Type	Object Code			Max
Sub-Index	Subindex (hexadezimal)			
Name	Name des Parameters			
Info	Einheit	rw	PDO	Datentyp
Value	Wertebereich		Defaultwert	

Die einzelnen Felder haben folgende Bedeutung:

Feld	Bedeutung
Index (hexadezimal)	Der Hauptindex des beschriebenen Parameters.
Subindex (hexadezimal)	Der Subindex des beschriebenen Parameters. Wenn dieser nicht angegeben ist, ist der Subindex Null.
Name der Parametergruppe	Klartext-Name der Parametergruppe.
Name des Parameters	Klartext-Name des Parameters.
Object Code	Angabe, ob es sich um einen einfachen oder zusammengesetzten Datentyp handelt: <ul style="list-style-type: none"> • VAR: Einfacher Datentyp • ARRAY: Gruppe von Parametern, die alle den identischen Datentyp haben. • RECORD: Gruppe von Parametern, die unterschiedliche Datentypen haben.
Max	Höchster Subindex der Gruppe.
Datentyp	Datentyp des Parameters oder des ARRAYS: Eine Liste der unterstützten Datentypen findet sich im Abschnitt 6.2 <i>SDO-Zugriff</i> auf Seite 173.
Einheit	Physikalische Einheit des Parameters.
Zugriff	Angabe ob der Parameter gelesen (ro), geschrieben (wo) oder gelesen und geschrieben (rw) werden darf.
PDO PDO	Angabe, ob der Parameter in ein PDO gemappt werden darf.
Wertebereich	Bereich, in dem zulässige Werte für diesen Parameter liegen.
Defaultwert	Wert, der im Auslieferungszustand bzw. nach erfolgreichem Schreiben auf das <i>Objekt 1011h: restore_default_parameters</i> wirksam ist.

3.1 Parametersätze laden und speichern

3.1.1 Übersicht

Der Servoregler verfügt über drei Parametersätze:

Aktueller Parametersatz

Dieser Parametersatz befindet sich im flüchtigen Speicher (RAM) des Servoreglers und enthält die Parameter, die aktuell verwendet werden. Er kann mit dem Parametrierprogramm Metronix ServoCommander® oder über den CAN-Bus beliebig gelesen und beschrieben werden. Beim Einschalten des Servoreglers wird der **Applikations-Parametersatz** in den **aktuellen Parametersatz** kopiert.

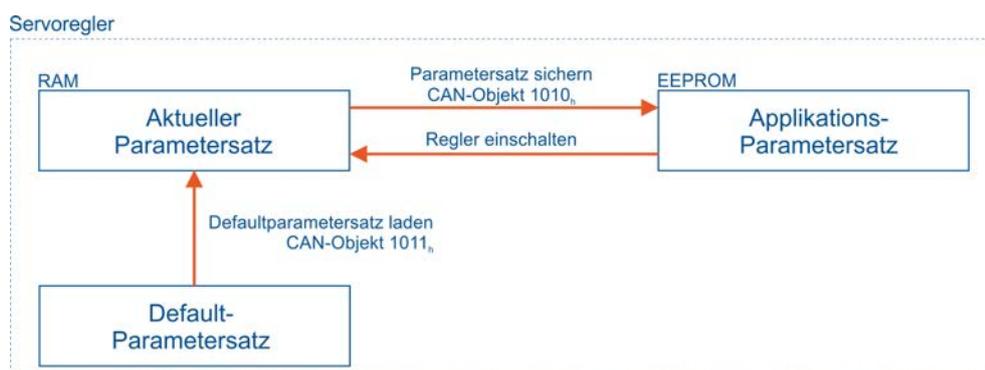
Applikations-Parametersatz

Der **aktuelle Parametersatz** kann in den nichtflüchtigen Flash-Speicher (EEPROM) gesichert werden, damit dieser nach dem nächsten Einschalten wieder zur Verfügung steht. Der Speichervorgang wird mit einem Schreibzugriff auf das CANopen-Objekt 1010_{h_01h} (`save_all_parameters`) ausgelöst.

Default-Parametersatz

Dieses ist der vom Hersteller standardmäßig vorgegebene unveränderliche Parametersatz des Servoreglers. Durch einen Schreibvorgang in das CANopen-Objekt 1011_{h_01h} (`restore_all_default_parameters`) kann der **Default-Parametersatz** in den **aktuellen Parametersatz** kopiert werden. Dieser Kopiervorgang ist nur bei ausgeschalteter Endstufe möglich.

Die nachfolgende Grafik veranschaulicht die Zusammenhänge zwischen den einzelnen Parametersätzen.



Es sind zwei unterschiedliche Konzepte zur Parametersatzverwaltung denkbar:

Konzept 1: Der Parametersatz wird mit dem Metronix ServoCommander® erstellt und ebenfalls mit dem Metronix ServoCommander® komplett in die einzelnen Servoregler 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 (Servoregleraustausch) die Parametriersoftware benötigt wird.

Konzept 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** durch Aufruf des CANopen-Objekts 1011_h_01_h ([restore_all_default_parameters](#)) geladen. Danach werden nur die abweichenden Objekte übertragen, was aufgrund der geringen Anzahl von Objekten sehr schnell geht. Vorteilhaft ist, dass dieses Verfahren auch bei unparametrierten Servoreglern funktioniert, so dass die Inbetriebnahme von neuen Anlagen oder der Austausch einzelner Servoregler unproblematisch ist und die Parametriersoftware Metronix ServoCommander® hierfür nicht benötigt wird.

⚠ VORSICHT Verletzungsgefahr durch falsch parametrieren Servoregler

Ein falsch parametrierter Servoregler kann unkontrollierte Drehbewegungen und dadurch Personenschäden oder Sachschäden verursachen.

Stellen Sie vor dem allerersten Einschalten der Endstufe sicher, dass der Servoregler die von Ihnen gewünschten Parameter enthält.

3.1.2 Beschreibung der Objekte

3.1.2.1 Objekt 1011_h: restore_default_parameters

Index	1011 _h		
Name	restore_parameters		
Type	ARRAY		01 _h
Sub-Index	01 _h		
Name	restore_all_default_parameters		
Info	--	rw	PDO UINT32
Value	64616F6C _h („load“), 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 08 00 00 22h erzeugt. Wird die falsche Kennung gesendet, wird der Fehler 08 00 00 20h erzeugt. Wird lesend auf das Objekt zugegriffen, wird eine 1 zurückgegeben, um anzuzeigen, dass das Zurücksetzen auf Defaultwerte unterstützt wird.

3.1.2.2 Objekt 1010_h: store_parameters

Index	1010 _h		
Name	store_parameters		
Type	ARRAY		01 _h
Sub-Index	01 _h		
Name	save_all_parameters		
Info	--	rw	PDO UINT32
Value	65766173 _h („save“), 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.

Wird das Objekt über ein SDO geschrieben, ist das Defaultverhalten, dass das SDO sofort beantwortet wird. Die Antwort spiegelt somit nicht das Ende des Speichervorgangs wider. Das Verhalten kann jedoch über das Objekt 6510_h_F0_h (compatibility_control) geändert werden.

3.2 Kompatibilitäts- Einstellungen

3.2.1 Übersicht

Um einerseits kompatibel zu früheren Gerätefamilien bleiben zu können und andererseits Änderungen und Korrekturen gegenüber der DSP402 und der DS301 ausführen zu können, wurde das Objekt `compatibility_control` eingefügt. Im Defaultparametersatz liefert dieses Objekt 0, d.h. Kompatibilität zu früheren Versionen. Für neue Applikationen empfehlen wir, die definierten Bits zu setzen, um so eine möglichst hohe Übereinstimmung mit den genannten Standards zu ermöglichen.

3.2.2 Beschreibung der Objekte

3.2.2.1 Objekt 6510_h_F0_h: `compatibility_control`

Index	6510 _h		
Name	drive_data		
Type	RECORD		F0 _h
Sub-Index	F0 _h		
Name	compatibility_control		
Info	--	rw	PBC UINT16
Value	0...7FFh, siehe Tabelle	--	

Bit	Name	Wert	Beschreibung
Bit 0	homing_method_scheme*	0001 _h	Das Bit hat die gleiche Bedeutung wie Bit 2 und ist aus Kompatibilitätsgründen vorhanden. Wird Bit 2 gesetzt, wird dieses Bit auch gesetzt und umgekehrt.
Bit 1	reserved	0002 _h	Das Bit ist reserviert. Es darf nicht gesetzt werden.
Bit 2	homing_method_scheme	0004 _h	Wenn dieses Bit gesetzt ist, sind die Referenzfahrtmethoden 32... 35 gemäß DSP402 nummeriert, anderenfalls ist die Nummerierung kompatibel zu früheren Metronix- Implementierungen. (Siehe auch Abschnitt 5.2.3 <i>Referenzfahrt-Abläufe</i> auf Seite 139). Wird dieses Bit gesetzt, wird auch Bit 0 gesetzt und umgekehrt
Bit 3	reserved	0008 _h	Das Bit ist reserviert. Es darf nicht gesetzt werden.

Bit	Name	Wert	Beschreibung
Bit 4	response_after_save	0010 _h	Wenn dieses Bit gesetzt ist, wird die Antwort auf save_all_parameters erst gesendet, wenn das Speichern abgeschlossen wurde. Dies kann mehrere Sekunden dauern, was ggf. zu einem Timeout in der Steuerung führt. Ist das Bit gelöscht, wird sofort geantwortet, es ist allerdings zu berücksichtigen, dass der Speichervorgang noch nicht abgeschlossen ist.
Bit 5	reserved	0020 _h	Das Bit ist reserviert. Es darf nicht gesetzt werden.
Bit 6	homing_to_zero	0040 _h	Unter CANopen besteht die Referenzfahrt nur aus 2 Phasen (Suchfahrt und Kriechfahrt). Der Antrieb fährt NICHT auf die ermittelte Nullposition (die z.B. durch den <code>homing_offset</code> zur gefundenen Referenzposition verschoben sein kann). Durch Setzen dieses Bits, wird die im Metronix ServoCommander® unter Fahrt auf Nullposition nach Referenzfahrt ausgewählte Option verwendet. Siehe hierzu Abschnitt 5.2 <i>Betriebsart Referenzfahrt (Homing Mode)</i> auf Seite 135
Bit 7	device_control	0080 _h	Wenn dieses Bit gesetzt ist, wird Bit 4 des statusword (<code>voltage_enabled</code>) gemäß DSP 402 v2.0 ausgegeben. Außerdem ist der Zustand <code>FAULT_REACTION_ACTIVE</code> vom Zustand <code>FAULT</code> unterscheidbar. Siehe hierzu Abschnitt 4 <i>Gerätesteuerung (Device Control)</i> auf Seite 112.
Bit 8	reserved	0100 _h	Das Bit ist reserviert. Es darf nicht gesetzt werden.
Bit 9	uzk_preload_ready	0200 _h	Wenn dieses Bit gesetzt ist, zeigt ein gesetztes Bit 4 (<code>voltage_enabled</code>) im statusword an, dass der Zwischenkreis vollständig geladen ist. Wenn dieses Bit gelöscht ist, zeigt Bit 4 an, dass die Endstufe eingeschaltet ist. Siehe hierzu Abschnitt 4 <i>Gerätesteuerung (Device Control)</i> auf Seite 112.
Bit 10	home_offset_sign	0400 _h	Wenn dieses Bit gesetzt ist, wird der home_offset (<code>607C_h</code>) von der Referenzposition abgezogen statt addiert, so dass der Antrieb nach der Referenzfahrt an der Position home_offset (statt <code>-home_offset</code>) steht.

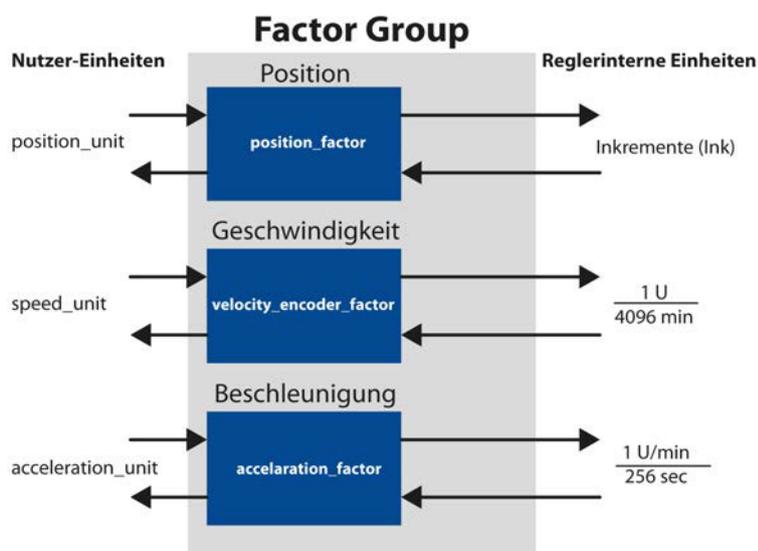
3.3 Umrechnungsfaktoren (Factor Group)

3.3.1 Übersicht

Die über den CAN-Bus übertragenen Werte werden in der Regel in der Steuerung derart umgerechnet, dass sie zur verwendeten Applikation passen. Ist dies nicht der Fall, kann mithilfe der **Factor Group** direkt die Skalierung der Werte angepasst werden, die auf dem Bus übertragen werden.

Dies kann ebenfalls nötig sein, wenn die Auflösung der auf dem Bus übertragenen Werte nicht ausreicht, z.B. weil bei den Standardeinstellungen nur +-32768 Umdrehungen unterschieden werden können.

Der Servoregler rechnet die Eingaben bzw. Ausgaben 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_unit`, `speed_unit` oder `acceleration_unit` bezeichnet. Die folgende Abbildung verdeutlicht die Funktion der Factor Group:



Alle Parameter werden im Servoregler grundsätzlich in 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 und während einer Parametrierung nicht geändert werden.**

Standardmäßig ist die **Factor Group** auf folgende Einheiten eingestellt:

Größe	Bezeichnung	Einheit	Erklärung
Länge	position_unit	Inkrement	65536 Inkremente pro Umdrehung
Geschwindigkeit	speed_unit	min ⁻¹	Umdrehung pro Minute
Beschleunigung	acceleration_unit	(min ⁻¹)/s	Drehzahlerhöhung in Umdrehung pro Minute pro Sekunde

3.3.2 Parametrierung der Factor Group

Die **Factor Group** kann komfortabel über den Metronix ServoCommander® eingestellt werden:

Parameter/Feldbus/CANopen/Anzeigeeinheiten bzw.
Parameter/Feldbus/Ethercat/Anzeigeeinheiten

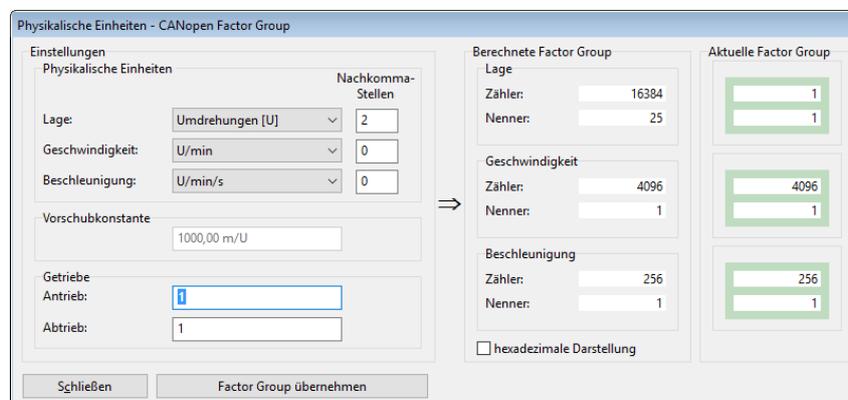


Abbildung 3: Fenster "CANopen Factor Group"

Unter **Einstellungen/Physikalische Einheiten** lässt sich die gewünschte Einheit für die Positionswerte (**Lage**), die **Geschwindigkeit** und die **Beschleunigung** separat auswählen. Zudem kann die gewünschte Anzahl an **Nachkommastellen** und ein **Getriebe** mit eingerechnet werden.

Wird als Lage-Einheit eine Längeneinheit ausgewählt, kann zudem die **Vorschubkonstante** angegeben werden.

Die Ergebnisse der so gewählten Einstellung werden unter **Berechnete Factor Group** angezeigt und können mit der Schaltfläche **Factor Group übernehmen** in den Servoregler übertragen werden.

3.3.3 Beschreibung der Objekte

3.3.3.1 Objekt 6093_h: position_factor

Das Objekt `position_factor` dient zur Umrechnung aller Längeneinheiten der Applikation von `position_unit` in die interne Einheit **Inkmente** (65536 Inkmente entsprechen 1 Umdrehung). Es besteht aus Zähler und Nenner. Der `position_factor` darf insgesamt nicht größer als 2^{24} sein.

Index	6093 _h			
Name	position_factor			
Type	ARRAY			02 _h
Sub-Index	01 _h			
Name	numerator			
Info	--	rw	PDO	UINT32
Value	--	1		
Sub-Index	02 _h			
Name	divisor			
Info	--	rw	PDO	UINT32
Value	--	1		

3.3.3.2 Objekt 6094_h: velocity_encoder_factor

Das Objekt `velocity_encoder_factor` dient zur Umrechnung aller Geschwindigkeitswerte der Applikation von `speed_unit` in die interne Einheit **Umdrehungen pro 4096 Minuten**. Es besteht aus Zähler und Nenner.

Index	6094 _h			
Name	velocity_encoder_factor			
Type	ARRAY			02 _h
Sub-Index	01 _h			
Name	numerator			
Info	--	rw	PDO	UINT32
Value	--	1000 _h		
Sub-Index	02 _h			
Name	divisor			
Info	--	rw	PDO	UINT32
Value	--	1		

3.3.3.3 Objekt 6097_h: acceleration_factor

Das Objekt `acceleration_factor` dient zur Umrechnung aller Beschleunigungswerte der Applikation von `acceleration_unit` in die interne Einheit **Umdrehungen pro Minute pro 256 Sekunden**. Es besteht aus Zähler und Nenner.

Index	6097 _h			
Name	acceleration_factor			
Type	ARRAY			02 _h
Sub-Index	01 _h			
Name	numerator			
Info	--	rw	PDO	UINT32
Value	--	100 _h		
Sub-Index	02 _h			
Name	divisor			
Info	--	rw	PDO	UINT32
Value	--	1		

3.3.3.4 Objekt 607E_h: polarity

Das Vorzeichen der Positions- und Geschwindigkeitswerte des Servoreglers kann mit dem Objekt `polarity` 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.

Das Setzen des `position_polarity_flag` bzw. des `velocity_polarity_flag` beeinflusst nur Parameter beim Lesen und beim Schreiben. Bereits im Servoregler vorhandene Parameter werden nicht verändert.

Index	607E _h			
Name	polarity			
Info	--	rw	PDO	UINT8
Value	0, 40 _h , 80 _h , C0 _h			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)

3.4 Endstufenparameter

3.4.1 Übersicht

Die Netzspannung wird über eine Vorladeschaltung in die Endstufe eingespeist. Beim Einschalten der Leistungsversorgung wird der Einschaltstrom begrenzt und das Laden überwacht. Nach erfolgter Vorladung des Zwischenkreises wird die Ladeschaltung überbrückt. Dieser Zustand ist Voraussetzung für das Erteilen der Servoreglerfreigabe.

Die gleichgerichtete Netzspannung wird mit den Kondensatoren des Zwischenkreises geglättet. Aus dem Zwischenkreis wird der Motor über die IGBTs gespeist. Die Endstufe enthält eine Reihe von Überwachungsfunktionen, die zum Teil parametrierbar sind:

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

3.4.2 Beschreibung der Objekte

3.4.2.1 Objekt 6510_h_10_h: enable_logic

Damit die Endstufe des Servoreglers aktiviert werden kann, müssen die digitalen Eingänge **Endstufenfreigabe** (Nur ARS 2000 FS) und **Servoreglerfreigabe** gesetzt sein: Die **Endstufenfreigabe** wirkt direkt auf die Ansteuersignale der Leistungstransistoren und würde diese auch bei einem defekten Mikroprozessor unterbrechen können. Das Wegnehmen der Endstufenfreigabe bei laufendem Motor bewirkt somit, dass der Motor ungebremst austrudelt bzw. nur durch die eventuell vorhandene Haltebremse gestoppt wird. Die **Servoreglerfreigabe** wird vom Mikrokontroller des Servoreglers verarbeitet. Je nach Betriebsart reagiert der Servoregler nach der Wegnahme dieses Signals unterschiedlich:

› Positionierbetrieb und drehzahl geregelter 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.

⚠ GEFAHR Lebensgefahr durch elektrischen Schlag! ⚠

Die Wegnahme der Servoreglerfreigabe bzw. der Endstufenfreigabe garantiert nicht, dass der Motor spannungsfrei ist.

Beim Betrieb des Servoreglers über den CAN-Bus können die beiden digitalen Eingänge **Endstufenfreigabe** und **Servoreglerfreigabe** gemeinsam auf 24V gelegt und die Freigabe über den CAN-Bus gesteuert werden. Dazu muss das Objekt 6510_h_10_h (**enable_logic**) auf zwei gesetzt werden. Aus Sicherheitsgründen erfolgt dies bei der Aktivierung von CANopen (auch nach einem Reset des Servoreglers) automatisch.

Index	6510 _h		
Name	drive_data		
Type	RECORD		F0 _h
Sub-Index	10 _h		
Name	enable_logic		
Info	--	rw	DD UINT16
Value	0...18 _h	--	

Wert	Bedeutung
0	Digitaler Eingang DIN5
1 _h	DIN5 + Parametrierschnittstelle
2 _h	DIN5 + CAN
3 _h	DIN5 + PROFIBUS/PROFINET
8 _h	DIN5 + EtherCAT
11 _h	Nur Parametrierschnittstelle
12 _h	Nur CAN
13 _h	Nur PROFIBUS/PROFINET
18 _h	Nur EtherCAT

3.4.2.2 Objekt 6510_h_30_h: pwm_frequency

Die Schaltverluste der Endstufe sind proportional zur Schaltfrequenz der Leistungstransistoren. Aus einigen Servoregler kann durch Halbieren der normalen PWM-Frequenz etwas mehr Leistung entnommen werden. Dadurch steigt allerdings die durch die Endstufe verursachte Stromwelligkeit. Die Umschaltung ist nur bei ausgeschalteter Endstufe möglich.

Index	6510 _h		
Name	drive_data		
Type	RECORD		F0 _h
Sub-Index	30 _h		
Name	pwm_frequency		
Info	--	rw	DD UINT16
Value	0, 1	0	

Wert	Bedeutung
0	Normale Endstufenfrequenz
1	Halbe Endstufenfrequenz

3.4.2.3 Objekt 6510_h_3A_h: enable_enhanced_modulation

Mit dem Objekt `enable_enhanced_modulation` kann die erweiterte Sinusmodulation aktiviert werden. Sie erlaubt eine bessere Ausnutzung der Zwischenkreisspannung und damit um ca. 14% höhere Drehzahlen. Nachteilig ist, dass das Regelverhalten und der Rundlauf des Motors bei sehr kleinen Drehzahlen geringfügig schlechter wird. Eine Änderung des Parameters darf nur bei ausgeschalteter Endstufe erfolgen und wird erst nach einem Reset wirksam. Dazu muss der Parametersatz zunächst gespeichert werden (`save_all_parameters`).

Index	6510 _h		
Name	drive_data		
Type	RECORD		F0 _h
Sub-Index	3A _h		
Name	enable_enhanced_modulation		
Info	--	rw	DD UINT16
Value	0, 1	0	

Wert	Bedeutung
0	Erweiterte Sinusmodulation AUS
1	Erweiterte Sinusmodulation EIN

3.4.2.4 Objekt 6510_h_31_h: power_stage_temperature

Die Temperatur der Endstufe kann über das Objekt `power_stage_temperature` ausgelesen werden. Wenn die im Objekt 6510_h_32_h (`max_power_stage_temperature`) angegebene Temperatur überschritten wird, schaltet die Endstufe aus und eine Fehlermeldung wird abgesetzt.

Index	6510 _h			
Name	drive_data			
Type	RECORD			F0 _h
Sub-Index	31 _h			
Name	power_stage_temperature			
Info	°C	ro	PDO	INT16
Value	--			--

3.4.2.5 Objekt 6510_h_32_h: max_power_stage_temperature

Die Temperatur der Endstufe kann über das Objekt 6510_h_31_h (`power_stage_temperature`) ausgelesen werden. Wenn die im Objekt `max_power_stage_temperature` angegebene Temperatur überschritten wird, schaltet die Endstufe aus und eine Fehlermeldung wird abgesetzt.

Index	6510 _h			
Name	drive_data			
Type	RECORD			F0 _h
Sub-Index	32 _h			
Name	max_power_stage_temperature			
Info	°C	ro	PDO	INT16
Value	--			--

Gerätetyp	Wert	Gerätetyp	Wert	Gerätetyp	Wert
ARS 2102 FS	100°C	ARS 2320 FS	80°C	BL 4102-C	85°C
ARS 2105 FS	80°C	ARS 2340 FS	80°C	BL 4104-C	85°C
ARS 2302 FS	80°C			BL 4304-C	90°C
ARS 2305 FS	80°C			BL 4308-C	85°C
ARS 2310 FS	80°C			BL 4312-C	75°C

3.4.2.6 Objekt 6510_h_33_h: nominal_dc_link_circuit_voltage

Über das Objekt `nominal_dc_link_circuit_voltage` kann die Gerätenennspannung in Millivolt ausgelesen werden.

Index	6510 _h		
Name	drive_data		
Type	RECORD	F0 _h	
Sub-Index	33 _h		
Name	nominal_dc_link_circuit_voltage		
Info	mV	ro	PDO UINT32
Value	--	--	

Gerätetyp	Wert	Gerätetyp	Wert	Gerätetyp	Wert
ARS 2102 FS	360000	ARS 2320 FS	560000	BL 4102-C	325000
ARS 2105 FS	360000	ARS 2340 FS	560000	BL 4104-C	325000
ARS 2302 FS	560000			BL 4304-C	560000
ARS 2305 FS	560000			BL 4308-C	560000
ARS 2310 FS	560000			BL 4312-C	560000

3.4.2.7 Objekt 6510_h_34_h: actual_dc_link_circuit_voltage

Über das Objekt `actual_dc_link_circuit_voltage` kann die aktuelle Spannung des Zwischenkreises in Millivolt ausgelesen werden.

Index	6510 _h		
Name	drive_data		
Type	RECORD	F0 _h	
Sub-Index	34 _h		
Name	actual_dc_link_circuit_voltage		
Info	mV	ro	PDO UINT32
Value	--	--	

3.4.2.8 Objekt 6510_h35_h: max_dc_link_circuit_voltage

Das Objekt `max_dc_link_circuit_voltage` gibt an, ab welcher Zwischenkreisspannung die Endstufe aus Sicherheitsgründen sofort ausgeschaltet und eine Fehlermeldung abgesetzt wird.

Index	6510_h		
Name	drive_data		
Type	RECORD	F0 _h	
Sub-Index	35_h		
Name	max_dc_link_circuit_voltage		
Info	mV	ro	PBC UINT32
Value	--	--	

Gerätetyp	Wert	Gerätetyp	Wert	Gerätetyp	Wert
ARS 2102 FS	460000	ARS 2320 FS	800000	BL 4102-C	439979
ARS 2105 FS	460000	ARS 2340 FS	800000	BL 4104-C	439979
ARS 2302 FS	800000			BL 4304-C	799976
ARS 2305 FS	800000			BL 4308-C	799976
ARS 2310 FS	800000			BL 4312-C	799976

3.4.2.9 Objekt 6510_h36_h: min_dc_link_circuit_voltage

Der Servoregler verfügt über eine Unterspannungsüberwachung. Diese kann über das Objekt 6510_h37_h (`enable_dc_link_undervoltage_error`) aktiviert werden. Das Objekt 6510_h36_h (`min_dc_link_circuit_voltage`) gibt die minimale Zwischenkreisspannung an. Unterhalb dieser Spannung wird der Fehler E 02-0 ausgelöst.

Index	6410_h		
Name	motor_data		
Type	RECORD	14 _h	
Sub-Index	36_h		
Name	min_dc_link_circuit_voltage		
Info	mV	rw	PBC UINT32
Value	0...1000000	--	

3.4.2.10 Objekt 6510_h_37_h: enable_dc_link_undervoltage_error

Mit dem Objekt `enable_dc_link_undervoltage_error` kann die Unterspannungsüberwachung aktiviert werden. Im Objekt 6510_h_36_h (`min_dc_link_circuit_voltage`) ist anzugeben, bis zu welcher unteren Zwischenkreisspannung der Servoregler arbeiten soll.

Index	6510 _h		
Name	drive_data		
Type	RECORD		F0 _h
Sub-Index	37 _h		
Name	enable_dc_link_undervoltage_error		
Info	--	rw	PBC UINT16
Value	0, 1	0	

Wert	Bedeutung
0	Unterspannungsfehler AUS (Reaktion WARNUNG)
1	Unterspannungsfehler EIN (Reaktion REGLERFREIGABE AUS)

Die Aktivierung des Fehlers E 02-0 erfolgt durch Änderung der Fehlerreaktion. Reaktionen, die zum Stillsetzen des Antriebs führen, werden als EIN, alle anderen als AUS zurückgegeben. Beim Beschreiben mit 0 wird die Fehlerreaktion WARNUNG gesetzt, beim Beschreiben mit 1 die Fehlerreaktion REGLERFREIGABE AUS. Siehe hierzu auch Abschnitt 3.18 *Fehlermanagement* auf Seite 110.

3.4.2.11 Objekt 6510_h_40_h: nominal_current

Mit dem Objekt `nominal_current` kann der Gerätenennstrom ausgelesen werden. Dies ist der obere Grenzwert, der in das Objekt 6075_h (`motorRatedCurrent`) eingeschrieben werden kann. Aufgrund eines Leistungsderating werden abhängig von der Servoregler-Zykluszeit und der Endstufentaktfrequenz gegebenenfalls andere Werte angezeigt.

Index	6510 _h		
Name	drive_data		
Type	RECORD		F0 _h
Sub-Index	40 _h		
Name	nominal_current		
Info	mA	ro	DDC UIN32
Value	--	siehe Tabelle	

Gerätetyp	Wert	Gerätetyp	Wert	Gerätetyp	Wert
ARS 2102 FS	2500	ARS 2320 FS	17427	BL 4102-C	2000
ARS 2105 FS	5000	ARS 2340 FS	34672	BL 4104-C	4000
ARS 2302 FS	2500			BL 4304-C	4000
ARS 2305 FS	5000			BL 4308-C	8000
ARS 2310 FS	7127			BL 4312-C	12000

3.4.2.12 Objekt 6510_h_41_h: peak_current

Mit dem Objekt `peak_current` kann der Gerätespitzenstrom ausgelesen werden. Dies ist der obere Grenzwert, der in das Objekt 6073_h (`max_current`) eingeschrieben werden kann. Aufgrund eines Leistungsderating werden abhängig von der Servoregler-Zykluszeit und der Endstufentaktfrequenz gegebenenfalls andere Werte angezeigt.

Index	6510 _h		
Name	drive_data		
Type	RECORD		F0 _h
Sub-Index	41 _h		
Name	peak_current		
Info	mA	ro	DDC UINT32
Value	--	siehe Tabelle	

Gerätetyp	Wert	Gerätetyp	Wert	Gerätetyp	Wert
ARS 2102 FS	5000	ARS 2320 FS	31461	BL 4102-C	6400
ARS 2105 FS	10000	ARS 2340 FS	53248	BL 4104-C	12800
ARS 2302 FS	7500			BL 4304-C	12000
ARS 2305 FS	15000			BL 4308-C	24000
ARS 2310 FS	14254			BL 4312-C	30000

3.5 Stromregler und Motoranpassung

ACHTUNG Sachschäden durch Falsche Einstellungen.

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

3.5.1 Übersicht

VORSICHT Verletzungsgefahr durch gefährliche Bewegungen

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.

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 Metronix 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.

3.5.2 Beschreibung der Objekte

3.5.2.1 Objekt 6075_h: motorRatedCurrent

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 Servoregler-Nennstromes (6510_{h_40_h}, nominalCurrent) liegt.

Index	6075 _h			
Name	motorRatedCurrent			
Info	mA	rw	PDO	UINT32
Value	0...nominalCurrent		--	

HINWEIS Objekte nicht unabhängig

Wird das Objekt 6075_h (motorRatedCurrent) mit einem neuen Wert beschrieben, muss in jedem Fall auch das Objekt 6073_h (maxCurrent) neu parametrieren werden.

3.5.2.2 Objekt 6073_h: maxCurrent

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, motorRatedCurrent) und wird in Tausendstel eingestellt. Der Wertebereich wird nach oben durch den maximalen Servoreglerstrom (Objekt 6510_{h_41_h}, peakCurrent) 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 (maxCurrent) darf erst beschrieben werden, wenn zuvor das Objekt 6075_h (motorRatedCurrent) gültig beschrieben wurde.

Index	6073 _h			
Name	maxCurrent			
Info	‰ (1000 = motorRatedCurrent)	rw	PDO	UINT16
Value	--		--	

3.5.2.3 Objekt 604D_h: pole_number

Die Polzahl des Motors ist dem Motordatenblatt oder dem Parametrierprogramm Metronix ServoCommander® zu entnehmen. Die Polzahl ist immer geradzahlig. Oft wird statt der Polzahl die Polpaarzahl angegeben. Die Polzahl entspricht dann der doppelten Polpaarzahl. Dieses Objekt wird durch [restore_default_parameters](#) nicht geändert. Es kann allerdings über den Metronix ServoCommander® unter [Datei / Parametersatz / Default-Parametersatz laden](#) zurückgesetzt werden.

Index	604D _h			
Name	pole_number			
Info	--	rw	PDO	UINT8
Value	2...254		--	

3.5.2.4 Objekt 6410_h11_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 Servoregler 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 Metronix ServoCommander® kann dieser Winkel bestimmt werden ([Parameter / Geräteparameter / Winkelgeber-Einstellungen](#)).

Der mit dem Metronix ServoCommander® bestimmte Winkel liegt im Bereich von ±180°. Er muss folgendermaßen umgerechnet werden:

$$\text{encoder_offset_angle} = \text{„Offsetwinkel des Winkelgebers“} \times 32767 / 180^\circ$$

Dieses Objekt wird durch [restore_default_parameters](#) nicht geändert. Es kann allerdings über den Metronix ServoCommander® unter [Datei / Parametersatz / Default-Parametersatz laden](#) zurückgesetzt werden.

Index	6410 _h			
Name	motor_data			
Type	RECORD			14 _h
Sub-Index	11 _h			
Name	encoder_offset_angle			
Info	180° / 32767	rw	PDO	INT16
Value	--		--	

3.5.2.5 Objekt 6410_h_10_h: phase_order

In der Phasenfolge (`phase_order`) werden Verdrehungen zwischen Motorkabel und Winkelgeberkabel berücksichtigt. Sie kann dem Parametrierprogramm Metronix ServoCommander[®] entnommen werden.

Dieses Objekt wird durch `restore_default_parameters` nicht geändert. Es kann allerdings über den Metronix ServoCommander[®] unter **Datei / Parametersatz / Default-Parametersatz laden** zurückgesetzt werden.

Index	6410 _h		
Name	motor_data		
Type	RECORD		14 _h
Sub-Index	10 _h		
Name	phase_order		
Info	--	rw	DD UINT16
Value	0, 1	0	

Wert	Bedeutung
0	Rechts
1	Links

3.5.2.6 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 I²t-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		
Type	RECORD		14 _h
Sub-Index	03 _h		
Name	iit_time_motor		
Info	ms	rw	DD UINT16
Value	0...10000	--	

3.5.2.7 Objekt 6410_h_04_h: iit_ratio_motor

Über das Objekt kann `iit_ratio_motor` kann die aktuelle Auslastung der I²t-Begrenzung des Motors in Promille ausgelesen werden.

Index	6410 _h			
Name	motor_data			
Type	RECORD	14 _h		
Sub-Index	04 _h			
Name	iit_ratio_motor			
Info	%o	ro	PDO	UINT16
Value	--	--		

3.5.2.8 Objekt 6510_h_3D_h: iit_ratio_servo

Über das Objekt kann `iit_ratio_servo` kann die aktuelle Auslastung der I²t-Begrenzung des Leistungsteils in Promille ausgelesen werden.

Index	6510 _h			
Name	drive_data			
Type	RECORD	F0 _h		
Sub-Index	3D _h			
Name	iit_ratio_servo			
Info	%o	ro	PDO	UINT16
Value	--	--		

3.5.2.9 Objekt 6510_h_38_h: iit_error_enable

Über das Objekt `iit_error_enable` wird festgelegt, wie sich der Servoregler bei Auftreten der I²t-Begrenzung verhält. Entweder wird dieses nur im `statusword` angezeigt, oder es wird Fehler E 31-0 ausgelöst.

Index	6510 _h		
Name	drive_data		
Type	RECORD	F0 _h	
Sub-Index	38 _h		
Name	iit_error_enable		
Info	--	rw	PDO UINT16
Value	0, 1	0	

Wert	Bedeutung
0	I ² t-Fehler AUS (Priorität WARNUNG)
1	I ² t-Fehler EIN (Priorität REGLERFREIGABE AUS)

Die Aktivierung des Fehlers 31-0 erfolgt durch Änderung der Fehlerreaktion. Reaktionen, die zum Stillsetzen des Antriebs führen, werden als EIN, alle anderen als AUS zurückgegeben. Beim Beschreiben mit 0 wird die Fehlerreaktion WARNUNG gesetzt, beim Beschreiben mit 1 die Fehlerreaktion REGLERFREIGABE AUS. Siehe Abschnitt 3.18 *Fehlermanagement* auf Seite 110.

3.5.2.10 Objekt 6510_h_2E_h: motor_temperature

Mit diesem Objekt kann die aktuelle Motortemperatur ausgelesen werden, falls ein analoger Temperatursensor angeschlossen ist. Anderenfalls ist das Objekt undefiniert.

Index	6510 _h		
Name	drive_data		
Type	RECORD	F0 _h	
Sub-Index	2E _h		
Name	motor_temperature		
Info	°C	ro	PDO INT16
Value	--	--	

3.5.2.11 Objekt 6410_h_14_h: motor_temperature_sensor_polarity

Über dieses Objekt kann festgelegt werden, ob ein Öffner oder ein Schließer als digitaler Motortemperatur-Sensor verwendet wird.

Index	6410 _h		
Name	motor_data		
Type	RECORD		14 _h
Sub-Index	14 _h		
Name	motor_temperature_sensor_polarity		
Info	--	rw	PDO INT16
Value	0, 1	0	

Wert	Bedeutung
0	Öffner
1	Schließer

3.5.2.12 Objekt 6510_h_2F_h: max_motor_temperature

Wird die in diesem Objekt definierte Motortemperatur überschritten, erfolgt eine Reaktion gemäß Fehlermanagement (Fehler E 03-0, Übertemperatur Motor analog). Ist eine Reaktion parametrierbar, die zum Stillsetzen des Antriebs führt, wird eine Emergency-Message gesendet. Zur Parametrierung des Fehlermanagements siehe Abschnitt 3.18 *Fehlermanagement* auf Seite 110.

Index	6510 _h		
Name	drive_data		
Type	RECORD		F0 _h
Sub-Index	2F _h		
Name	max_motor_temperature		
Info	°C	rw	PDO INT16
Value	20...300	--	

3.5.2.13 Objekt 60F6_h: torque_control_parameters

Die Daten des Stromreglers müssen dem Parametrierprogramm Metronix ServoCommander® entnommen werden. Hierbei sind folgende Umrechnungen zu beachten:

Die Verstärkung des Stromreglers muss mit 256 multipliziert werden. Bei einer Verstärkung von 1.5 im Menü „Stromregler“ des Parametrierprogramms Metronix ServoCommander® ist in das Objekt `torque_control_gain` der Wert $384 = 180_{\text{h}}$ einzuschreiben.

Die Zeitkonstante des Stromreglers ist im Parametrierprogramm Metronix ServoCommander® 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. Die untere Grenze darf nicht kleiner sein als die aktuelle Zykluszeit des Stromreglers (siehe 3.17.1.12 *Objekt 6510h_B0h: cycletime_current_controller* auf Seite 107).

Index	60F6_h		
Name	torque_control_parameters		
Type	RECORD		02 _h
Sub-Index	01_h		
Name	torque_control_gain		
Info	256 = „1“	rw	PBC UINT16
Value	0...(32*256)	--	
Sub-Index	02_h		
Name	torque_control_time		
Info	µs	rw	PBC UINT16
Value	104...64401	--	

3.5.2.14 Objekt 203A_h: torque_feed_forward

Gibt den Strom-Vorsteuerfaktor an. Dieser wird in 10^{-7} A pro eingestellter Beschleunigung parametrierbar. Somit kann ein über CANopen eingestelltes Beschleunigungsprofil abgefahren und der Strom beim Beschleunigen aufgezeichnet werden. Der Quotient aus Strom und Beschleunigung kann dann direkt in dieses Objekt geschrieben werden.

Index	203A_h		
Name	torque_feed_forward		
Info	A / (U/min/s)	rw	PBC UINT32
Value	0...208	--	

3.6 Drehzahlregler

3.6.1 Übersicht

ACHTUNG Sachschäden durch falsche Einstellungen

Falsche Einstellungen der Reglerparameter können zu starken Schwingungen führen und eventuell Teile der Anlage zerstören.

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 Metronix ServoCommander® optimal bestimmt werden.

3.6.2 Beschreibung der Objekte

3.6.2.1 Objekt 60F9_h: velocity_control_parameters

Die Daten des Drehzahlreglers können dem Parametrierprogramm Metronix ServoCommander® entnommen werden. Hierbei sind folgende Umrechnungen zu beachten:

Die Verstärkung des Drehzahlreglers muss mit 256 multipliziert werden. Bei einer Verstärkung von 1.5 im Menü „Drehzahlregler“ des Parametrierprogramms Metronix ServoCommander® ist in das Objekt `velocity_control_gain` der Wert 384 = 180_h einzuschreiben.

Die Zeitkonstante des Drehzahlreglers ist im Parametrierprogramm Metronix 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. Gleiches gilt für das Objekt `velocity_control_filter_time`, mit dem das Drehzahlwertfilter parametrier wird.

Index	60F9 _h		
Name	velocity_control_parameter_set		
Type	RECORD		04 _h
Sub-Index	01 _h		
Name	velocity_control_gain		
Info	256 = „1“	rw	PBO UINT16
Value	20...(64*256)	--	
Sub-Index	02 _h		
Name	velocity_control_time		
Info	µs	rw	PBO UINT16
Value	1...32000	--	

Sub-Index	04_h		
Name	velocity_control_filter_time		
Info	µs	rw	PBO UINT16
Value	1...32000	--	

3.6.2.2 Objekt 2073_h: velocity_display_filter_time

Mit dem Objekt `velocity_display_filter_time` kann die Filterzeit des Anzeigedrehzahl-Istwertfilters eingestellt werden.

Index	2073_h		
Name	velocity_display_filter_time		
Info	µs	rw	PBO UINT32
Value	1000...50000	--	

HINWEIS Objekt wird für den Durchdrehschutz verwendet

Beachten Sie, dass das Objekt `velocity_actual_value_filtered` für den Durchdrehschutz verwendet wird. Bei sehr großer Filterzeit wird ein Durchdrehfehler erst mit entsprechender Verzögerung erkannt.

3.7 Lageregler (Position Control Function)

3.7.1 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 Servoreglers umgerechnet.

› Schleppfehler (Following_Error)

Als Schleppfehler (`following_error_actual_value`) 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.

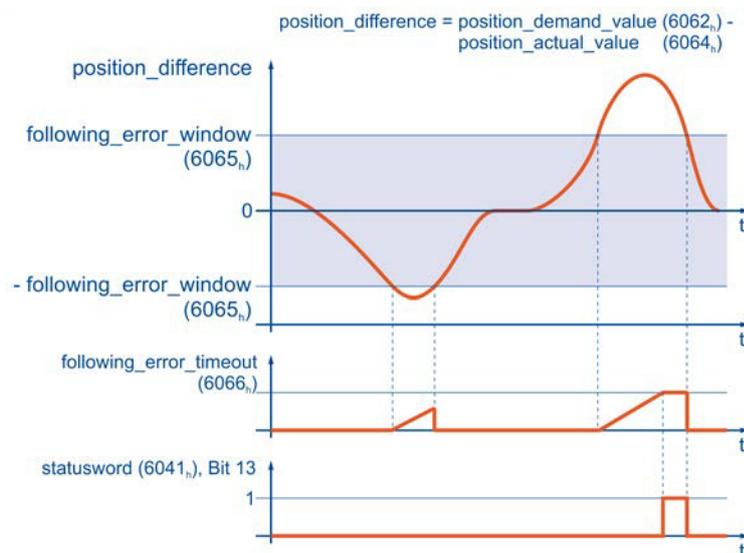


Abbildung 4: Schleppfehler – Funktionsübersicht

Die Abbildung 4 (*Schleppfehler – Funktionsübersicht*) zeigt, wie die Fensterfunktion für die Meldung „Schleppfehler“ definiert ist. Es wird überwacht, ob die Differenz aus Sollposition (`position_demand_value`) und Istposition (`position_actual_value`) das symmetrische Schleppfehlerfenster (`following_error_window`) verlässt. Kehrt die Positionsdifferenz nicht innerhalb einer Karenzzeit (`following_error_time_out`) in das Fenster zurück, wird Bit 13 im `statusword` gesetzt.

› 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, wird das damit verbundene Bit 10 (`target_reached`) im `statusword` gesetzt.

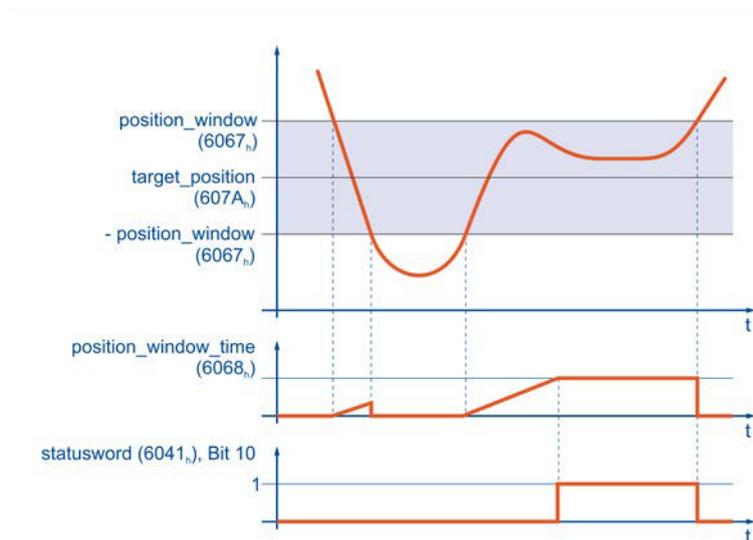


Abbildung 5: Position erreicht – Funktionsübersicht

Die Abbildung 5 (*Position erreicht – Funktionsübersicht*) zeigt, wie die Fensterfunktion für die Meldung „Position erreicht“ definiert ist. Es wird überwacht, ob sich die Istposition (`position_actual_value`) im symmetrischen Zielpositionsfenster (`target_position+position_window`, `target_position-position_window`) befindet. Bleibt die Positionsdifferenz länger als die Karenzzeit (`target_window_time`) im Zielfenster und ist die Positionierung abgeschlossen, wird Bit 10 im `statusword` gesetzt.

3.7.2 Beschreibung der Objekte

3.7.2.1 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 Metronix ServoCommander[®] optimal bestimmt werden.

ACHTUNG Sachschäden durch falsche Einstellungen

Falsche Einstellungen der Reglerparameter 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 Servoregler arbeitet daher intern mit Aufschaltungen, so dass die

Ausregelarbeit für den Lageregler minimiert wird und der Servoregler schnell einschwingen kann. Als Lageregler genügt normalerweise ein Proportional-Glied.

Die Daten des Lagereglers können dem Parametrierprogramm Metronix ServoCommander® entnommen werden. Hierbei sind folgende Umrechnungen zu beachten: Die Verstärkung des Lagereglers muss mit 256 multipliziert werden. Bei einer Verstärkung von 1.5 im Menü **Lageregler** des Parametrierprogramms Metronix ServoCommander® ist in das Objekt `position_control_gain` der Wert 384 einzuschreiben.

Normalerweise kommt der Lageregler ohne Integrator aus. Dann ist in das Objekt `position_control_time` der Wert Null einzuschreiben. Andernfalls muss die Zeitkonstante des Lagereglers in Mikrosekunden umgerechnet werden. Bei einer Zeit von 4.0 Millisekunden ist entsprechend der Wert 4000 in das Objekt `position_control_time` einzutragen. 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		
Type	RECORD		05 _h
Sub-Index	01_h		
Name	position_control_gain		
Info	256 = „1“	rw	PBQ UINT16
Value	0...(64*256)	--	
Sub-Index	02_h		
Name	position_control_time		
Info	µs	rw	PBQ UINT16
Value	0	--	
Sub-Index	04_h		
Name	position_control_v_max		
Info	speed_unit	rw	PBQ UINT32
Value	0...131072 min ⁻¹	--	
Sub-Index	05_h		
Name	position_error_tolerance_window		
Info	position_unit	rw	PBQ UINT32
Value	--	--	

3.7.2.2 Objekt 6062_h: position_demand_value

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

Index	6062 _h			
Name	position_demand_value			
Info	position_unit	ro	PDO	INT32
Value	--		--	

3.7.2.3 Objekt 202D_h: position_demand_sync_value

Über dieses Objekt kann die Soll-Lage des Synchronisationsgebers ausgelesen werden. Diese wird durch das Objekt 2022_h synchronization_encoder_select definiert. Dieses Objekt wird in benutzerdefinierten Einheiten angegeben.

Index	202D _h			
Name	position_demand_sync_value			
Info	position_unit	ro	PDO	INT32
Value	--		--	

3.7.2.4 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			
Info	position_unit	ro	PDO	INT32
Value	--		--	

3.7.2.5 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			
Info	ms	rw	PDO	UINT16
Value	0...27314		--	

3.7.2.6 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 parametrier

Index	6065_h			
Name	following_error_window			
Info	position_unit	rw	PDO	UINT32
Value	--		--	

3.7.2.7 Objekt 60F4_h: following_error_actual_value

Die aktuelle Differenz aus `position_demand_value` (6062_h) und `position_actual_value` (6064_h) kann aus diesem Objekt ausgelesen werden.

Index	60F4_h			
Name	following_error_actual_value			
Info	position_unit	ro	PDO	INT32
Value	--		--	

3.7.2.8 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			
Info	speed_unit	ro	PDO	INT32
Value	--		--	

3.7.2.9 Objekt 6410_h_0F_h: rotor_position

Über das Objekt kann `rotor_position` die Rotorlage in Promille einer Umdrehung ausgelesen werden.

Index	6410 _h			
Name	motor_data			
Type	RECORD	14 _h		
Sub-Index	0F _h			
Name	rotor_position			
Info	‰ (1000 = 1 U)	ro	PDO	UINT16
Value	--	--		

3.7.2.10 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			
Info	position_unit	rw	PDO	UINT32
Value	--	--		

3.7.2.11 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			
Info	ms	rw	PDO	UINT16
Value	--	--		

3.7.2.12 Objekt 6510_h_22_h: position_error_switch_off_limit

Im Objekt `position_error_switch_off_limit` kann die maximal zulässige Abweichung zwischen der Soll- und der Istposition eingetragen werden. Im Gegensatz zur o.g. Schleppfehlermeldung wird bei einer Überschreitung die Endstufe sofort abgeschaltet und ein Fehler ausgelöst. Der Motor trudelt somit ungebremst aus (außer es ist eine Haltebremse vorhanden).

Index	6510 _h		
Name	drive_data		
Type	RECORD		F0 _h
Sub-Index	22 _h		
Name	position_error_switch_off_limit		
Info	position_unit	rw	PBC UINT32
Value	--		--

Wert	Bedeutung
0	Grenzwert Schleppfehler AUS (Reaktion KEINE AKTION)
> 0	Grenzwert Schleppfehler EIN (Reaktion ENDSTUFE SOFORT ABSCHALTEN)

Die Aktivierung des Fehlers 17-0 erfolgt durch Änderung der Fehlerreaktion. Die Reaktion ENDSTUFE SOFORT ABSCHALTEN wird als EIN, alle anderen als AUS zurückgegeben. Beim Beschreiben mit 0 wird die Fehlerreaktion KEINE AKTION gesetzt, beim Beschreiben mit einem Wert größer 0 die Fehlerreaktion ENDSTUFE SOFORT ABSCHALTEN. Siehe hierzu auch Abschnitt 3.18 *Fehlermanagement* auf Seite 110.

3.7.2.13 Objekt 2030_h: set_position_absolute

Über das Objekt `set_position_absolute` kann die auslesbare Istposition verschoben werden, ohne dass sich die physikalische Lage ändert. Der Antrieb führt dabei keine Bewegung aus. Wenn ein absolutes Gebersystem angeschlossen ist, wird die Lageverschiebung im Geber gespeichert, sofern das Gebersystem dies zulässt. Die Lageverschiebung bleibt in diesem Fall also nach einem Reset erhalten. Diese Speicheroperation läuft unabhängig von diesem Objekt im Hintergrund ab. Es werden dabei ebenfalls alle dem Geberspeicher zugehörigen Parameter mit ihren aktuellen Werten gespeichert.

Index	2030 _h		
Name	set_position_absolute		
Info	position_unit	wo	PBC INT32
Value	--		--

3.7.2.14 Objekt 607D_h: software_position_limit

Die Objektgruppe `software_position_limit` enthält zwei Unterparameter, die den maximalen Positionierbereich beschränken. Verlässt der Antrieb im [Profile Position Mode](#) diesen Bereich, wird Fehler 40-0 (Negativer SW-Endschalter erreicht) bzw. 40-1 (Positiver SW-Endschalter erreicht) ausgelöst.

Index	607D_h			
Name	software_position_limit			
Type	ARRAY			02 _h
Sub-Index	01_h			
Name	min_position_limit			
Info	position_unit	rw	PDO	INT32
Value	--	--		
Sub-Index	02_h			
Name	max_position_limit			
Info	position_unit	rw	PDO	INT32
Value	--	--		

3.7.2.15 Objekt 607B_h: position_range_limit

Die Objektgruppe `position_range_limit` enthält zwei Unterparameter, die den numerischen Bereich der Positionswerte beschränken. Wenn eine dieser Grenzen überschritten wird, springt der Positionswert automatisch an die jeweils andere Grenze. Dieses ermöglicht die Parametrierung von sogenannten Rundachsen. Anzugeben sind die Grenzen, die physikalisch der gleichen Position entsprechen sollen, also beispielsweise 0° und 360°.

Damit diese Grenzen wirksam werden, muss über das Objekt 6510_{h_20h} (`position_range_limit_enable`) ein Rundachsenmodus ausgewählt werden.

Index	607B_h			
Name	position_range_limit			
Type	ARRAY			02 _h
Sub-Index	01_h			
Name	min_position_range_limit			
Info	position_unit	rw	PDO	INT32
Value	--	--		
Sub-Index	02_h			
Name	max_position_range_limit			
Info	position_unit	rw	PDO	INT32
Value	--	--		

3.7.2.16 Objekt 6510_h_20_h: position_range_limit_enable

Über das Objekt `position_range_limit_enable` können die durch das Objekt 607B_h definierten Bereichsgrenzen aktiviert werden. Es sind verschiedene Modi möglich:

Wird der Modus "Kürzester Weg" gewählt, werden Positionierungen immer auf der physikalisch kürzeren Strecke zum Ziel ausgeführt. Der Antrieb passt dazu selber das Vorzeichen der Fahrgeschwindigkeit an. Bei den beiden Modi "Feste Drehrichtung" erfolgt die Positionierung grundsätzlich nur in die im Modus angegebene Richtung.

Index	6510 _h		
Name	drive_data		
Type	RECORD	F0 _h	
Sub-Index	20 _h		
Name	position_range_limit_enable		
Info	--	rw	DD UINT16
Value	0...5	--	

Wert	Bedeutung
0	Aus
1	Kürzester Weg (Aus Kompatibilitätsgründen)
2	Kürzester Weg
3	Reserviert
4	Feste Drehrichtung „Positiv“
5	Feste Drehrichtung „Negativ“

3.8 Sollwert- Begrenzung

3.8.1 Objekt 2415_h: current_limitation

Mit der Objektgruppe `current_limitation` kann in den Betriebsarten `Profile Position Mode`, `Interpolated Position Mode`, `Cyclic Synchronous Position Mode`, `Homing Mode` und `Profile Velocity Mode` 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 (Fester Wert) oder der Vorgabe über einen analogen Eingang gewählt werden. Über das Objekt `limit_current` wird je nach gewählter Quelle entweder das Begrenzungsmoment (Quelle = Fester Wert) 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		
Type	RECORD		02 _h
Sub-Index	01 _h		
Name	limit_current_input_channel		
Info	--	rw	PDO INT8
Value	0...4	0	
Sub-Index	02 _h		
Name	limit_current		
Info	mA	rw	PDO INT32
Value	--	--	

Wert	Bedeutung
0	Keine Begrenzung
1	AIN0
2	AIN1
3	AIN2
4	Fester Wert / Feldbus (Feldbus-Selektor 2)

3.8.2 Objekt 2416_h: speed_limitation

Mit der Objektgruppe `speed_limitation` kann in der Betriebsart `profile_torque_mode` die Maximaldrehzahl des Motors begrenzt werden, wodurch ein drehzahlbegrenzter Drehmomentbetrieb ermöglicht wird. Über das Objekt `limit_speed_input_channel` wird die Sollwert-Quelle der Begrenzungsdrehzahl vorgegeben. Hier kann zwischen der Vorgabe eines direkten Sollwerts (Fester Wert) oder der Vorgabe über einen analogen Eingang gewählt werden. Über das Objekt `limit_speed` wird je nach gewählter Quelle entweder die Begrenzungsdrehzahl (Quelle = Fester Wert) oder der Skalierungsfaktor für die Analogeingänge (Quelle = Analogeingang) vorgegeben. Im ersten Fall wird direkt auf die angegebene Drehzahl begrenzt, im zweiten Fall wird die Drehzahl angegeben, die einer anliegenden Spannung von 10V entsprechen soll.

Index	2416_h		
Name	speed_limitation		
Type	RECORD		02 _h
Sub-Index	01_h		
Name	limit_speed_input_channel		
Info	--	rw	PBC INT8
Value	0...4	0	
Sub-Index	02_h		
Name	limit_speed		
Info	speed_unit	rw	PBC INT32
Value	--	--	

Wert	Bedeutung
0	Keine Begrenzung
1	AIN0
2	AIN1
3	AIN2
4	Fester Wert / Feldbus (Feldbus-Selektor 2)

3.9 Geberanpassungen

3.9.1 Übersicht

Dieses Kapitel beschreibt die Konfiguration des Winkelgebereingangs X2A, X2B und des Leitfrequenzeingangs (ARS 2000: X10, BL 4000-C: X1).

ACHTUNG Sachschäden durch falsche Winkelgeber-Einstellungen

Falsche Winkelgeber-Einstellungen können den Antrieb unkontrolliert drehen lassen und eventuell Teile der Anlage zerstören.

3.9.2 Beschreibung der Objekte

3.9.2.1 Objekt 2024_h: encoder_x2a_data_field

Im Record `encoder_x2a_data_field` sind Parameter zusammengefasst, die für den Betrieb des Winkelgebers am Stecker X2A notwendig sind.

Da zahlreiche Winkelgeber-Einstellungen nur nach einem Reset wirksam werden, sollte die Auswahl und die Einstellung der Geber über den Metronix ServoCommander[®] erfolgen. Unter CANopen lassen sich folgende Einstellungen auslesen bzw. ändern:

Das Objekt `encoder_x2a_resolution` gibt an, wie viele Inkremente vom Geber pro Umdrehung oder Längeneinheit erzeugt werden. Da am Eingang X2A nur Resolver angeschlossen werden können, die immer mit 16 Bit ausgewertet werden, wird hier immer 65536 zurückgegeben. Mit dem Objekt `encoder_x2a_numerator` und `encoder_x2a_divisor` kann ein eventuelles Getriebe (auch mit Vorzeichen) zwischen Motorwelle und Geber berücksichtigt werden.

Index	2024 _h		
Name	encoder_x2a_data_field		
Type	RECORD		03 _h
Sub-Index	01 _h		
Name	encoder_x2a_resolution		
Info	Inkremente (4 * Strichzahl)	ro	PBO UINT32
Value	--	--	
Sub-Index	02 _h		
Name	encoder_x2a_numerator		
Info	--	rw	PBO INT16
Value	-32768...32767 (außer 0)	1	

Sub-Index	03_h		
Name	encoder_x2a_divisor		
Info	--	rw	PDO INT16
Value	1...32767	1	

3.9.2.2 Objekt 2026_h: encoder_x2b_data_field

Im Record **encoder_x2b_data_field** sind Parameter zusammengefasst, die für den Betrieb des Winkelgebers am Stecker X2B notwendig sind.

Das Objekt **encoder_x2b_resolution** gibt an, wie viele Inkremente vom Geber pro Umdrehung erzeugt werden (Bei Inkrementalgebern entspricht dies dem Vierfachen der Strichzahl bzw. der Perioden pro Umdrehung).

Das Objekt **encoder_x2b_counter** liefert die aktuell gezählte Inkrementzahl. Es liefert daher Werte zwischen 0 und der eingestellten Inkrementzahl-1. Mit den Objekten **encoder_x2b_numerator** und **encoder_x2b_divisor** kann ein Getriebe (auch mit Vorzeichen) zwischen Motorwelle und dem an X2B angeschlossenen Geber berücksichtigt werden.

Index	2026_h		
Name	encoder_x2b_data_field		
Type	RECORD		16 _h
Sub-Index	01_h		
Name	encoder_x2b_resolution		
Info	Inkremente (4 * Strichzahl)	rw	PDO UINT32
Value	--	--	
Sub-Index	02_h		
Name	encoder_x2b_numerator		
Info	--	rw	PDO INT16
Value	-32768...32767 (außer 0)	1	
Sub-Index	03_h		
Name	encoder_x2b_divisor		
Info	--	rw	PDO INT16
Value	1...32767	1	
Sub-Index	04_h		
Name	encoder_x2b_counter		
Info	Inkremente (4 * Strichzahl)	ro	PDO UINT32
Value	0 ... (encoder_x2b_resolution – 1)	--	

3.9.2.3 Objekt 2025_h: encoder_x10_data_field

Im Record `encoder_X10_data_field` sind Parameter zusammengefasst, die für den Betrieb des Leitfrequenzeingangs notwendig sind. Bei der Gerätefamilie ARS 2000 befindet sich dieser auf dem Stecker X10, während er sich bei der Gerätefamilie smartServo BL 4000-C auf dem Stecker X1 befindet. Am Leitfrequenzeingang kann wahlweise ein digitaler Inkrementalgeber oder emulierte Inkrementalsignale beispielsweise eines anderen Servoreglers (Leitfrequenzausgang) angeschlossen werden. Die Signale des Leitfrequenzeingangs können wahlweise als Sollwert oder als Istwert verwendet werden.

Im Objekt `encoder_X10_resolution` muss angegeben werden, wie viele Inkremente vom Geber pro Umdrehung des Gebers erzeugt werden. Dies entspricht dem Vierfachen der Strichzahl. Das Objekt `encoder_X10_counter` liefert die aktuell gezählte Inkrementzahl (Zwischen 0 und der eingestellten Inkrementzahl-1).

Mit dem Objekt `encoder_X10_numerator` und `encoder_X10_divisor` kann ein eventuelles Getriebe (auch mit Vorzeichen) berücksichtigt werden.

Bei der Verwendung des X10- Signals als Istwert entspräche dies einem Getriebe zwischen dem Motor und dem an X10 angeschlossenen Istwertgeber, welcher am Abtrieb montiert ist. Bei der Verwendung des X10- Signals als Sollwert, können hiermit Getriebeübersetzungen zwischen Master und Slave realisiert werden.

Index	2025 _h			
Name	encoder_x10_data_field			
Type	RECORD			05 _h
Sub-Index	01 _h			
Name	encoder_x10_resolution			
Info	Inkremente (4 * Strichzahl)	rw	PDO	UINT32
Value	geberabhängig	--		
Sub-Index	02 _h			
Name	encoder_x10_numerator			
Info	--	rw	PDO	INT16
Value	-32768...32767 (außer 0)	1		
Sub-Index	03 _h			
Name	encoder_x10_divisor			
Info	--	rw	PDO	INT16
Value	1...32767	1		
Sub-Index	04 _h			
Name	encoder_x10_counter			
Info	Inkremente (4 * Strichzahl)	ro	PDO	UINT32
Value	0 ... (encoder_x10_resolution - 1)	--		

Sub-Index	05_h			
Name	encoder_x10_position			
Info	--	ro	PDO	INT32
Value	--	--		

3.9.2.4 Objekt 202C_h: max_comm_enc_pos_enc_difference

Das Objekt `max_comm_enc_pos_enc_difference` gibt die maximale Differenz zwischen dem Kommutiergeber und Lageistwertgeber zurück.

Index	202C_h			
Name	max_comm_enc_pos_enc_difference			
Info	position_unit	rw	PDO	INT32
Value	--	--		

3.10 Leitfrequenzausgang

3.10.1 Übersicht

Diese Objektgruppe ermöglicht es, den Leitfrequenzausgang (ARS 2000 FS: X11, BL 4000-C: X1) zu parametrieren. Somit können Master-Slave-Applikationen, bei denen der Leitfrequenzausgang (Inkrementalgeber-Emulation) des Masters an den Leitfrequenzeingang des Slave angeschlossen ist, unter CANopen parametriert werden.

3.10.2 Beschreibung der Objekte

3.10.2.1 Objekt 201A_h: encoder_emulation_data

Der Objekt-Record `encoder_emulation_data` kapselt alle Einstellmöglichkeiten für den Leitfrequenzausgang.

Über das Objekt `encoder_emulation_resolution` kann die ausgegebene Inkrementzahl (= vierfache Strichzahl) als Vielfaches von 4 frei eingestellt werden. In einer Master-Slave-Applikation muss diese der `encoder_X10_resolution` des Slave entsprechen, um ein Verhältnis von 1:1 zu erreichen.

Mit dem Objekt `encoder_emulation_offset` kann die Position des ausgegebenen Nullimpulses gegenüber der Nulllage des Istwertgebers verschoben werden.

Index	201A_h		
Name	encoder_emulation_data		
Type	RECORD		02 _h
Sub-Index	01_h		
Name	encoder_emulation_resolution		
Info	Inkmente (4 * Strichzahl)	rw	DD INT32
Value	4 * (1...8192)	--	
Sub-Index	02_h		
Name	encoder_emulation_offset		
Info	32767 = 180°	rw	DD INT16
Value	-32768...32767	--	

3.10.2.2 Objekt 2028_h: encoder_emulation_resolution

Das Objekt `encoder_emulation_resolution` ist nur aus Kompatibilitätsgründen vorhanden. Es entspricht dem Objekt 201A_h01_h.

3.11 Soll- / Istwertaufschaltung

3.11.1 Übersicht

Mit Hilfe der nachfolgenden Objekte kann die Quelle für den Sollwert und den Istwert geändert werden. Als Standard verwendet der Servoregler den Eingang für den Motorgeber X2A bzw. X2B als Istwert für den Lageregler. Bei Verwendung eines externen Lagegebers, z.B. hinter einem Getriebe, kann der über X10 eingespeiste Lagewert als Istwert für den Lageregler aufgeschaltet werden. Darüber hinaus ist es möglich über X10 eingehende Signale (z.B. eines zweiten Servoreglers) als zusätzlichen Sollwert aufzuschalten, wodurch Synchronbetriebsarten ermöglicht werden.

3.11.2 Beschreibung der Objekte

3.11.2.1 Objekt 201F_h: commutation_encoder_select

Das Objekt `commutation_encoder_select` gibt den Gebereingang an, der als Kommutiergeber verwendet wird. Da dieser Wert erst nach einem Reset wirksam wird, sollte die Einstellung des Kommutiergebers grundsätzlich über den Metronix ServoCommander[®] erfolgen.

Index	201F _h		
Name	commutation_encoder_select		
Info	--	rw	PBC INT16
Value	0, 2		--

Wert	Bezeichnung
0	X2A
2	X2B

3.11.2.2 Objekt 2021_h: position_encoder_selection

Das Objekt `position_encoder_selection` gibt den Gebereingang an, der zur Bestimmung der Istlage (Istwertgeber) verwendet wird. Dieser Wert kann geändert werden, um auf Lageregelung über einen externen (am Abtrieb angeschlossenen) Geber umzuschalten. Dabei kann zwischen X10 und dem als Kommutiergeber ausgewählten Gebereingang (X2A / X2B) umgeschaltet werden. Wird einer der Gebereingänge X2A / X2B als Lageistwertgeber ausgewählt, so muss derjenige verwendet werden, der als Kommutiergeber genutzt wird. Wird der jeweils andere Geber angewählt, wird automatisch auf den Kommutiergeber umgeschaltet.

Index	2021 _h		
Name	position_encoder_selection		
Info	--	rw	PBC INT16
Value	0...2	--	

Wert	Bezeichnung
0	X2A
1	X2B
2	X10

HINWEIS Konfiguration

Es kann nur zwischen dem Gebereingang X10 und dem jeweiligen Kommutiergeber X2A oder X2B als Lageistwertgeber gewählt werden. Die Konfiguration X2A als Kommutiergeber und X2B als Lageistwertgeber zu nutzen, bzw. umgekehrt, ist nicht möglich.

3.11.2.3 Objekt 2022_h: synchronisation_encoder_selection

Das Objekt `synchronisation_encoder_selection` gibt den Gebereingang an, der als Synchronisationssollwert verwendet wird. Je nach Betriebsart entspricht dieses einem Lagesollwert ([Profile Position Mode](#)) oder einem Drehzahlsollwert ([Profile Velocity Mode](#)).

Als Synchronisationseingang kann nur X10 verwendet werden. Somit kann zwischen X10 und "Kein Geber" ausgewählt werden. Als Synchronisationssollwert sollte nicht der gleiche Eingang wie für den Istwertgeber gewählt werden.

Index	2022 _h		
Name	synchronisation_encoder_selection		
Info	--	rw	PBC INT16
Value	-1, 2	--	

Wert	Bezeichnung
-1	Kein Geber / undefiniert
2	X10

3.11.2.4 Objekt 202F_h: synchronisation_selector_data

Über das Objekt `synchronisation_main` kann die Aufschaltung eines Synchronsollwerts erfolgen. Damit der Synchronsollwert überhaupt berechnet wird, muss Bit 0 gesetzt werden. Bit 1 ermöglicht es die Synchronlage erst durch das Starten eines Positionssatzes aufzuschalten (fliegende Säge). Über das Bit 8 kann festgelegt werden, dass die Referenzfahrt ohne Aufschaltung der Synchronlage erfolgen soll, um Master und Slave getrennt referenzieren zu können.

Index	202F _h		
Name	synchronisation_selector_data		
Type	RECORD		07 _h
Sub-Index	07 _h		
Name	synchronisation_main		
Info	--	rw	PBC UINT16
Value	siehe Tabelle	--	

Bit	Wert	Bedeutung
0	0001 _h	0: Synchronisation inaktiv 1: Synchronisation aktiv
1	0002 _h	0: "Fliegende Säge" inaktiv 1: "Fliegende Säge" aktiv
8	0100 _h	0: Synchronisation während der Referenzfahrt 1: Keine Synchronisation während der Referenzfahrt

3.11.2.5 Objekt 2023_h: synchronisation_filter_time

Über das Objekt `synchronisation_filter_time` wird die Filterzeitkonstante eines PT1- Filters festgelegt, mit dem die Synchronisationsdrehzahl geglättet wird. Dies kann insbesondere bei geringen Strichzahlen nötig sein, da hier bereits kleine Änderungen des Eingangswertes hohen Drehzahlen entsprechen. Andererseits ist der Antrieb bei hohen Filterzeiten ggf. nicht mehr in der Lage schnell genug einem dynamischen Eingangssignal zu folgen.

Index	2023 _h		
Name	synchronisation_filter_time		
Info	µs	rw	PBC UINT32
Value	10...50000	--	

3.12 Analoge Eingänge

3.12.1 Übersicht

Die Servoregler verfügen über analoge Eingänge, über die dem Servoregler beispielsweise Sollwerte vorgegeben werden können. Für alle diese analogen Eingänge bieten die nachfolgenden Objekte die Möglichkeit, die aktuelle Eingangsspannung auszulesen ([analog_input_voltage](#)) und einen Offset einzustellen ([analog_input_offset](#)). Je nach Servoregler-Reihe (ARS 2000 FS / BL 4000-C) sind unterschiedlich viele analoge Eingänge vorhanden.

3.12.2 Beschreibung der Objekte

3.12.2.1 Objekt 2400_h: analog_input_voltage (Eingangsspannung)

Die Objektgruppe [analog_input_voltage](#) liefert die aktuelle Eingangsspannung des jeweiligen Kanals unter Berücksichtigung des Offsets in Millivolt.

Index	2400_h			
Name	analog_input_voltage			
Type	ARRAY			03 _h
Sub-Index	01_h			
Name	analog_input_voltage_ch_0			
Info	mV	ro	PDO	INT16
Value	--	--		
Sub-Index	02_h			
Name	analog_input_voltage_ch_1			
Info	mV	ro	PDO	INT16
Value	--	--		
Sub-Index	03_h			
Name	analog_input_voltage_ch_2			
Info	mV	ro	PDO	INT16
Value	--	--		

3.12.2.2 Objekt 2401_h: analog_input_offset (Offset Analogeingänge)

Über die Objektgruppe `analog_input_offset` kann die Offsetspannung in Millivolt für die jeweiligen Eingänge gesetzt bzw. gelesen werden. Mit Hilfe des Offsets kann eine eventuelle anliegende Gleichspannung ausgeglichen werden. Ein positiver Offset kompensiert dabei eine positive Eingangsspannung.

Index	2401_h		
Name	analog_input_offset		
Type	ARRAY		03 _h
Sub-Index	01_h		
Name	analog_input_offset_ch_0		
Info	mV	rw	PDO INT32
Value	-10000...10000	--	
Sub-Index	02_h		
Name	analog_input_offset_ch_1		
Info	mV	rw	PDO INT32
Value	-10000...10000	--	
Sub-Index	03_h		
Name	analog_input_offset_ch_2		
Info	mV	rw	PDO INT32
Value	-10000...10000	--	

3.13 Digitale Ein- und Ausgänge

3.13.1 Übersicht

Alle digitalen Eingänge des Servoreglers können über den CAN-Bus gelesen und fast alle digitalen Ausgänge können beliebig gesetzt werden. Zudem können den digitalen Ausgängen des Servoreglers Statusmeldungen zugeordnet werden. Beim ARS 2000 FS kann auch das optionale Technologiemodul EA88 derart parametrierung werden. Je nach Gerätefamilie sind möglicherweise nicht alle hier beschriebenen digitalen Ein-/Ausgänge bei jedem Gerät vorhanden.

3.13.2 Beschreibung der Objekte

3.13.2.1 Objekt 60FD_h: digital_inputs

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

Index	60FD _h			
Name	digital_inputs			
Info	--	ro	PDO	UINT32
Value	gemäß Tabelle		--	

Bit	Wert	digitaler Eingang
0	00000001 _h	Negativer Endschalter
1	00000002 _h	Positiver Endschalter
2	00000004 _h	Referenzschalter
3	00000008 _h	Interlock (Servoregler- oder Endstufenfreigabe oder STO fehlt)
16...23	00FF0000 _h	Gegebenenfalls zusätzliche digitale Eingänge eines EA88-Moduls (EA88-0)
24...27	0F000000 _h	DIN0...DIN3
28	10000000 _h	DIN8
29	20000000 _h	ARS 2000: DIN9, BL 4100-C: DIN4

3.13.2.2 Objekt 60FE_h: digital_outputs

Über das Objekt 60FE_h können die digitalen Ausgänge angesteuert werden. Ein gesetztes Bit im Objekt digital_outputs_mask gibt an, welcher digitale Ausgang angesteuert werden soll. Über das Objekt digital_outputs_data können die ausgewählten Ausgänge dann beliebig gesetzt werden. Es ist zu beachten, dass bei der Ansteuerung der digitalen Ausgänge eine Verzögerung von bis zu 10 ms auftreten kann. Wann die Ausgänge wirklich gesetzt werden, kann durch Zurücklesen des Objekts 60FE_h festgestellt werden.

Index	60FE_h			
Name	digital_outputs			
Type	ARRAY			02 _h
Sub-Index	01_h			
Name	digital_outputs_data			
Info	--	rw	PDO	UINT32
Value	--			
Sub-Index	02_h			
Name	digital_outputs_mask			
Info	--	rw	PDO	UINT32
Value	--			

Bit	Wert	Digitaler Ausgang
0	0000001 _h	1 = Bremse anziehen
16...23	00FF0000 _h	ggf. zusätzliche digitale Ausgänge eines EA88-Moduls (EA88-0)
25...27	0E000000 _h	DOUT1...DOUT3

ACHTUNG Sachschäden möglich

Wenn die Bremsansteuerung über `digital_output_mask` freigegeben ist, wird durch Löschen von Bit 0 in `digital_output_data` die Haltebremse manuell gelüftet!

Dies kann bei hängenden Achsen zu einem Absacken der Achse führen.

3.13.2.3 Objekt 2420_h: digital_output_state_mapping

Über die Objektgruppe `digital_outputs_state_mapping` können verschiedene Statusmeldungen des Servoreglers über die digitalen Ausgänge ausgegeben werden.

Für die integrierten digitalen Ausgänge des Servoreglers ist hierzu für jeden Ausgang ein eigener Subindex vorhanden. Für die optional verfügbaren Ausgänge eines EA88- Moduls im Technologieschacht 1 sind innerhalb eines Subindex immer 4 Ausgänge zusammengefasst. Somit ist für jeden Ausgang ein Byte vorhanden, in das die Funktionsnummer einzutragen ist.

Wenn einem digitalen Ausgang eine derartige Funktion zugeordnet wurde und der Ausgang dann direkt über `digital_outputs` (60FE_h) ein- oder ausgeschaltet wird, wird auch das Objekt `digital_outputs_state_mapping` auf AUS (0) bzw. EIN (12) gesetzt.

Index	2420_h		
Name	digital_outputs_state_mapping		
Type	RECORD		12 _h
Sub-Index	01_h		
Name	dig_out_state_mapp_dout_1		
Info	--	rw	PBO UINT8
Value	0...16, siehe Tabelle	--	
Sub-Index	02_h		
Name	dig_out_state_mapp_dout_2		
Info	--	rw	PBO UINT8
Value	0...16, siehe Tabelle	--	
Sub-Index	03_h		
Name	dig_out_state_mapp_dout_3		
Info	--	rw	PBO UINT8
Value	0...16, siehe Tabelle	--	

Wert	Bezeichnung	Wert	Bezeichnung
0	Aus (Ausgang ist Low)	9	Unterspannung Zwischenkreis
1	Position $X_{soll} = X_{ziel}$	10	Feststellbremse gelüftet
2	Position $X_{ist} = X_{ziel}$	11	Endstufe aktiv
3	Reserviert	12	Ein (Ausgang ist High)
4	Restweg	13	Reserviert
5	Referenzfahrt aktiv	14	Reserviert
6	Vergleichsdrehzahl erreicht	15	Linearmotor identifiziert
7	I ² t-Überwachung aktiv	16	Referenzposition gültig
8	Schleppfehler		

Sub-Index	11_h		
Name	dig_out_state_mapp_ea88_0_low		
Info	--	rw	PBO UINT32
Value		--	

Bit	Maske	Name	Bezeichnung
0 ... 7	000000FF _h	EA88_0_dout_0_mapping	Funktion für EA88 0 DOUT1
8 ... 15	0000FF00 _h	EA88_0_dout_1_mapping	Funktion für EA88 0 DOUT2
16 ... 23	00FF0000 _h	EA88_0_dout_2_mapping	Funktion für EA88 0 DOUT3
23 ... 31	FF000000 _h	EA88_0_dout_3_mapping	Funktion für EA88 0 DOUT4

Sub-Index	12 _h		
Name	dig_out_state_mapp_ea88_0_high		
Info	--	rw	DD UINT32
Value	--		

Bit	Maske	Name	Bezeichnung
0 ... 7	000000FF _h	EA88_0_dout_4_mapping	Funktion für EA88 0 DOUT5
8 ... 15	0000FF00 _h	EA88_0_dout_5_mapping	Funktion für EA88 0 DOUT6
16 ... 23	00FF0000 _h	EA88_0_dout_6_mapping	Funktion für EA88 0 DOUT7
23 ... 31	FF000000 _h	EA88_0_dout_7_mapping	Funktion für EA88 0 DOUT8

3.14 Endschalter / Referenzschalter

3.14.1 Übersicht

Für die Definition der Referenzposition des Servoreglers können wahlweise Endschalter (limit switch) oder Referenzschalter (homing switch) verwendet werden. Nähere Informationen zu den möglichen Referenzfahrt-Methoden finden sie im Abschnitt 5.2 *Betriebsart Referenzfahrt (Homing Mode)* auf Seite 135.

3.14.2 Beschreibung der Objekte

3.14.2.1 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.

Index	6510 _h		
Name	drive_data		
Type	RECORD		F0 _h
Sub-Index	11 _h		
Name	limit_switch_polarity		
Info	--	rw	DD INT16
Value	0, 1	1	

Wert	Bedeutung
0	Öffner
1	Schließer

3.14.2.2 Objekt 6510_h_12_h: limit_switch_selector

Über das Objekt 6510_h_12_h (limit_switch_selector) kann die Zuordnung der Endschalter (negativ, positiv) vertauscht werden, ohne Änderungen an der Verkabelung vornehmen zu müssen. Um die Zuordnung der Endschalter zu tauschen, ist eine Eins einzutragen.

Index	6510 _h		
Name	drive_data		
Type	RECORD		F0 _h
Sub-Index	12 _h		
Name	limit_switch_selector		
Info	--	rw	PBC INT16
Value	0, 1	0	

Wert	Bedeutung
0	DIN6 = E0 (Endschalter negativ) DIN7 = E1 (Endschalter positiv)
1	DIN6 = E1 (Endschalter positiv) DIN7 = E0 (Endschalter negativ)

3.14.2.3 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).

Index	6510 _h		
Name	drive_data		
Type	RECORD		F0 _h
Sub-Index	15 _h		
Name	limit_switch_deceleration		
Info	acceleration_unit	rw	PBC INT32
Value	0...3000000 min ⁻¹ /s	--	

3.14.2.4 Objekt 6510_h_14_h: homing_switch_polarity

Die Polarität des Referenzschalters kann durch das Objekt 6510_h_14_h (`homing_switch_polarity`) programmiert werden. Für einen öffnenden Referenzschalter ist in dieses Objekt eine Null, bei der Verwendung von schließenden Kontakten ist eine Eins einzutragen.

Index	6510 _h		
Name	drive_data		
Type	RECORD	F0 _h	
Sub-Index	14 _h		
Name	homing_switch_polarity		
Info	--	rw	PBC INT16
Value	0, 1	1	

Wert	Bedeutung
0	Öffner
1	Schließer

3.14.2.5 Objekt 6510_h_13_h: homing_switch_selector

Das Objekt 6510_h_13_h (`homing_switch_selector`) legt fest, ob DIN8 oder DIN9 als Referenzschalter verwendet werden soll.

Index	6510 _h		
Name	drive_data		
Type	RECORD	F0 _h	
Sub-Index	13 _h		
Name	homing_switch_selector		
Info	--	rw	PBC INT16
Value	0, 1	0	

Wert	Bedeutung
0	DIN9
1	DIN8

3.15 Erfassen von Positionen

3.15.1 Übersicht

Die Servoregler bieten die Möglichkeit, den Lageistwert auf der steigenden oder fallenden Flanke eines digitalen Eingangs (z.B. eines Messtasters) hin abzuspeichern (Sampling). Dieser Lagewert kann dann z.B. zur Berechnung innerhalb einer Steuerung ausgelesen werden.

Alle notwendigen Objekte sind in dem Record `sample_data` zusammengefasst: Das Objekt `sample_mode` legt die Art des Samplings fest: Soll nur ein einmaliges Sample-Ereignis aufgezeichnet werden oder soll kontinuierlich gesampelt werden. Über das Objekt `sample_status` kann die Steuerung abfragen, ob ein Sample-Ereignis aufgetreten ist. Dies wird durch ein gesetztes Bit signalisiert, welches ebenfalls im `statusword` angezeigt werden kann, wenn das Objekt `sample_status_mask` entsprechend gesetzt ist.

Das Objekt `sample_control` dient dazu, die Freigabe des Sample-Ereignisses zu steuern und letztlich können über die Objekte `sample_position_rising_edge` und `sample_position_falling_edge` die gesampelten Positionen ausgelesen werden.

Welcher digitale Eingang verwendet wird, lässt sich mit dem Metronix ServoCommander® unter **Parameter / IOs / Digitale Eingänge / Sample- Eingang** festlegen.

3.15.2 Beschreibung der Objekte

3.15.2.1 Objekt 204A_h: `sample_data`

Index	204A_h		
Name	sample_data		
Type	RECORD		06 _h

Mit dem folgenden Objekt kann gewählt werden, ob auf jedes Auftreten eines Sample-Events die Position bestimmt werden soll (Kontinuierliches Sampling) oder ob das Sampling nach einem Sample-Ereignis gesperrt werden soll, bis das Sampling erneut freigegeben wird. Beachten Sie hierbei, dass auch bereits ein Prellen beide Flanken auslösen kann.

Sub-Index	01_h		
Name	sample_mode		
Info	--	rw	PBC UINT16
Value	0...1	--	

Wert	Bezeichnung
0	Kontinuierliches Sampling
1	Autolock sampling

Das folgenden Objekt zeigt ein neues Sample- Ereignis an.

Sub-Index	02_h		
Name	sample_status		
Info	--	ro	PDO UINT8
Value	0...3	--	

Bit	Wert	Name	Beschreibung
0	01 _h	falling_edge_occurred	= 1: Neue Sample-Position (fallende Flanke)
1	02 _h	rising_edge_occurred	= 1: Neue Sample-Position (steigende Flanke)

Mit dem folgenden Objekt können die Bits des Objekts **sample_status** festgelegt werden, die auch zum Setzen von Bit 15 des **statusword** führen sollen. Dadurch ist im üblicherweise ohnehin zu übertragenden **statusword** die Information "Sample- Ereignis aufgetreten" vorhanden, so dass die Steuerung nur in diesem Fall das Objekt **sample_status** lesen muss, um ggf. festzustellen welche Flanke aufgetreten ist.

Sub-Index	03_h		
Name	sample_status_mask		
Info	--	rw	PDO UINT8
Value	0...3	--	

Bit	Wert	Name	Beschreibung
0	01 _h	falling_edge_visible	Wenn falling_edge_occured = 1 wird im statusword Bit 15 gesetzt
1	02 _h	rising_edge_visible	Wenn rising_edge_occured = 1 wird im statusword Bit 15 gesetzt

Das Setzen des jeweiligen Bits in **sample_control** setzt zum einen das entsprechende Statusbit in **sample_status** zurück und schaltet im Falle des "Autolock"- Samplings das Sampling wieder frei.

Sub-Index	04_h		
Name	sample_control		
Info	--	wo	PDO UINT8
Value	0...3	0	

Bit	Wert	Name	Beschreibung
0	01 _h	falling_edge_enable	Sampling bei fallender Flanke
1	02 _h	rising_edge_enable	Sampling bei steigender Flanke

Die folgenden Objekte enthalten die gesampelten Positionen.

Sub-Index	05_h			
Name	sample_position_rising_edge			
Info	position_unit	ro	PDO	INT32
Value	--	--		
Sub-Index	06_h			
Name	sample_position_falling_edge			
Info	position_unit	ro	PDO	INT32
Value	--	--		

3.16 Bremsen-Ansteuerung

3.16.1 Übersicht

Mittels der nachfolgenden Objekte kann parametrierung werden, wie der Servoregler eine eventuell im Motor integrierte Haltebremse ansteuert. Die Haltebremse wird immer freigeschaltet, sobald die Servoreglerfreigabe eingeschaltet wird. Für Haltebremsen mit hoher mechanischer Trägheit kann eine Verzögerungszeit t_A parametrierung werden, damit die Haltebremse in Eingriff ist, bevor die Endstufe ausgeschaltet wird (Durchsacken vertikaler Achsen). Ebenso wird die Ansteuerung des Motors verzögert (t_F), bis die Haltebremse vollständig gelöst ist. Beide Verzögerungen werden gleichzeitig durch das Objekt `brake_delay_time` ($t_A = t_F$) parametrierung.

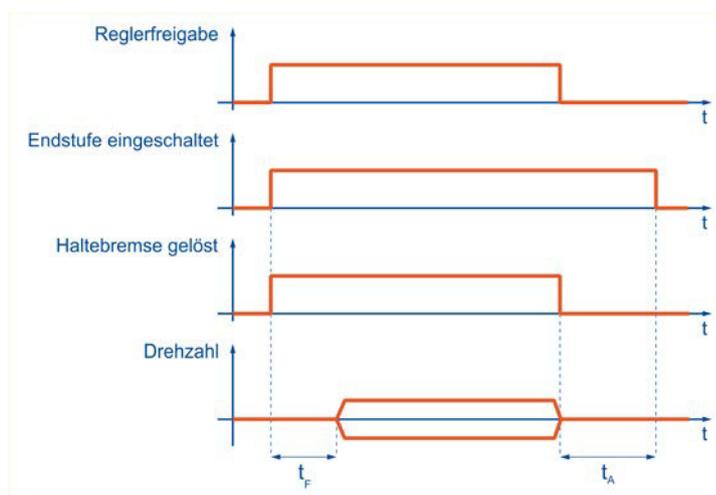


Abbildung 6: Funktion der Bremsverzögerung (bei Drehzahlregelung / Positionieren)

3.16.2 Beschreibung der Objekte

3.16.2.1 Objekt 6510_h_18_h: brake_delay_time

Über das Objekt `brake_delay_time` kann die Bremsverzögerungszeit parametrierung werden.

Index	6510_h			
Name	drive_data			
Type	RECORD			F0 _h
Sub-Index	18_h			
Name	brake_delay_time			
Info	ms	rw	DDC	UINT16
Value	0...32000	--		

3.17 Geräteinformationen

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

3.17.1 Beschreibung der Objekte

3.17.1.1 Objekt 1000_h: device_type

Das Objekt `device_type` zeigt in den unteren 16 Bit an, dass das Geräteprofil 402 unterstützt wird und in den oberen 16 Bit, dass es sich um ein Servo drive (Bit 17) handelt.

Index	1000 _h		
Name	device_type		
Info	--	ro	DDC UINT32
Value	--		00020192 _h

3.17.1.2 Objekt 1008_h: manufacturer_device_name

Über das Objekt `manufacturer_device_name` kann der Name der Gerätefamilie in Klartext ausgelesen werden.

Index	1008 _h		
Name	manufacturer_device_name		
Info	--	ro	DDC VISSTR
Value	--		--

3.17.1.3 Objekt 1009_h: manufacturer_hardware_version

Über das Objekt `manufacturer_hardware_version` kann die Hardware-Revision des Geräts ausgelesen werden. Diese wird auch im Metronix ServoCommander® unter [Hilfe / Info](#) Registerkarte [Firmware / Hardware](#) angezeigt.

Index	1009 _h		
Name	manufacturer_hardware_version		
Info	MMM.SSS	ro	DDC VISSTR
Value	--		--

Wert	Bedeutung
M	main version
S	sub version

3.17.1.4 Objekt 100A_h: manufacturer_software_version

Über das Objekt `manufacturer_software_version` kann die Firmwareversion in Klartext ausgelesen werden. Die einzelnen Teile der Versionsnummer sind als ASCII-Zeichen ohne führende Nullen formatiert und sind durch Punkte getrennt, z. B. "1.0.0.1.2".

Index	100A _h		
Name	manufacturer_software_version		
Info	M.S.C.K.k	ro	PRO VISSTR
Value	--		

Wert	Bedeutung
M	Entspricht MMMM von Objekt 6510h_A9h: firmware_main_version
S	Entspricht SSSS von Objekt 6510h_A9h: firmware_main_version
C	Entspricht Objekt 6510h_AAh: firmware_custom_version
K	Entspricht MMMM von Objekt 6510h_ADh: km_release
k	Entspricht SSSS von Objekt 6510h_ADh: km_release

3.17.1.5 Objekt 1018_h: identity_object

Über das in der DS301 festgelegte `identity_object` kann der Servoregler 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 (`serial_number`) ausgelesen werden.

Index	1018 _h		
Name	identity_object		
Type	RECORD		04 _h
Sub-Index	01 _h		
Name	vendor_id		
Info	--	ro	PRO UINT32
Value	--	000000E4 _h	
Sub-Index	02 _h		
Name	product_code		
Info	--	ro	PRO UINT32
Value	--		

Wert	Bedeutung	Wert	Bedeutung
2005 _h	ARS 2102	2086 _h	ARS 2105 SE
2006 _h	ARS 2105	208B _h	ARS 2310 SE
2009 _h	ARS 2302	2090 _h	ARS 2108 SE

Wert	Bedeutung	Wert	Bedeutung
200A _h	ARS 2305	2089 _h	ARS 2302 SE
200B _h	ARS 2310	208A _h	ARS 2305 SE
200C _h	ARS 2320	8202 _h	BL 4102-C
2008 _h	ARS 2320W	8203 _h	BL 4104-C
200D _h	ARS 2340	8208 _h	BL 4304-C
200E _h	ARS 2360W	8209 _h	BL 4308-C
2045 _h	ARS 2102 FS	8212 _h	BL 4312-C
2046 _h	ARS 2105 FS	820A _h	BL 4104-M ETH
2050 _h	ARS 2108 FS	820C _h	BL 4104-D ETH
2049 _h	ARS 2302 FS	820D _h	BL 4840-M ETH
204A _h	ARS 2305 FS	820F _h	BL 4840-D ETH
204B _h	ARS 2310 FS	820B _h	BL 4104-M CAN
204C _h	ARS 2320 FS	8210 _h	BL 4104-D CAN
204D _h	ARS 2340 FS	820E _h	BL 4840-M CAN
204E _h	ARS 2360W FS	8211 _h	BL 4840-D CAN
2085 _h	ARS 2102 SE		

Sub-Index	03_h		
Name	revision_number		
Info	--	ro	PDO UINT32
Value	--	00040002 _h	
Sub-Index	04_h		
Name	serial_number		
Info	--	ro	PDO UINT32
Value	--	--	

3.17.1.6 Objekt 6510_h_A0_h: drive_serial_number

Über das Objekt `drive_serial_number` kann die Seriennummer des Servoreglers ausgelesen werden. Dieses Objekt dient der Kompatibilität zu früheren Versionen.

Index	6510_h		
Name	drive_data		
Type	RECORD		F0 _h
Sub-Index	A0_h		
Name	drive_serial_number		
Info	--	ro	PDO UINT32
Value	--	--	

3.17.1.7 Objekt 6510_h_A1_h: drive_type

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

Index	6510 _h		
Name	drive_data		
Type	RECORD		F0 _h
Sub-Index	A1 _h		
Name	drive_type		
Info	siehe 1018 _h _02 _h (product code)	ro	PBC UINT32
Value	siehe 1018 _h _02 _h (product code)	--	

3.17.1.8 Objekt 6510_h_A9_h: firmware_main_version

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

Index	6510 _h		
Name	drive_data		
Type	RECORD		F0 _h
Sub-Index	A9 _h		
Name	firmware_main_version		
Info	MMMMSSSS _h	ro	PBC UINT32
Value	--	--	

Wert	Bedeutung
M	main version
S	sub version

3.17.1.9 Objekt 6510_h_AA_h: firmware_custom_version

Über das Objekt `firmware_custom_version` kann die Versionsnummer der kundenspezifischen Variante der Firmware ausgelesen werden.

Index	6510 _h		
Name	drive_data		
Type	RECORD	F0 _h	
Sub-Index	AA _h		
Name	firmware_custom_version		
Info	--	ro	DDC UINT32
Value	--	--	

3.17.1.10 Objekt 6510_h_AD_h: km_release

Über die Versionsnummer des `km_release` können Firmwarestände der gleichen Produktstufe unterschieden werden.

Index	6510 _h		
Name	drive_data		
Type	RECORD	F0 _h	
Sub-Index	AD _h		
Name	km_release		
Info	MMMMSSSS _h	ro	DDC UINT32
Value	--	--	

Wert	Bedeutung
M	main version
S	sub version

3.17.1.11 Objekt 6510_h_AC_h: firmware_type

Über das Objekt `firmware_type` kann ausgelesen werden, für welche Gerätefamilie und für welchen Winkelgeber typ die geladene Firmware geeignet ist. Da seit der ARS 2000-Familie das Winkelgeber-Interface nicht mehr steckbar ist, sind im Parameter G grundsätzlich alle Bits gesetzt (F_h).

Index	6510_h		
Name	drive_data		
Type	RECORD	F0 _h	
Sub-Index	AC_h		
Name	firmware_type		
Info	00000GX _h	ro	PBO UINT32
Value	F2 _h	--	

Wert (X)	Bedeutung
0 _h	IMD-F
1 _h	ARS
2 _h	ARS 2000, ARS 2000 FS, ARS 2000 SE und BL 4000-C

3.17.1.12 Objekt 6510_h_B0_h: cycletime_current_controller

Über das Objekt `cycletime_current_controller` kann die Zykluszeit des Stromreglers in Mikrosekunden ausgelesen werden.

Index	6510_h		
Name	drive_data		
Type	RECORD	F0 _h	
Sub-Index	B0_h		
Name	cycletime_current_controller		
Info	µs	ro	PBO UINT32
Value	--	--	

3.17.1.13 Objekt 6510_h_B1_h: cycletime_velocity_controller

Über das Objekt `cycletime_velocity_controller` kann die Zykluszeit des Drehzahlreglers in Mikrosekunden ausgelesen werden.

Index	6510_h		
Name	drive_data		
Type	RECORD		F0 _h
Sub-Index	B1_h		
Name	cycletime_velocity_controller		
Info	µs	ro	PBC UINT32
Value	--	--	

3.17.1.14 Objekt 6510_h_B2_h: cycletime_position_controller

Über das Objekt `cycletime_position_controller` kann die Zykluszeit des Lagereglers in Mikrosekunden ausgelesen werden.

Index	6510_h		
Name	drive_data		
Type	RECORD		F0 _h
Sub-Index	B2_h		
Name	cycletime_position_controller		
Info	µs	ro	PBC UINT32
Value	--	--	

3.17.1.15 Objekt 6510_h_B3_h: cycletime_trajectory_generator

Über das Objekt `cycletime_trajectory_generator` kann die Zykluszeit der Positioniersteuerung in Mikrosekunden ausgelesen werden.

Index	6510_h		
Name	drive_data		
Type	RECORD		F0 _h
Sub-Index	B3_h		
Name	cycletime_trajectory_generator		
Info	µs	ro	PBC UINT32
Value	--	--	

3.17.1.16 Objekt 6510_h_C0_h: commissioning_state

ACHTUNG Ungeeignete Parametrierung möglich

Dieses Objekt enthält keinerlei Informationen darüber, ob der Servoregler dem Motor und der Applikation entsprechend richtig parametriert wurde, sondern nur, ob die genannten Punkte nach der Auslieferung mindestens einmal überhaupt parametriert wurden.

HINWEIS „A“ im 7-Segment-Display

Beachten Sie, dass mindestens ein Bit im Objekt `commissioning_state` gesetzt werden muss, um das „A“ auf dem Displays Ihres Servoreglers zu unterdrücken.

Index	6510 _h		
Name	drive_data		
Type	RECORD		F0 _h
Sub-Index	C0 _h		
Name	commissioning_state		
Info	--	rw	DD UINT32
Value	--	--	

Bit	Bedeutung	Bit	Bedeutung
0	Nennstrom gültig	9	Reserviert
1	Maximalstrom gültig	10	Physik. Einheiten gültig
2	Polzahl des Motors gültig	11	Drehzahlregler gültig
3	Offsetwinkel / Drehsinn gültig	12	Lageregler gültig
4	Reserviert	13	Sicherheitsparameter gültig
5	Offsetwinkel / Drehsinn Hallgeber gültig	14	Reserviert
6	Reserviert	15	Endschalter-Polarität gültig
7	Absolutlage Gebersystem gültig	16..31	Reserviert
8	Stromregler-Parameter gültig		

3.17.1.17 Objekt 20FD_h: user_device_name

Über das Objekt `user_device_name` kann der frei parametrierbare Name des Antriebs (z.B. "X-Achse") gelesen und geschrieben werden.

Index	20FD _h		
Name	user_device_name		
Info	--	rw	DD VISSTR
Value	--	--	

3.18 Fehlermanagement

3.18.1 Übersicht

Die Servoregler bieten die Möglichkeit, die Fehlerreaktion einzelner Ereignisse, wie z.B. das Auftreten eines Schleppfehlers, zu ändern. Dadurch reagiert der Servoregler unterschiedlich, wenn ein bestimmtes Ereignis eintritt: So kann je nach Einstellung heruntergebremst werden, die Endstufe sofort ausgeschaltet werden oder lediglich eine Warnung auf dem Display angezeigt werden.

Für jedes Ereignis ist herstellerseitig eine Mindestreaktion vorgesehen, die nicht unterschritten werden kann. So lassen sich „kritische“ Fehler wie beispielsweise 06-0 Kurzschluss Endstufe nicht umparametrieren, da hier eine sofortige Abschaltung notwendig ist, um den Servoregler vor einer eventuellen Zerstörung zu schützen.

Wird eine niedrigere Fehlerreaktion als für den jeweiligen Fehler zulässig eingetragen, wird der Wert auf die niedrigst zulässige Fehlerreaktion begrenzt. Eine Liste aller Fehlernummern befindet sich im Softwarehandbuch "Servoregler ARS 2000" bzw. dem Produkthandbuch BL 4000-C.

3.18.2 Beschreibung der Objekte

3.18.2.1 Objekt 2100_h: error_management

Index	2100 _h		
Name	error_management		
Type	RECORD		02 _h

Im Objekt `error_number` muss die Hauptfehlernummer angegeben werden, deren Reaktion geändert werden soll. Die Hauptfehlernummer ist in der Regel vor dem Bindestrich angegeben (z.B. Fehler 08-2, Hauptfehlernummer 8).

Sub-Index	01 _h		
Name	error_number		
Info	--	rw	DD UINT8
Value	1...96	--	

Im Objekt `error_reaction_code` kann die Reaktion des Fehlers verändert werden. Wird die herstellerseitige Mindestreaktion unterschritten, wird auf diese begrenzt. Die wirklich eingestellte Reaktion kann durch Rücklesen bestimmt werden.

Sub-Index	02_h		
Name	error_reaction_code		
Info	--	rw	PDO UINT8
Value	0, 1, 3, 5, 7, 8	--	

Wert	Bedeutung
0	Keine Aktion
1	Eintrag im Puffer
3	Warnung auf dem 7-Segment-Display
5	Servoreglerfreigabe aus
7	Bremsen mit Maximalstrom
8	Endstufe aus

3.18.2.2 Objekt 200F_h: last_warning_code

Warnungen sind bemerkenswerte Ereignisse des Antriebs (z.B. ein Schleppfehler), die im Gegensatz zu einem Fehler nicht zum Stillsetzen des Antriebs führen sollen. Warnungen werden auf der 7-Segmentanzeige des Servoreglers angezeigt und danach automatisch vom Servoregler zurückgesetzt.

Die letzte aufgetretene Warnung kann über das folgende Objekt ausgelesen werden: Dabei zeigt Bit 15 an, ob die Warnung aktuell noch aktiv ist.

Index	200F_h		
Name	last_warning_code		
Info	--	ro	PDO UINT16
Value	--	--	

Bit	Wert	Beschreibung
0... 3	000F _h	Unternummer der Warnung
4... 11	0FF0 _h	Hauptnummer der Warnung
15	8000 _h	Warnung ist aktiv

4 Gerätesteuerung (Device Control)

4.1 Übersicht

Das nachfolgende Kapitel beschreibt, wie der Servoregler unter CANopen gesteuert wird, also wie beispielsweise die Endstufe eingeschaltet oder ein Fehler quittiert wird.

Unter CANopen wird die gesamte Steuerung des Servoreglers über zwei Objekte realisiert: Über das **controlword** kann der Host den Servoregler steuern, während der Status des Servoreglers im Objekt **statusword** zurückgelesen werden kann. Zur Erklärung der Servoreglersteuerung werden die folgenden Begriffe verwandt:

Begriff	Erklärung
Zustand: (State)	Je nachdem ob beispielsweise die Endstufe eingeschaltet oder ein Fehler aufgetreten ist befindet sich der Servoregler in verschiedenen Zuständen. Die unter CANopen definierten Zustände werden im Laufe des Kapitels vorgestellt. Beispiel: OPERATION_ENABLE
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 Servoregler, 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.

4.2 Das Zustandsdiagramm des Servoreglers (State Machine)

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.

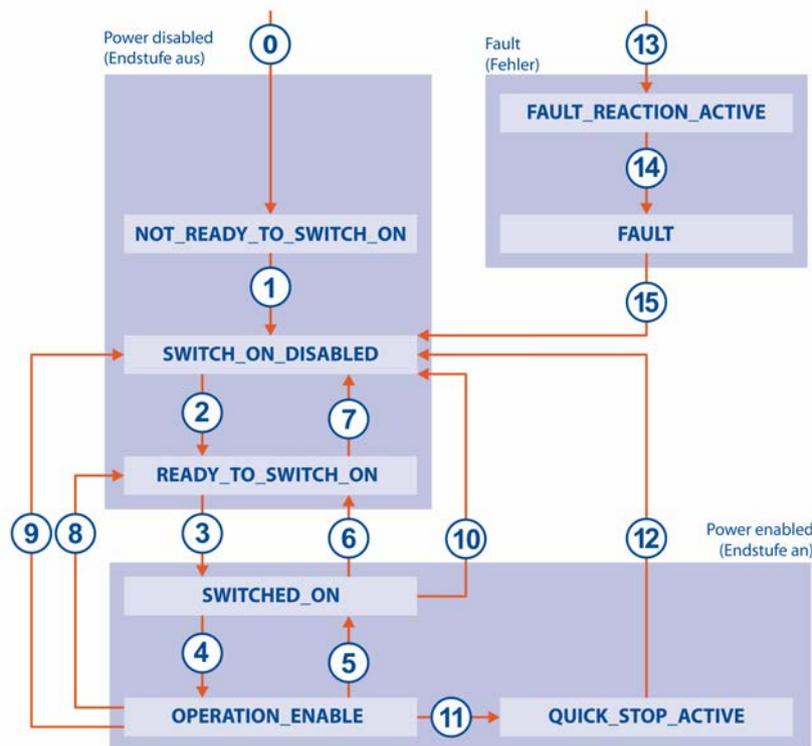


Abbildung 7: Zustandsdiagramm des Servoreglers

Nach dem Einschalten initialisiert sich der Servoregler und erreicht schließlich den Zustand SWITCH_ON_DISABLED. In diesem Zustand ist die CAN-Kommunikation voll funktionsfähig und der Servoregler kann parametrieren (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-Servoreglerfreigabe 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 parametrieren ist und ein entsprechender Sollwert gleich Null ist. Der Zustandsübergang 9 entspricht der Wegnahme der Freigabe, d.h. ein noch laufender Motor würde unregelmäßig austrudeln. 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.

› Wichtige Zustandsübergänge

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	Endstufen- u. Reglerfreigabe vorhanden + Shutdown	Shutdown	x	1	1	0	Keine
3	Switch On	Switch On	x	1	1	1	Einschalten der Endstufe
4	Enable Operation	Enable Operation	1	1	1	1	Regelung gemäß eingestellter Betriebsart
9	Disable Voltage	Disable Voltage	x	x	0	x	Endstufe wird gesperrt. Motor ist frei drehbar.
15	Fehler behoben + Fault Reset	Fault Reset	Bit 7 = 				Fehler quittieren

BEISPIEL

Nachdem der Servoregler parametrierung wurde, soll der Regler freigegeben, d.h. die Endstufe eingeschaltet, werden:

1. Der Servoregler ist im Zustand SWITCH_ON_DISABLED
2. Der Regler soll in den Zustand OPERATION_ENABLED
3. Es sind die Zustandsübergänge 2, 3 und 4 auszuführen.
4. Aus der vorhergehenden Tabelle folgt:

Übergang	controlword	Neuer Zustand
2	0006 _h	READY_TO_SWITCH_ON
3	0007 _h	SWITCHED_ON
4	000F _h	OPERATION_ENABLE

Hinweise:

- Um das Prinzip zu verdeutlichen, sind keine weiteren Bits im controlword gesetzt.
- Die Übergänge 3 und 4 können zusammengefasst werden, indem gleich 000F_h geschrieben wird, da das gesetzte Bit 3 für den Übergang 3 nicht relevant ist.
- Es muss jeweils abgewartet werden, bis der Regler den Zustand eingenommen hat. Dies wird im folgenden Abschnitt noch näher erläutert.

4.2.1 Zustandsdiagramm: Zustände

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

Name	Bedeutung
NOT_READY_TO_SWITCH_ON	Der Servoregler führt einen Selbsttest durch. Die CAN-Kommunikation arbeitet noch nicht.
SWITCH_ON_DISABLED	Der Servoregler hat seinen Selbsttest abgeschlossen. CAN-Kommunikation ist möglich.
READY_TO_SWITCH_ON	Der Servoregler wartet bis die digitalen Eingänge „Endstufen-“ und „Servoreglerfreigabe“ an 24 V liegen. (Reglerfreigabelogik „Digitaler Eingang und CAN“).
SWITCHED_ON * ¹⁾	Die Endstufe ist eingeschaltet.
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.

4.2.2 Zustandsdiagramm: Zustandsübergänge

⚠ GEFAHR Lebensgefährliche elektrische Spannung! ⚠

Endstufe gesperrt bedeutet, dass die Leistungshalbleiter 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.

Das Signal garantiert nicht, dass der Motor wirklich spannungsfrei ist

⚠ VORSICHT Verletzungsgefahr

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, Resolver-Offsetwinkel etc.) kann es zu einem unkontrollierten Verhalten des Antriebes kommen.

In der folgenden Tabelle sind alle Zustandsübergänge 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	Endstufen- und Reglerfreigabe vorhanden + Shutdown	Shutdown	x	1	1	0	-
3	Switch On	Switch On	x	1	1	1	Einschalten der Endstufe
4	Enable Operation	Enable Operation	1	1	1	1	Regelung gemäß eingestellter Betriebsart
5	Disable Operation	Disable Operation	0	1	1	1	Endstufe wird gesperrt. Motor ist frei drehbar
6	Shutdown	Shutdown	x	1	1	0	Endstufe wird gesperrt. Motor ist frei drehbar
7	Quick Stop	Quick Stop	x	0	1	x	-

Nr.	Wird durchgeführt wenn	Bitkombination (controlword)				Aktion	
		Bit	3	2	1		0
8	Shutdown	Shutdown	x	1	1	0	Endstufe wird gesperrt. Motor ist frei drehbar
9	Disable Voltage	Disable Voltage	x	x	0	x	Endstufe wird gesperrt. Motor ist frei drehbar.
10	Disable Voltage	Disable Voltage	x	x	0	x	Endstufe wird gesperrt. Motor ist frei drehbar
11	Quick Stop	Quick Stop	x	0	1	x	Es wird eine Bremsung gemäß quick_stop_option_code eingeleitet.
12	Bremsung beendet oder 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)	

4.3 controlword (Steuerwort)

Objekt 6040_h: controlword

Mit dem **controlword** kann der aktuelle Zustand des Servoreglers 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 Servoreglers ab, die nach diesem Kapitel erläutert wird.

Index	6040 _h			
Name	controlword			
Info	--	rw	PDO	UINT16
Value	--			

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	Reserviert, mit 0 beschreiben
10	0400 _h	Reserviert, mit 0 beschreiben
11	0800 _h	Reserviert, mit 0 beschreiben
12	1000 _h	Reserviert, mit 0 beschreiben
13	2000 _h	Reserviert, mit 0 beschreiben
14	4000 _h	Reserviert, mit 0 beschreiben
15	8000 _h	Reserviert, mit 0 beschreiben

› Beschreibung der Kommandos (Bits 0...3, Bit 7)

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	x	x	1	1	0
Switch On	x	x	1	1	1
Disable Voltage	x	x	x	0	x
Quick Stop	x	x	0	1	x
Disable Operation	x	0	1	1	1
Enable Operation	x	1	1	1	1
Fault Reset		x	x	x	x

HINWEIS Statusänderungen

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.

› Beschreibung der weiteren Bits

Nachfolgend sind die restlichen Bits des **controlwords** erläutert. Einige Bits haben dabei je nach Betriebsart (**modes_of_operation**), d.h. ob der Servoregler 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 Servoregler, dass ein neuer Fahrauftrag übernommen werden soll. Siehe dazu unbedingt auch Abschnitt 5.3 <i>Betriebsart Positionieren (Profile Position Mode)</i> auf Seite 145.
start_homing_operation	Im Homing Mode : Eine steigende Flanke bewirkt, dass die parametrisierte 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 den Abschnitt 5.4 <i>Interpolated Position Mode</i> auf Seite 150

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 hierzu unbedingt auch den Abschnitt 5.3 <i>Betriebsart Positionieren (Profile Position Mode)</i> auf Seite 145.
Bit 6	
relative	Nur im Profile Position Mode : Bei gesetztem Bit bezieht der Servoregler 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 Servoregler 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 Servoregler wieder beschleunigt.
halt	Im Profile Torque Mode : Bei gesetztem Bit wird das Drehmoment auf Null abgesenkt. Dies geschieht mit der torque_slope . Das Löschen des Bits bewirkt, dass der Servoregler wieder beschleunigt.
halt	Im Homing Mode : Bei gesetztem Bit wird die laufende Referenzfahrt abgebrochen. Das Löschen des Bits hat keine Auswirkung.

4.4 Auslesen des Servoreglerzustands

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 Servoregler 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	x	0	0	0	0	004F _h	0000 _h
Switch_On_Disabled	1	x	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
Quick_Stop_Active	0	0	0	1	1	1	006F _h	0007 _h
Fault_Reaction_Active	0	x	1	1	1	1	004F _h	000F _h
Fault	0	x	1	1	1	1	004F _h	000F _h
Fault (gemäß DS402) ¹⁾	0	x	1	0	0	0	004F _h	0008 _h

HINWEIS Zustand FAULT wird nicht gemäß DS 402 zurückgemeldet

In bisherigen CANopen-Implementierungen wird der Zustand FAULT nicht gemäß DS 402 zurückgemeldet. Daher besteht die Möglichkeit über das Objekt **compatibility_control** (siehe Abschnitt 3.2 *Kompatibilitäts-Einstellungen* auf Seite 45) die Rückmeldung gemäß DS402 auszuwählen. Für Kompatibilität zu früheren Firmwareversionen brauchen keine Änderungen durchgeführt werden.

BEISPIEL

Das Beispiel auf Seite 114 zeigt, welche Bits im **controlword** gesetzt werden müssen, um den Regler freizugeben. In diesem Beispiel soll erläutert werden, wie daraufhin der aktuelle Status des Reglers aus dem **statusword** ausgelesen wird.

Übergang	controlword	Neuer Zustand	Warten bis
2	0006 _h	READY_TO_SWITCH_ON	(statusword & 006F _h) = 0021 _h
3+4	000F _h	OPERATION_ENABLE	(statusword & 006F _h) = 0027 _h

Hinweise:

- Um das Prinzip zu verdeutlichen, sind keine weiteren Bits im **controlword** gesetzt.
- Zur eindeutigen Bestimmung des Reglerzustands müssen auch nicht gesetzte Bits im **statusword** abgefragt werden. Daher muss das **statusword** entsprechend maskiert werden.

4.5 statuswords (Statusworte)

4.5.1 Objekt 6041_h: statusword

Index	6041 _h			
Name	statusword			
Info	--	ro	PDO	UINT16
Value	--			--

Bit	Wert	Name
0	0001 _h	Zustand des Servoreglers, siehe Abschnitt 4.4 <i>Auslesen des Servoreglerzustands</i> auf Seite 121. Diese Bits müssen gemeinsam ausgewertet werden
1	0002 _h	
2	0004 _h	
3	0008 _h	
5	0020 _h	
6	0040 _h	
4	0010 _h	
7	0080 _h	warning
8	0100 _h	drive_is_moving
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	manufacturer_statusbit
15	8000 _h	trigger_result

Alle Bits des **statusword** sind nicht gepuffert. Sie repräsentieren den aktuellen Gerätestatus.

Neben dem Servoreglerstatus 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	<p>Dieses Bit ist gesetzt, wenn die Endstufentransistoren ausgeschaltet sind.</p> <p>In bisherigen CANopen- Implementierungen wird Bit 4 (voltage_enabled) im Gegensatz zur Spezifikation in der DS 402 invertiert zurückgemeldet. Daher besteht die Möglichkeit über das Objekt compatibility_control (siehe Abschnitt 3.2 <i>Kompatibilitäts- Einstellungen</i> auf Seite 45) die Rückmeldung gemäß DS402 auszuwählen.</p> <p>Wenn im Objekt 6510h_F0h (compatibility_control) Bit 7 gesetzt ist, gilt: Dieses Bit ist gesetzt, wenn die Endstufentransistoren eingeschaltet sind. Für Kompatibilität zu früheren Firmwareversionen brauchen keine Änderungen durchgeführt werden.</p>
Bit 5	
quick_stop	Bei gelöschtem Bit führt der Antrieb einen Quick Stop gemäß quick_stop_option_code aus.
Bit 7	
warning	Die Bedeutung dieses Bits ist konfigurierbar: Es kann gesetzt werden, wenn ein beliebiges Bit in manufacturer_warnings_1 gesetzt wird. Siehe hierzu auch den Abschnitt 4.5.5 <i>Objekt 2001h: manufacturer_warnings</i> auf Seite 129.
Bit 8	herstellerspezifisch
drive_is_moving	Dieses Bit wird – unabhängig von modes_of_operation – gesetzt, wenn sich die aktuelle Ist-Drehzahl (velocity_actual_value) des Antriebes außerhalb des zugehörigen Toleranzfenster befindet (velocity_threshold).
Bit 9	
remote	Dieses Bit zeigt an, dass die Endstufe des Servoreglers über das CAN-Netzwerk freigegeben werden kann. Es ist gesetzt, wenn die Reglerfreigabelogik über das Objekt enable_logic entsprechend eingestellt ist.

Bit 10	Abhangig von <i>modes_of_operation</i> :
target_reached	<p>Im Profile Position Mode:</p> <p>Das Bit wird gesetzt, wenn die aktuelle Zielposition erreicht ist und sich die aktuelle Position (position_actual_value) im parametrisierten Positionsfenster (position_window) befindet. Auerdem wird es gesetzt, wenn der Antrieb bei gesetztem Halt-Bit zum Stillstand kommt. Es wird gelscht, sobald ein neues Ziel vorgegeben wird.</p>
target_reached	<p>Im Profile Velocity Mode:</p> <p>Das Bit wird gesetzt, wenn sich die Drehzahl (velocity_actual_value) des Antriebs im Toleranzfenster befindet (velocity_window, velocity_window_time).</p>
Bit 11	
internal_limit_active	Dieses Bit zeigt an, dass die I ² t-Begrenzung aktiv ist.
Bit 12	Abhangig von <i>modes_of_operation</i> :
set_point_acknowledge	<p>Im Profile Position Mode:</p> <p>Dieses Bit wird gesetzt, wenn der Servoregler das gesetzte Bit new_set_point im controlword erkannt hat. Es wird wieder gelscht, nachdem das Bit new_set_point im controlword auf Null gesetzt wurde. Siehe hierzu unbedingt auch den Abschnitt 5.3 <i>Betriebsart Positionieren (Profile Position Mode)</i> auf Seite 145.</p>
speed_0	<p>Im Profile Velocity Mode:</p> <p>Dieses Bit wird gesetzt, wenn sich die aktuelle Ist-Drehzahl (velocity_actual_value) des Antriebes im zugehrigen Toleranzfenster befindet (velocity_threshold).</p>
homing_attained	<p>Im Homing Mode:</p> <p>Dieses Bit wird gesetzt, wenn die Referenzfahrt ohne Fehler beendet wurde.</p>
ip_mode_active	<p>Im Interpolated Position Mode:</p> <p>Dieses Bit zeigt an, dass die Interpolation aktiv ist und die Interpolations-Datensatze ausgewertet werden. Es wird gesetzt, wenn dies durch das Bit enable_ip_mode im controlword angefordert wurde. Siehe hierzu unbedingt auch den Abschnitt 5.4 <i>Interpolated Position Mode</i> auf Seite 150.</p>

Bit 13	Abhängig von <i>modes_of_operation</i>:
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 parametrisierten 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 (min_position_limit , max_position_limit).
Bit 14	herstellerspezifisch
manufacturer_statusbit	Die Bedeutung dieses Bits ist konfigurierbar: Es kann gesetzt werden, wenn ein beliebiges Bit des manufacturer_statusword_1 gesetzt bzw. zurückgesetzt wird. Siehe hierzu auch den Abschnitt 4.5.2 <i>Objekt 2000h: manufacturer_statuswords</i> auf Seite 126.
Bit 15	herstellerspezifisch
trigger_result	Die Bedeutung dieses Bits ist konfigurierbar: Es wird gesetzt, wenn ein Sample- Ereignis eingetreten ist und die Samplemaske entsprechend gesetzt ist. Siehe hierzu auch den Abschnitt 3.15 <i>Erfassen von Positionen</i> auf Seite 98.

4.5.2 Objekt 2000_h: manufacturer_statuswords

Die Objektgruppe `manufacturer_statuswords` zeigt herstellerspezifisch weitere Zustände des Servoreglers an.

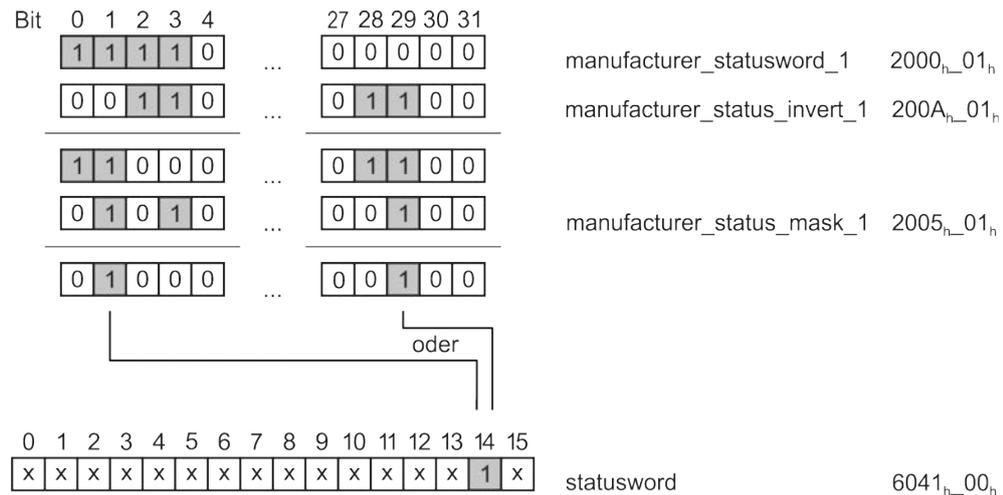
Index	2000 _h		
Name	manufacturer_statuswords		
Type	RECORD		01 _h
Sub-Index	01 _h		
Name	manufacturer_statusword_1		
Info	--	ro	PDO UINT32
Value	--	--	

Bit	Wert	Name
0	00000001 _h	is_referenced
1	00000002 _h	commutation_valid
2	00000004 _h	ready_for_enable
3	00000008 _h	ipo_in_target
...		
8	00000100 _h	safe_standstill

Bit 0	
is_referenced	Das Bit wird gesetzt, wenn der Servoregler referenziert ist. Dies ist der Fall, wenn entweder eine Referenzfahrt erfolgreich durchgeführt wurde oder aufgrund des angeschlossenen Gebersystems (z.B. bei einem Absolutwertgeber) keine Referenzfahrt nötig ist.
Bit 1	
commutation_valid	Das Bit wird gesetzt, wenn die Kommutierinformation gültig ist. Es ist insbesondere bei Gebersystemen ohne Kommutierinformation (z.B. Linearmotoren) hilfreich, weil dort die automatische Kommutierungsfindung einige Zeit in Anspruch nehmen kann. Wird dieses Bit überwacht, kann z.B. ein Timeout der Steuerung bei Freigabe des Servoreglers verhindert werden.

Bit 2	
ready_for_enab	<p>Das Bit wird gesetzt, wenn alle Bedingungen vorliegen, um den Servoregler freizugeben und nur noch die Servoreglerfreigabe selber fehlt. Folgende Bedingungen müssen vorliegen:</p> <ul style="list-style-type: none">• Der Antrieb ist fehlerfrei• Der Zwischenkreis ist geladen• Die Winkelgeberauswertung ist bereit. Es sind keine Prozesse (z.B. serielle Übertragung) aktiv, die eine Freigabe verhindern• Es ist kein blockierender Prozess aktiv (z.B. die automatische Motorparameter- Identifikation)
Bit 3	
ipo_in_target	<p>Das Bit wird gesetzt, wenn die Positioniersteuerung die Zielposition erreicht hat. Im Gegensatz zu target_reached wird nicht zusätzlich überprüft, ob die Istposition auch der Zielposition entspricht.</p>
Bit 8	
safe_standstill	<p>Das Bit wird gesetzt, wenn der Regler den sicheren Zustand „Safe Torque Off“ (STO) eingenommen hat. Siehe hierzu auch den jeweiligen Abschnitt im Produkthandbuch, z.B. Abschnitt <i>STO (Safe Torque Off)</i> im Produkthandbuch BL 4000-C.</p>

Mithilfe der Objekte `manufacturer_status_masks` und `manufacturer_status_invert` können ein oder mehrere Bits der `manufacturer_statuswords` in Bit 14 (`manufacturer_statusbit`) des `statusword` (`6041h`) einblendend werden. Alle Bits des `manufacturer_statusword_1` können über das korrespondierende Bit in `manufacturer_status_invert_1` invertiert werden. Somit können auch Bits auf den Zustand „zurückgesetzt“ überwacht werden. Nach der Invertierung werden die Bits maskiert, d.h. nur wenn das korrespondierende Bit in `manufacturer_status_mask_1` gesetzt ist, wird das Bit weiter ausgewertet. Ist nach der Maskierung noch mindestens ein Bit gesetzt, wird auch Bit 14 des `statusword` gesetzt. Die folgende Abbildung verdeutlicht dieses beispielhaft:



BEISPIEL

Bit 14 des `statusword` soll gesetzt werden, wenn der Antrieb referenziert ist:

Objekt	Wert	
<code>manufacturer_status_invert_1</code>	0x00000000	Kein Bit invertieren
<code>manufacturer_status_mask_1</code>	0x00000001	Bit 0 einblenden

Bit 14 des `statusword` soll gesetzt werden, wenn der Antrieb keine gültige Kommutierlage hat:

Objekt	Wert	
<code>manufacturer_status_invert_1</code>	0x00000002	Bit 1 invertieren
<code>manufacturer_status_mask_1</code>	0x00000002	Bit 1 einblenden

Bit 14 des `statusword` soll gesetzt werden, wenn der Antrieb nicht bereit zur Freigabe ODER referenziert ist:

Objekt	Wert	
<code>manufacturer_status_invert_1</code>	0x00000004	Bit 2 invertieren
<code>manufacturer_status_mask_1</code>	0x00000005	Bit 0 und Bit 2 einblenden

4.5.3 Objekt 2005_h: manufacturer_status_masks

Mit dieser Objektgruppe wird festgelegt, welche gesetzten Bits der `manufacturer_statuswords` in das `statusword` eingeblendet werden.

Index	2005 _h			
Name	manufacturer_status_masks			
Type	RECORD			01 _h
Sub-Index	01 _h			
Name	manufacturer_status_mask_1			
Info	--	rw	PDO	UINT32
Value	--	0		

4.5.4 Objekt 200A_h: manufacturer_status_invert

Mit dieser Objektgruppe wird festgelegt, welche Bits der `manufacturer_statuswords` invertiert in das `statusword` eingeblendet werden.

Index	200A _h			
Name	manufacturer_status_invert			
Type	RECORD			01 _h
Sub-Index	01 _h			
Name	manufacturer_status_invert_1			
Info	--	rw	PDO	UINT32
Value	--	0		

4.5.5 Objekt 2001_h: manufacturer_warnings

Die Objektgruppe `manufacturer_warnings` zeigt herstellerspezifisch weitere Zustände des Servoreglers an.

Index	2001 _h			
Name	manufacturer_warnings			
Type	RECORD			01 _h
Sub-Index	01 _h			
Name	manufacturer_warnings_1			
Info	--	ro	PDO	UINT32
Value	--	--		

Bit	Wert	Name
0	00000001 _h	l_lim_switch_lock
1	00000002 _h	r_lim_switch_lock
2	00000004 _h	warning_active

Bit 0	
l_lim_switch_lock	Dieses Bit zeigt an, dass eine Drehrichtung gesperrt ist, weil der linke Endschalter ausgelöst wurde. Die Sollwertsperre wird wieder gelöscht, wenn eine Fehlerquittierung durchgeführt wird (Siehe controlword , fault_reset).
Bit 1	
r_lim_switch_lock	Dieses Bit zeigt an, dass eine Drehrichtung gesperrt ist, weil der rechte Endschalter ausgelöst wurde. Die Sollwertsperre wird wieder gelöscht, wenn eine Fehlerquittierung durchgeführt wird (Siehe controlword , fault_reset).
Bit 2	
warning_active	Dieses Bit zeigt an, dass im Servoregler eine Warnung aktiv ist, siehe den entsprechenden Abschnitt im Produkthandbuch, z.B. Abschnitt <i>Störungsmeldungen</i> im Produkthandbuch BL 4000-C.

Mithilfe des Objekts [manufacturer_warning_masks](#) können ein oder mehrere Bits der [manufacturer_warnings](#) in Bit 7 (warning) des [statusword](#) (6041_h) eingeblendet werden. Nur wenn das korrespondierende Bit in [manufacturer_warning_mask_1](#) gesetzt ist, wird das Bit weiter ausgewertet. Ist nach der Maskierung noch mindestens ein Bit gesetzt, wird auch Bit 7 des [statusword](#) gesetzt.

4.5.6 Objekt 2006_h: manufacturer_warning_masks

Mit dieser Objektgruppe wird festgelegt, welche gesetzten Bits der [manufacturer_warnings](#) in das [statusword](#) eingeblendet werden.

Index	2006 _h			
Name	manufacturer_warning_masks			
Type	RECORD			01 _h
Sub-Index	01 _h			
Name	manufacturer_warning_mask_1			
Info	--	rw	PDO	UINT32
Value	--	0		

4.6 Beschreibung der weiteren Objekte

4.6.1 Objekt 605B_h: shutdown_option_code

Mit dem Objekt `shutdown_option_code` wird vorgegeben, wie sich der Servoregler beim Zustandsübergang 8 (von `OPERATION_ENABLE` nach `READY_TO_SWITCH_ON`) verhält. Das Objekt zeigt das unveränderliche Verhalten des Servoreglers an.

Index	605B _h		
Name	shutdown_option_code		
Info	--	rw	PBC INT16
Value	0	--	

Wert	Name
0	Endstufe wird ausgeschaltet, Motor ist frei drehbar

4.6.2 Objekt 605C_h: disable_operation_option_code

Mit dem Objekt `disable_operation_option_code` wird vorgegeben, wie sich der Servoregler beim Zustandsübergang 5 (von `OPERATION_ENABLE` nach `SWITCHED_ON`) verhält. Das Objekt zeigt das unveränderliche Verhalten des Servoreglers an.

Index	605C _h		
Name	disable_operation_option_code		
Info	--	rw	PBC INT16
Value	-1	--	

Wert	Name
-1	Bremsen mit <code>quickstop_deceleration</code>

4.6.3 Objekt 605A_h: quick_stop_option_code

Mit dem Parameter `quick_stop_option_code` wird vorgegeben, wie sich der Servoregler bei einem [Quick Stop](#) verhält. Das Objekt zeigt das unveränderliche Verhalten des Servoreglers an.

Index	605A _h		
Name	quick_stop_option_code		
Info	--	rw	PBO INT16
Value	2	--	

Wert	Name
2	Bremsen mit quickstop_deceleration

4.6.4 Objekt 605E_h: fault_reaction_option_code

Mit dem Objekt `fault_reaction_option_code` wird vorgegeben, wie sich der Servoregler bei einem Fehler (fault) verhält. Da bei Metronix Servoreglern die Fehlerreaktion vom jeweiligen Fehler abhängt, kann dieses Objekt nicht parametrierbar sein und gibt immer 0 zurück. Um die Fehlerreaktion der einzelnen Fehler zu verändern siehe Abschnitt 3.18 *Fehlermanagement* auf Seite 110.

Index	605E _h		
Name	fault_reaction_option_code		
Info	--	rw	PBO INT16
Value	0	--	

5 Betriebsarten

5.1 Einstellen der Betriebsart

5.1.1 Übersicht

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

- momentengeregelter Betrieb (profile torque mode)
- drehzahl geregelter Betrieb (profile velocity mode)
- Referenzfahrt (homing mode)
- Positionierbetrieb (profile position mode)
- Synchroner Positionsvorgabe (CANopen: interpolated position mode
Ethercat: cyclic synchronous position mode)

5.1.2 Beschreibung der Objekte

5.1.2.1 Objekt 6060_h: modes_of_operation

Mit dem Objekt `modes_of_operation` wird die Betriebsart des Servoreglers eingestellt.

Index	6060 _h			
Name	modes_of_operation			
Info	--	rw	PDO	INT8
Value	1, 3, 4, 6, 7, 8		--	

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

HINWEIS Aktuelle Betriebsart

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.

5.1.2.2 Objekt 6061h: modes_of_operation_display

Im Objekt `modes_of_operation_display` kann die aktuelle Betriebsart des Servoreglers gelesen werden.

Index	6061 _h		
Name	modes_of_operation_display		
Info	--	ro	PDO INT8
Value	-14, -13, -11, -1, 1, 3, 4, 6, 7, 8		--

Wird eine Betriebsart über das Objekt 6060_h eingestellt, werden neben der eigentlichen Betriebsart auch die folgenden Sollwert- Aufschaltungen (Sollwert- Selektor) vorgenommen, die für einen Betrieb des Servoreglers unter CANopen nötig sind:

Selektor	Profile Velocity Mode	Profile Torque Mode
A	Drehzahl- Sollwert (Feldbus 1)	Drehmoment- Sollwert (Feldbus 1)
B	Ggf. Momentenbegrenzung	inaktiv
C	Drehzahl- Sollwert (Synchrondrehz.)	inaktiv

Außerdem wird die Sollwert- Rampe grundsätzlich eingeschaltet. Nur wenn diese Aufschaltungen in der genannten Weise eingestellt sind, wird auch eine der CANopen- Betriebsarten zurückgegeben. Werden diese Einstellungen z.B. mit dem Metronix ServoCommander® geändert, wird eine jeweilige „User“- Betriebsart zurückgegeben, um anzuzeigen, dass die Selektoren verändert wurden.

Wert	Aktion
-1	Unbekannte Betriebsart / Betriebsartenwechsel
-11	User Position Mode
-13	User Velocity Mode
-14	User Torque Mode
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
8	Cyclic Synchronous Position Mode

HINWEIS Setzen der Betriebsart

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 kurzzeitig „ungültige Betriebsart“ (-1) angezeigt werden.

5.2 Betriebsart Referenzfahrt (Homing Mode)

5.2.1 Übersicht

In diesem Kapitel wird beschrieben, wie der Servoregler 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.

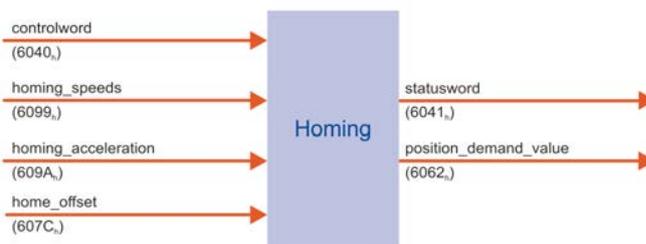


Abbildung 8: Die Referenzfahrt

Der Benutzer kann die Geschwindigkeit, die Beschleunigung und die Art der Referenzfahrt bestimmen. Mit dem Objekt `home_offset` kann die Nullposition des Servoreglers 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. Soll der Antrieb nicht neu referenziert werden, sondern lediglich die Position auf einen vorgegebenen Wert gesetzt werden, kann das Objekt `2030h` (`set_position_absolute`) benutzt werden. Siehe hierzu Abschnitt 3.7.2.13 *Objekt 2030_h: set_position_absolute* auf Seite 76.

HINWEIS Fahrt auf Nullposition nicht Bestandteil der Referenzfahrt

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

Dieses Verhalten kann durch das Objekt `6510h_F0h` (`compatibility_control`) geändert werden, sodass immer eine Fahrt auf Null ausgeführt wird (siehe Abschnitt 3.2 *Kompatibilitäts-Einstellungen* auf Seite 45).

5.2.2 Beschreibung der Objekte

5.2.2.1 Wichtige Objekte in anderen Kapiteln

Index	Name	Kapitel	Seite
6040 _h	controlword	Gerätesteuerung (Device Control)	112
6041 _h	statusword		

5.2.2.2 Objekt 607C_h: home_offset

Das Objekt `home_offset` legt die Verschiebung der Nullposition gegenüber der ermittelten Referenzposition fest. Die Auswirkung dieses Objekts kann angepasst werden. Siehe hierzu auch Abschnitt 3.2.2.1 *Objekt 6510h_F0h: compatibility_control* auf Seite 45.

Index	607C_h			
Name	home_offset			
Info	position_unit	rw	PDO	INT32
Value	--			--

5.2.2.3 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 vier mögliche Referenzfahrt-Signale: den negativen und positiven Endschalter, den Referenzschalter und den (periodischen) Nullimpuls des Winkelgebers.

Außerdem kann der Servoregler 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, der Referenzschalter, 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			
Info	--	rw	PDO	INT8
Value	-18, -17, -2, -1, 1, 2, 7, 11, 17, 18, 23, 27, 32, 33, 34, 35			--

	Richtung	Ziel	Bezugspunkt für Null	DS402
-18	positiv	Anschlag	Anschlag	-18
-17	negativ	Anschlag	Anschlag	-17
-2	positiv	Anschlag	Nullimpuls	-2
-1	negativ	Anschlag	Nullimpuls	-1
1	negativ	Endschalter	Nullimpuls	1
2	positiv	Endschalter	Nullimpuls	2
7	positiv	Referenzschalter	Nullimpuls	7
11	negativ	Referenzschalter	Nullimpuls	11
17	negativ	Endschalter	Endschalter	17
18	positiv	Endschalter	Endschalter	18
23	positiv	Referenzschalter	Referenzschalter	23
27	negativ	Referenzschalter	Referenzschalter	27
32	negativ	Nullimpuls	Nullimpuls	33
33	positiv	Nullimpuls	Nullimpuls	34
34		Keine Fahrt	Aktuelle Ist-Position	35

HINWEIS Referenzfahrt- Methoden nicht gemäß DS402 zugeordnet

In bisherigen CANopen- Implementierungen sind die Referenzfahrt- Methoden 32, 33, 34 und 35 nicht gemäß DS402 zugeordnet. Daher besteht die Möglichkeit über das Objekt [compatibility_control](#) (siehe Abschnitt 3.2 *Kompatibilitäts- Einstellungen* auf Seite 45) die Zuordnung gemäß DS402 auszuwählen. In diesem Fall sind die Methoden- Nummern in der Spalte "DS402" zu verwenden.

Für Kompatibilität zu früheren Versionen brauchen keine Änderungen durchgeführt werden und es können die bisherigen Nummern verwendet werden.

Die [homing_method](#) kann nur verstellt werden, wenn die Referenzfahrt nicht aktiv ist. Ansonsten wird die Fehlermeldung *08 00 00 22h* zurückgegeben. Der Ablauf der einzelnen Methoden ist in Abschnitt 5.2.3 *Referenzfahrt-Abläufe* auf Seite 139 ausführlich erläutert.

5.2.2.4 Objekt 6099_h: homing_speeds

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

Index	6099 _h		
Name	homing_speeds		
Type	ARRAY		02 _h

Sub-Index	01_h			
Name	speed_during_search_for_switch			
Info	speed_unit	rw	PDO	UINT32
Value	--	--		
Sub-Index	02_h			
Name	speed_during_search_for_zero			
Info	speed_unit	rw	PDO	UINT32
Value	--	--		

HINWEIS Setzen von Bit 6 im Objekt `compatibility_control`

Wird Bit 6 im Objekt `compatibility_control`, (siehe Abschnitt 3.2 *Kompatibilitäts-Einstellungen* auf Seite 45) gesetzt, kann nach der Referenzfahrt zum Beispiel eine Fahrt auf Null durchgeführt werden.

Ist dieses Bit gesetzt und das Objekt `speed_during_search_for_switch` wird beschrieben, wird daher sowohl die Geschwindigkeit für die Schaltersuche als auch die Geschwindigkeit für die Fahrt auf Null beschrieben.

5.2.2.5 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			
Info	acceleration_unit	rw	PDO	UINT32
Value	--	--		

5.2.2.6 Objekt 2045_h: `homing_timeout`

Die Referenzfahrt kann auf ihre maximale Ausführungszeit überwacht werden. Dazu kann mit dem Objekt `homing_timeout` die maximale Ausführungszeit angegeben werden. Wird diese Zeit überschritten, ohne dass die Referenzfahrt beendet wurde, wird der Fehler 11-3 ausgelöst.

Index	2045_h			
Name	homing_timeout			
Info	ms	rw	PDO	UINT16
Value	0 (aus), 1 ... 65535	--		

5.2.3 Referenzfahrt-Abläufe

5.2.3.1 Methode -17 und -18: Anschlag

Bei dieser Methode bewegt sich der Antrieb in positiver (-18) oder negativer (-17) Richtung, bis er den Anschlag erreicht. Im Normalfall wird eine Erhöhung des i^2t -Werts um 50 % als Kriterium verwendet, um den Anschlag zu erkennen. Alternativ kann ein Vergleichsmoment angegeben werden, bei dem der Anschlag als erkannt gilt (siehe Abschnitt *Registerkarte: Momente* im jeweiligen Produkthandbuch). Der Anschlag muss mechanisch so dimensioniert sein, dass er bei dem parametrisierten Maximalstrom keinen Schaden nimmt. Die Nullposition bezieht sich direkt auf den Anschlag. Da in diesem Fall die Nullposition direkt auf dem Anschlag liegen würde, sollte der Parameter **Offset Startposition** verwendet werden, um die Nullposition geeignet zu verschieben.

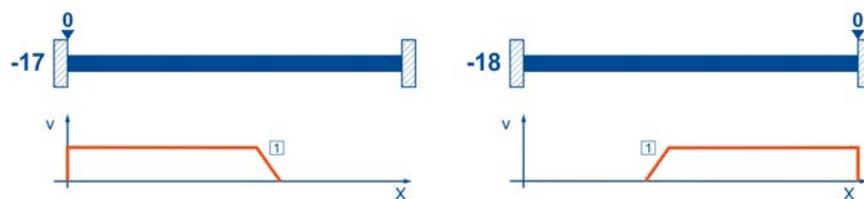


Abbildung 9: Referenzfahrt auf den Anschlag

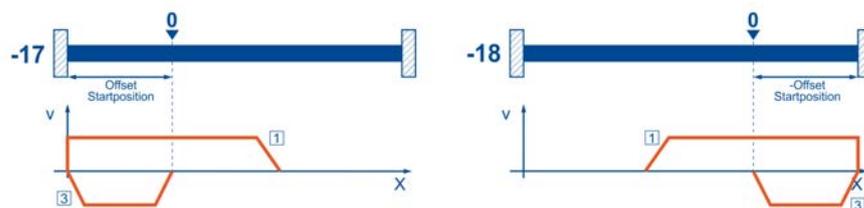


Abbildung 10: Verwendung von "Offset Startposition"

5.2.3.2 Methoden -1 und -2: Anschlag mit Nullimpulsauswertung

Diese Methoden entsprechen den Methoden -17 und -18, die Nullposition bezieht sich allerdings zusätzlich auf den ersten Nullimpuls des Winkelgebers in negativer (-2) bzw. positiver (-1) Richtung vom Anschlag.

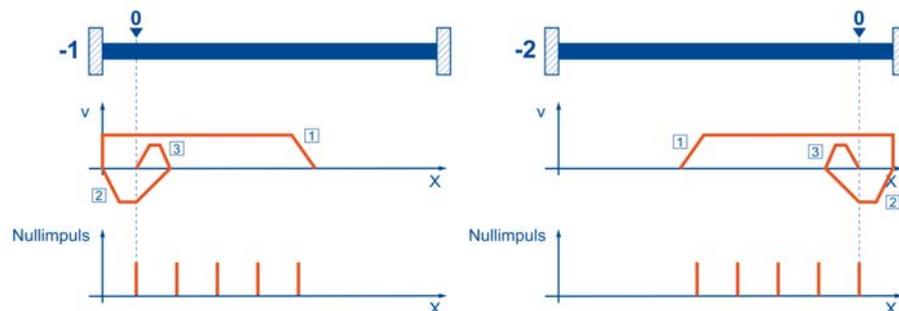


Abbildung 11: Referenzfahrt auf Anschlag mit Auswertung des Nullimpulses

5.2.3.3 Methoden 17 und 18: Positiver und negativer Endschalter

Bei diesen Methoden bewegt sich der Antrieb zunächst mit Suchgeschwindigkeit in positiver (18) bzw. negativer (17) Richtung, bis er den Endschalter erreicht. Danach fährt der Antrieb in Kriechgeschwindigkeit zurück und sucht die genaue Position des Endschaltes. Die Nullposition bezieht sich auf die fallende Flanke vom Endschalter.

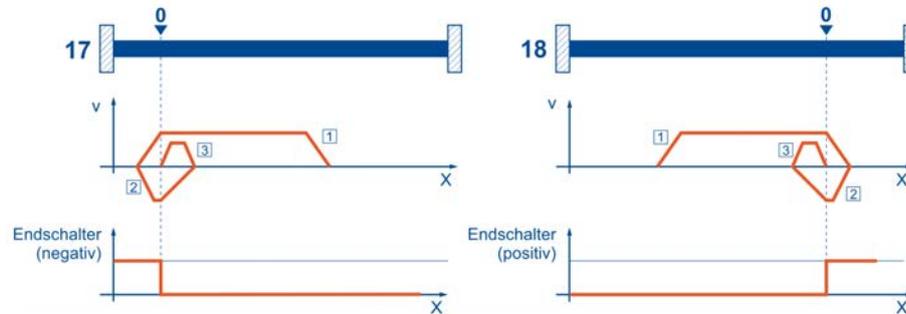


Abbildung 12: Referenzfahrt auf den Endschalter

5.2.3.4 Methoden 1 und 2: Positiver und negativer Endschalter mit Nullimpulsauswertung

Wie bei der vorhergehenden Methode wird zunächst auch der Endschalter gesucht. Zusätzlich bezieht sich die Nullposition allerdings auf den ersten Nullimpuls (Index pulse) des Winkelgebers in negativer (1) bzw. positiver (2) Richtung vom Endschalter.

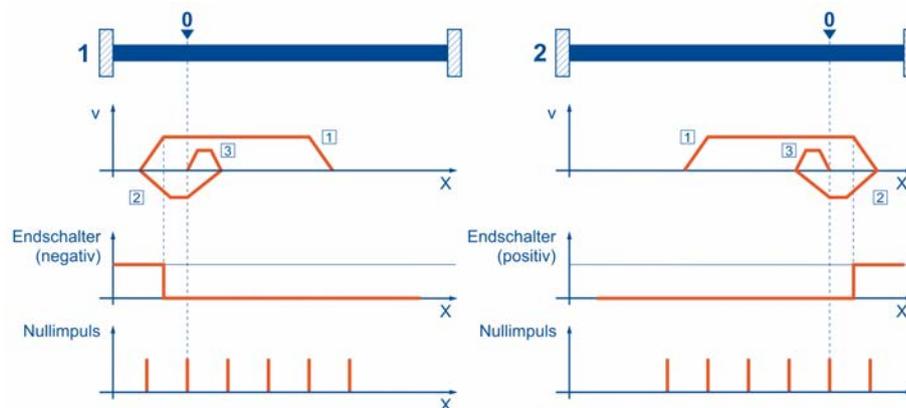


Abbildung 13: Referenzfahrt auf den Endschalter mit Auswertung Nullimpuls

5.2.3.5 Methoden 23 und 27: Referenzschalter

Diese beiden Methoden nutzen einen Referenzschalter, der nur über einen Teil der Strecke aktiv ist. Diese Referenzmethode bietet sich besonders für Rundachsen-Applikationen an, wo der Referenzschalter einmal pro Umdrehung aktiviert wird. Bei dieser Methode bewegt sich der Antrieb zunächst mit Suchgeschwindigkeit in positiver (23) bzw. negativer (27) Richtung, bis er den Referenzschalter erreicht. Danach fährt der Antrieb in Kriechgeschwindigkeit zurück und sucht die genaue Position des Referenzschalters. Die Nullposition bezieht sich auf die fallende Flanke vom Referenzschalter. Falls der Antrieb sich zunächst vom Referenzschalter weg bewegt, bewirkt der jeweilige Endschalter eine Drehrichtungsumkehr, so dass auch in diesem Fall der Referenzschalter gefunden wird.

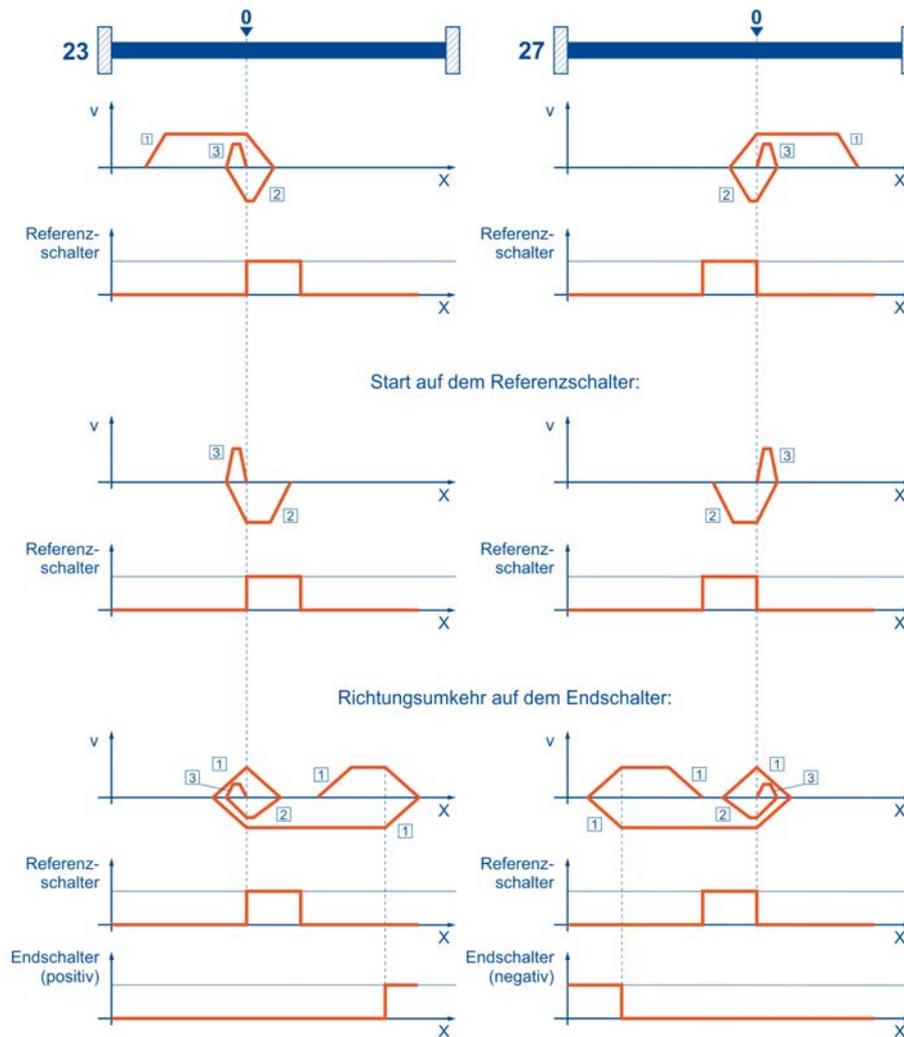


Abbildung 14: Referenzfahrt auf den Referenzschalter

5.2.3.6 Methoden 7 und 11: Referenzschalter und Nullimpulsauswertung

Die Methoden 7 und 11 benutzen wie die Methoden 23 und 27 den Referenzschalter, zusätzlich wird allerdings die Nullposition auf den ersten Nullimpuls in negativer oder positiver Richtung vom Referenzschalter bezogen.

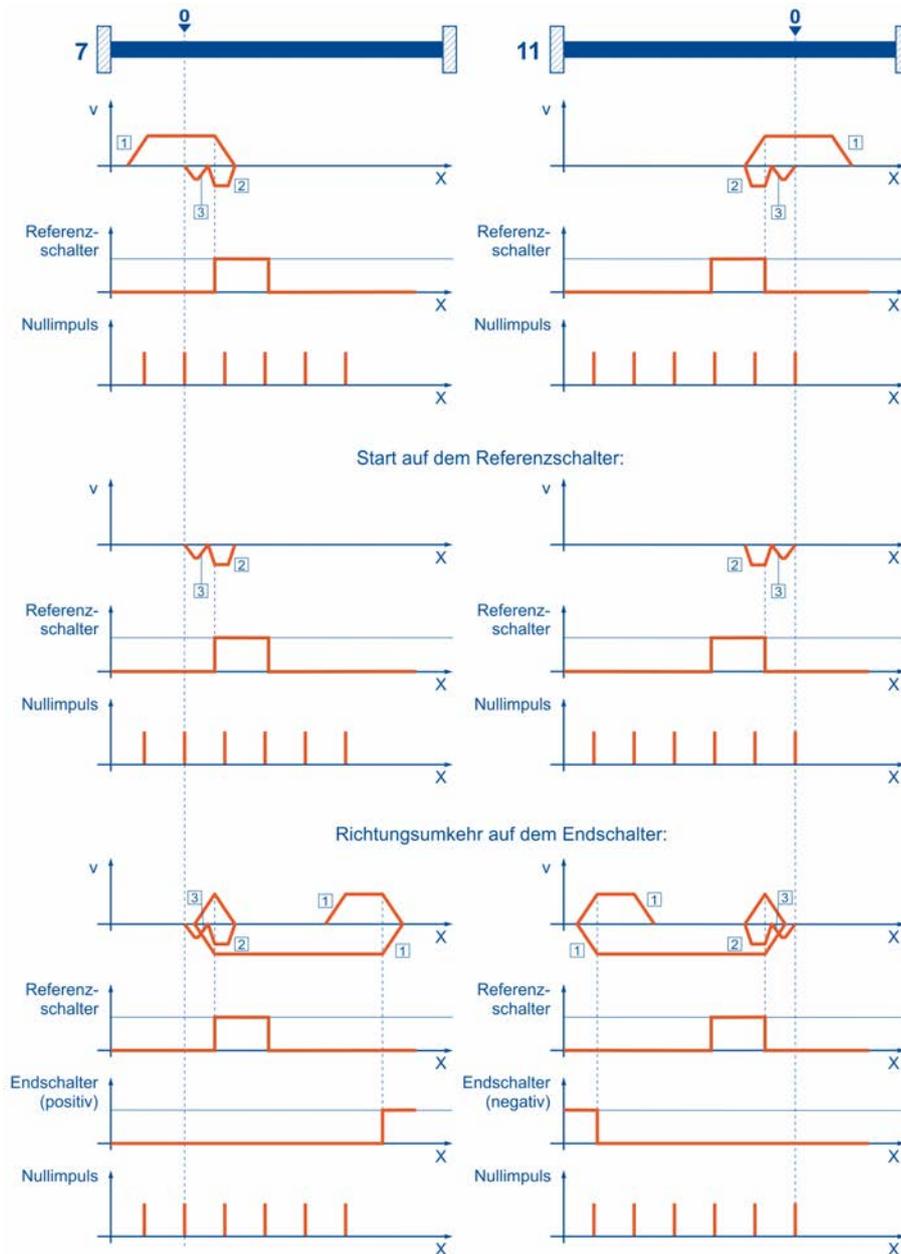


Abbildung 15: Referenzfahrt auf den Referenzschalter mit Nullimpulsauswertung

5.2.3.7 Methoden -23 und -27: Referenzfahrt (pos/neg) auf den Referenzschalter

Diese Methoden ähneln den Methoden 23 und 27. Allerdings wird hier zuerst das jeweilige Ende des Bewegungsbereiches gesucht, z.B. der Endanschlag oder ein Endschalter. Erst dann wird der Referenzschalter gesucht. Dadurch können an dem gleichen Eingang für den Referenzschalter mehrere Schalter angeschlossen sein. Während der Referenzfahrt wird dann der „letzte“ Schalter in Suchrichtung als Referenzschalter verwendet. Bei der Methode -23 bewegt sich der Antrieb zunächst in positiver und bei Methode -27 in negativer Richtung. Die Nullposition bezieht sich auf die fallende Flanke vom Referenzschalter.

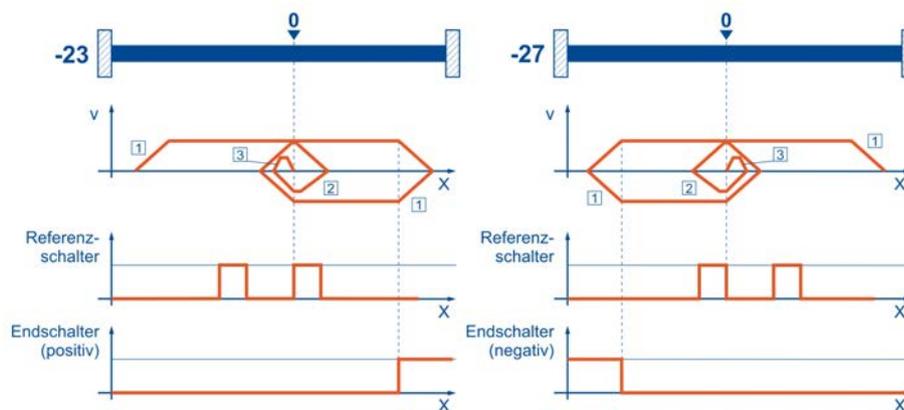


Abbildung 16: Referenzschalter bei positiver und negativer Anfangsbewegung

5.2.3.8 Methoden 32 und 33: Referenzfahrt auf den Nullimpuls

Bei den Methoden 32 und 33 ist die Richtung der Referenzfahrt negativ bzw. positiv. Die Nullposition bezieht sich auf den ersten Nullimpuls vom Winkelgeber in Suchrichtung.

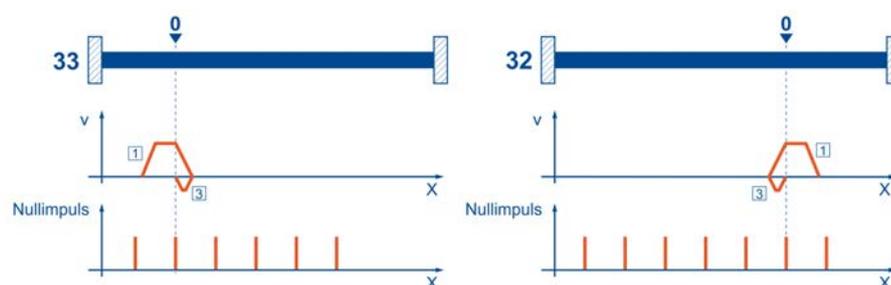


Abbildung 17: Nullimpuls bei negativer (32) und positiver (33) Anfangsbewegung

5.2.3.9 Methode 34: Referenzfahrt auf die aktuelle Position

Bei der Methode 34 wird die Nullposition auf die aktuelle Position bezogen, d.h., die aktuelle Position des Antriebs wird zu Null gesetzt.

5.2.4 Steuerung der Referenzfahrt

Die Referenzfahrt wird durch das `controlword` / `statusword` gesteuert und überwacht. Das Starten erfolgt durch Setzen des Bit 4 im `controlword`.

Bit 4	Bedeutung
0	Referenzfahrt ist nicht aktiv
0 ► 1	Referenzfahrt starten
1	Referenzfahrt ist aktiv
1 ◀ 0	Referenzfahrt unterbrechen

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 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

5.3 Betriebsart Positionieren (Profile Position Mode)

5.3.1 Übersicht

Die Struktur dieser Betriebsart wird in Abbildung 18 (*Fahrkurven-Generator und Lageregler*) 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 (Abschnitt 3.7 *Lageregler (Position Control Function)* auf Seite 70). Diese zwei Funktionsblöcke können unabhängig voneinander eingestellt werden.

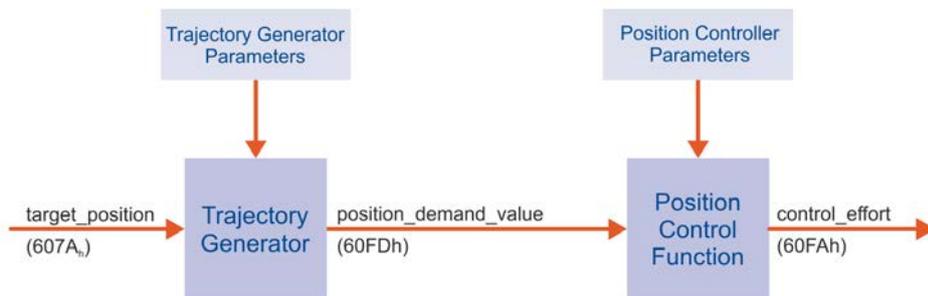


Abbildung 18: Fahrkurven-Generator und Lageregler

5.3.2 Funktionsbeschreibung

Es gibt zwei Möglichkeiten eine Zielposition an den Servoregler zu übergeben:

› Einfacher Fahrauftrag

Wenn der Servoregler eine Zielposition erreicht hat, signalisiert er dies dem Host mit dem Bit `target_reached` (Bit 10 im Objekt `statusword`). In dieser Betriebsart stoppt der Servoregler, wenn er das Ziel erreicht hat.

› Folge von Fahraufträgen

Nachdem der Servoregler ein Ziel erreicht hat, beginnt er sofort das nächste Ziel anzufahren. Dieser Übergang kann fließend erfolgen, ohne dass der Servoregler zwischendurch zum Stillstand kommt.

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

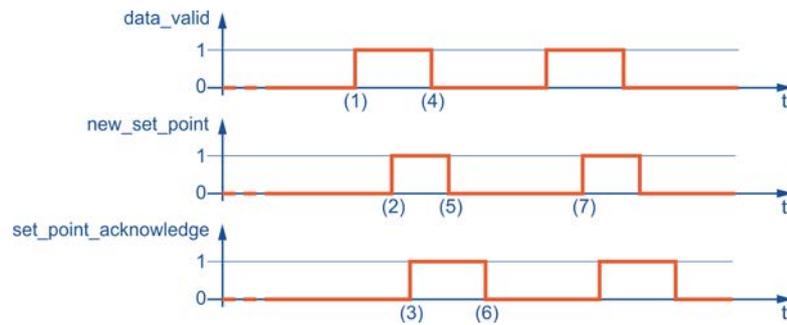


Abbildung 19: Fahrauftrag-Übertragung von einem Host

In Abbildung 19 (*Fahrauftrag-Übertragung von einem Host*) können Sie sehen, wie der Host und der Servoregler über den CAN-Bus miteinander kommunizieren:

Zuerst werden die Positionierdaten (Zielposition, Fahrgeschwindigkeit, Endgeschwindigkeit und die Beschleunigung) an den Servoregler 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 Servoregler 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 beginnen, einen neuen Positionierdatensatz in den Servoregler einzuschreiben (4) und das Bit `new_set_point` wieder zu löschen (5). Erst wenn der Servoregler 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).

Auf der linken Seite von Abbildung 20 (*Fahraufträge*) wird eine neue Positionierung erst gestartet, nachdem die vorherige vollständig abgeschlossen wurde. Der Host wertet hierzu das Bit `target_reached` im Objekt `statusword` aus.

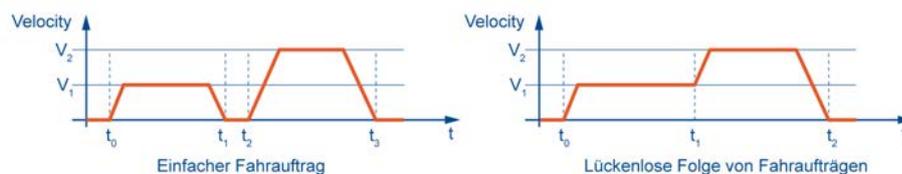


Abbildung 20: Fahraufträge

Auf der rechten Seite wird eine neue Positionierung bereits gestartet, während sich die vorherige noch in Bearbeitung befindet. Der Host übergibt hierzu dem Servoregler das nachfolgende Ziel schon dann, wenn dieser mit dem Löschen des Bits `set_point_acknowledge` signalisiert, dass er den Puffer gelesen und die zugehörige Positionierung gestartet hat. Die Positionierungen werden auf diese Weise nahtlos aneinander gereiht. Damit der Servoregler zwischen den einzelnen Positionierungen nicht jedes Mal kurzzeitig auf Null abbremst, sollte für diese Betriebsart das Objekt `end_velocity` mit dem gleichen Wert wie das Objekt `profile_velocity` beschrieben werden.

Wenn im `controlword` neben dem Bit `new_set_point` auch das Bit `change_set_immediately` auf „1“ gesetzt wird, weist der Host den Servoregler damit an, sofort den neuen Fahrauftrag zu beginnen. Ein bereits in Bearbeitung befindlicher Fahrauftrag wird in diesem Fall abgebrochen.

5.3.3 Beschreibung der Objekte

5.3.3.1 Wichtige Objekte in anderen Kapiteln

Index	Name	Kapitel	Seite
6040 _h	controlword	<i>Gerätesteuerung (Device Control)</i>	112
6041 _h	statusword		
605A _h	quick_stop_option_code		
607E _h	polarity	<i>Umrechnungsfaktoren (Factor Group)</i>	47
6093 _h	position_factor		
6094 _h	velocity_encoder_factor		
6097 _h	acceleration_factor		

5.3.3.2 Objekt 607A_h: target_position

Das Objekt `target_position` (Zielposition) bestimmt, an welche Position der Servoregler 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			
Info	position_unit	rw	PDO	INT32
Value	--			--

5.3.3.3 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_unit` angegeben.

Index	6081_h			
Name	profile_velocity			
Info	speed_unit	rw	PDO	UINT32
Value	--			--

5.3.3.4 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. Normalerweise ist dieses Objekt auf Null zu setzen, damit der Servoregler beim Erreichen der Zielposition (`target_position`) stoppt. Für lückenlose Positionierungen kann eine von Null abweichende Geschwindigkeit vorgegeben werden. Das Objekt `end_velocity` wird in denselben Einheiten wie das Objekt `profile_velocity` angegeben.

Index	6082_h			
Name	end_velocity			
Info	speed_unit	rw	PDO	UINT32
Value	--			--

5.3.3.5 Objekt 6083_h: profile_acceleration

Das Objekt `profile_acceleration` gibt die Beschleunigung an, mit der auf den Sollwert beschleunigt. Es wird in benutzerdefinierten Beschleunigungseinheiten (`acceleration_unit`) angegeben.

Index	6083_h			
Name	profile_acceleration			
Info	acceleration_unit	rw	PDO	UINT32
Value	--			--

5.3.3.6 Objekt 6084_h: profile_deceleration

Das Objekt `profile_deceleration` gibt die Beschleunigung an, mit der gebremst wird. Es wird in benutzerdefinierten Beschleunigungseinheiten (`acceleration_unit`) angegeben.

Index	6084_h			
Name	profile_deceleration			
Info	acceleration_unit	rw	PDO	UINT32
Value	--			--

5.3.3.7 Objekt 6085_n: quick_stop_deceleration

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

Index	6085 _n			
Name	quick_stop_deceleration			
Info	acceleration_unit	rw	PDO	UINT32
Value	--		--	

5.3.3.8 Objekt 6086_n: motion_profile_type

Das Objekt `motion_profile_type` wird verwendet, um die Art des Positionierprofils auszuwählen.

Index	6086 _n			
Name	motion_profile_type			
Info	--	rw	PDO	INT16
Value	0, 2		--	

Wert	Kurvenform
0	Lineare Rampe
2	Ruckfreie Rampe

5.4 Interpolated Position Mode

5.4.1 Übersicht

Der Interpolated Position Mode (IP) ermöglicht die Vorgabe von Lagesollwerten in einer mehrachsigen Anwendung des Servoregler. 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 Servoregler selbständig die Datenwerte zwischen zwei vorgegebenen Positionswerten, wie in der folgenden Grafik skizziert.

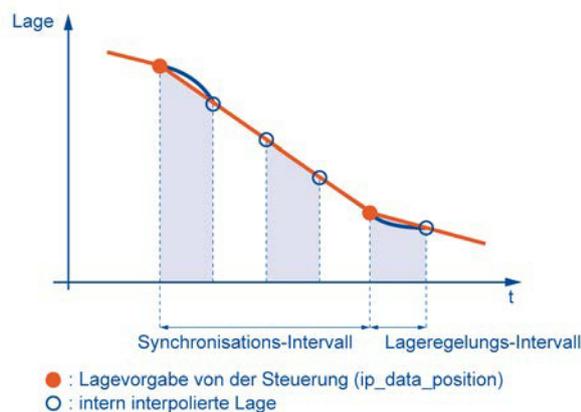


Abbildung 21: Fahrauftrag Lineare Interpolation zwischen zwei Datenwerten

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.

5.4.2 Funktionsbeschreibung

Bevor der Servoregler 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, der Interpolationstyp (`interpolation_submode_select`) und die Art der Synchronisation (`interpolation_sync_definition`). Zusätzlich muss der Zugriff auf den Positionspuffer über das Objekt `buffer_clear` freigegeben werden.

Für die Änderung des Interpolations-Intervalls (Zykluszeit) muss der Parametersatz einmalig gespeichert und der Regler neu gestartet werden. Ob das korrekte Intervall eingestellt ist, kann über das Objekt `synchronous_window_length` (`1006h`) ausgelesen werden. Wenn bereits das korrekte Intervall eingestellt ist, können die ersten vier Schritte im folgenden Beispiel entfallen.

BEISPIEL

Das Beispiel zeigt, welche Schritte nötig sind, um den Regler für den Interpolationsbetrieb vorzubereiten:

Aufgabe	Aktion
Zeiteinheit festlegen (1/10 ms)	60C2 _h 02 _h (interpolation_time_index) = -4
Zeitintervall festlegen (2 ms)	60C2 _h 01 _h (interpolation_time_units) = 20
Parameter sichern	1010 _h 01 _h (save_all_parameters) = 65766173 _h
Reset ausführen	siehe <i>Netzwerkmanagement (NMT-Service)</i>
Neustart abwarten	siehe <i>Bootup</i>
Interpolationsart setzen	60C0 _h (interpolation_submode_select) = -2
Puffer freigeben	60C4 _h 06 _h (buffer_clear) = 1
SYNC-Nachrichten senden	siehe <i>SYNC-Message</i>

Die weiteren Schritte sind in den folgenden Kapiteln beschrieben.

Der **Interpolated Position Mode** wird über das Objekt `modes_of_operation` (6060_h) aktiviert. Ab diesem Zeitpunkt versucht der Servoregler sich auf das externe Zeitraster, welches durch die SYNC-Telegramme vorgegeben wird, aufzusynchronisieren. Konnte sich der Servoregler erfolgreich aufsynchronisieren, meldet er die Betriebsart **Interpolated Position Mode** im Objekt `modes_of_operation_display` (6061_h). Während der Aufsynchronisation meldet der Servoregler "Ungültige Betriebsart (-1)" zurück. Werden nach der erfolgten Aufsynchronisation die SYNC-Telegramme nicht im richtigen Zeitraster gesendet, meldet der Servoregler erneut "Ungültige Betriebsart".

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 Servoregler aus und schreibt diese zyklisch als neuen Sollwert (`interpolation_data_record`) in den Servoregler. Über Handshake-Bits des `controlword` und des `statusword` wird die Übernahme der Daten durch den Servoregler 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 Servoregler über das Statusbit `ip_mode_active` im `statusword` dieses quittiert, werden die Datensätze ausgewertet.

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

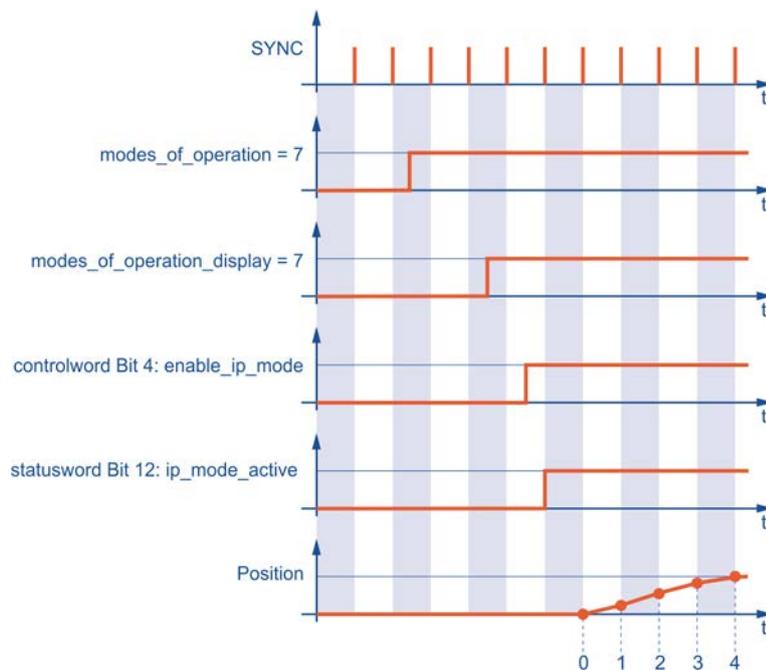


Abbildung 22: Aufsynchronisation und Datenfreigabe

BEISPIEL

Das Beispiel zeigt, welche Schritte nötig sind, um die Interpolation zu beginnen:

Aufgabe	Aktion
SYNC-Nachrichten senden	siehe <i>SYNC-Message</i>
Betriebsart anfordern	6060 _h (modes_of_operation) = 7
Warten, bis Betriebsart eingenommen	6061 _h (modes_of_operation_display) = 7
Aktuelle Istposition lesen	6064 _h (position_actual_value)
Gelesene Istposition als Sollwert vorgeben	60C1 _h _01 _h (ip_data_position)
Interpolation freigeben	6040 _h (controlword), enable_ip_mode setzen
Quittierung durch Regler abwarten	6041 _h (statusword), ip_mode_active abfragen
Interpoliert verfahren	

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.

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 Servoreglerfreigabe und Wechsel in den Zustand `SWITCH_ON_DISABLED`).

Die Interpolation kann dann nur durch eine erneute Aufsynchronisation fortgesetzt werden, da der Servoregler wieder in den Zustand `OPERATION_ENABLE` gebracht werden muss, wodurch das Bit `ip_mode_active` gelöscht wird.

5.4.3 Beschreibung der Objekte

5.4.3.1 Wichtige Objekte in anderen Kapiteln

Index	Name	Kapitel	Seite
6040 _h	controlword	Gerätsteuerung (Device Control)	112
6041 _h	statusword		
6093 _h	position_factor	Umrechnungsfaktoren (Factor Group)	47
6094 _h	velocity_encoder_factor		
6097 _h	acceleration_factor		

5.4.3.2 Objekt 60C0_h: interpolation_submode_select

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

Index	60C0_h			
Name	interpolation_submode_select			
Info	--	rw	PDO	INT16
Value	-2		--	

Wert	Interpolationstyp
-2	Lineare Interpolation ohne Puffer

5.4.3.3 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			
Type	RECORD			02 _h
Sub-Index	01_h			
Name	ip_data_position			
Info	position_unit	rw	PDO	INT32
Value	--	--		
Sub-Index	02_h			
Name	ip_data_controlword			
Info	--	rw	PDO	UINT8
Value	0, 1	0		

Wert	ip_data_position ist
0	Absolute Position
1	Relative Entfernung

HINWEIS Interne Datenübernahme

Die interne Datenübernahme erfolgt bei Schreibzugriff auf Subindex 1. Soll außerdem Subindex 2 verwendet werden, muss dieser vor Subindex 1 beschrieben werden.

5.4.3.4 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` parametrisiert wird.

Zur Synchronisation wird die komplette Reglerkaskade (Strom-, Drehzahl- und Lageregler) auf den externen Takt aufsynchrosiert. Die Änderung des Synchronisationsintervalls wird daher nur nach einem Reset wirksam. Soll das Interpolationsintervall über den CAN-Bus geändert werden, muss daher der Parametersatz gesichert (siehe Abschnitt 3.1 *Parametersätze laden und speichern* auf Seite 42) und ein Reset ausgeführt werden (siehe Abschnitt 6.6 *Netzwerkmanagement (NMT-Service)* auf Seite 190), damit das neue Synchronisations-Intervall wirksam wird. Das Synchronisations-Intervall muss exakt eingehalten werden.

Index	60C2_h			
Name	interpolation_time_period			
Type	RECORD			02 _h
Sub-Index	01_h			
Name	ip_time_units			
Info	gemäß ip_time_index	rw	PDO	UINT8
Value	ip_time_index = -3: 1, 2, ..., 10 ip_time_index = -4: 10, 20, ..., 100	--		
Sub-Index	02_h			
Name	ip_time_index			
Info	--	rw	PDO	INT8
Value	-3, -4	--		

Wert	ip_time_units wird angegeben in
-3	10 ⁻³ Sekunden (ms)
-4	10 ⁻⁴ Sekunden (0.1 ms)

HINWEIS Änderung des Synchronisationsintervalls

Die Änderung des Synchronisationsintervalls wird nur nach einem Reset wirksam. Soll das Interpolationsintervall über den CAN-Bus geändert werden, muss der Parametersatz gesichert und ein Reset ausgeführt werden.

5.4.3.5 Objekt 60C3_h: interpolation_sync_definition

Über das Objekt `interpolation_sync_definition` wird die Art (`synchronize_on_group`) und die Anzahl (`ip_sync_every_n_event`) von Synchronisations-Telegrammen pro Synchronisations-Intervall vorgegeben. Bei Metronix Servoreglern kann nur das Standard-SYNC-Telegramm und 1 SYNC pro Intervall eingestellt werden.

Index	60C3_h			
Name	interpolation_sync_definition			
Type	ARRAY			02 _h
Sub-Index	01_h			
Name	synchronize_on_group			
Info	--	rw	PDO	UINT8
Value	0	0		

Wert	Bedeutung
0	Standard SYNC-Telegramm verwenden

Sub-Index	02_h			
Name	ip_sync_every_n_event			
Info	--	rw	PDO	UINT8
Value	1	1		

5.4.3.6 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			
Type	RECORD			06 _h
Sub-Index	01_h			
Name	max_buffer_size			
Info	--	ro	PDO	UINT32
Value	0	0		
Sub-Index	02_h			
Name	actual_size			
Info	--	rw	PDO	UINT32
Value	0	0		
Sub-Index	03_h			
Name	buffer_organisation			
Info	--	rw	PDO	UINT8
Value	0	0		

Wert	Bedeutung
0	FIFO

Sub-Index	04_h			
Name	buffer_position			
Info	--	rw	PDO	UINT16
Value	0	0		

Sub-Index	05_h			
Name	size_of_data_record			
Info	--	wo	PDO	UINT8
Value	2	2		
Sub-Index	06_h			
Name	buffer_clear			
Info	--	wo	PDO	UINT8
Value	0, 1	0		

Wert	Bedeutung
0	Puffer löschen / Zugriff auf 60C1 _h nicht erlaubt
1	Zugriff auf 60C1 _h freigegeben

5.4.3.7 Objekt 1006_h: communication_cycle_period

Über das Objekt 1006_h ([communication_cycle_period](#)) kann das eingestellte Interpolations-Intervall (=Buszykluszeit) ausgelesen werden. Es entspricht der Zeit t_p , die im Abschnitt *Zykluszeiten der Regelkreise* im Produkthandbuch BL 4000-C beschrieben ist.

Index	1006_h			
Name	communication_cycle_period			
Info	μ s	rw	PDO	UINT32
Value	--			00000000 _h

5.5 Cyclic Synchronous Position Mode

5.5.1 Übersicht

Der *Cyclic Synchronous Position Mode* (CSP) ermöglicht ebenso wie der *Interpolated Position Mode* (IP) die Vorgabe von Lagesollwerten in einer mehrachsigen Anwendung des Servoreglers.

Die wesentlichen Unterschiede sind:

- Die Sollwertvorgabe erfolgt über das Objekt `target_position` (607A_h)
- Die Sollwerte werden direkt nach dem Wechsel in den *Cyclic Synchronous Position Mode* ausgewertet. Es ist nicht nötig, das Bit `enable_ip_mode` im `controlword` zu setzen und auch das Objekt `buffer_clear` (60C4_{h_06}_h) muss nicht beschrieben werden.

5.5.2 Beschreibung der Objekte

5.5.2.1 Wichtige Objekte in anderen Kapiteln

Index	Name	Kapitel	Seite
607A _h	target_position	Objekt 607Ah: target_position	147
60C2 _h	interpolation_time_period	Interpolated Position Mode	112
6040 _h	controlword	Gerätesteuerung (Device Control)	112
6041 _h	statusword		
6093 _h	position_factor	Umrechnungsfaktoren (Factor Group)	47
6094 _h	velocity_encoder_factor		
6097 _h	acceleration_factor		

Der *Cyclic Synchronous Position Mode* definiert keine eigenen Objekte.

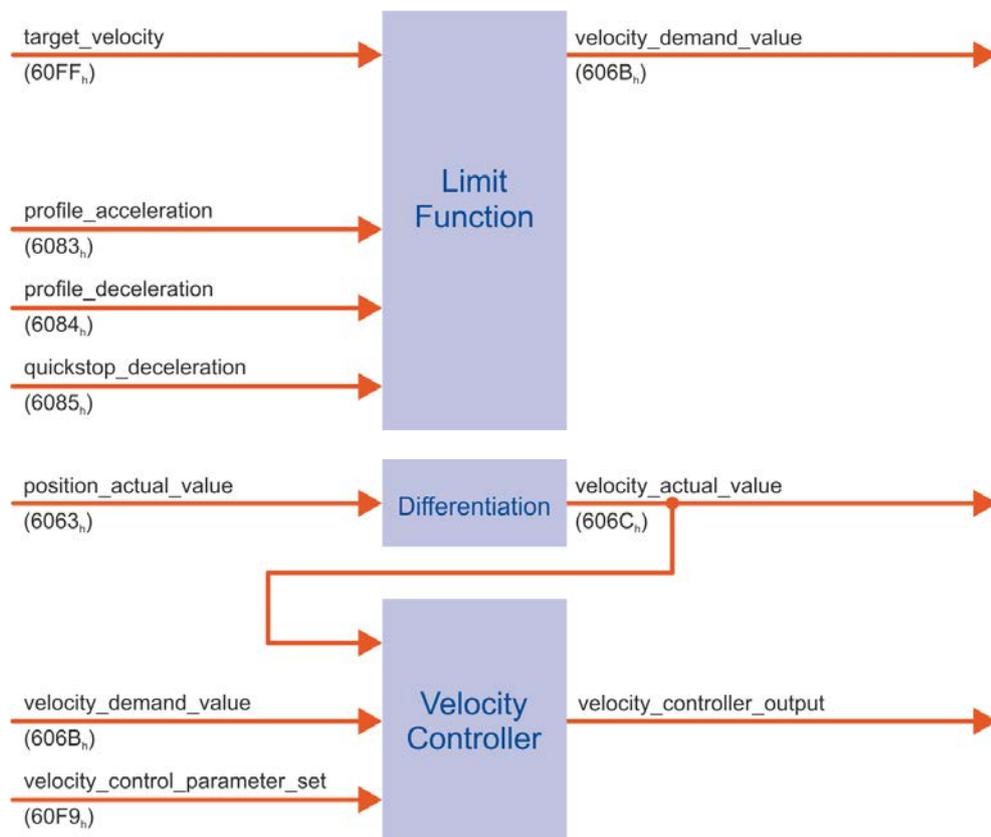
5.6 Betriebsart Drehzahlregelung (Profile Velocity Mode)

5.6.1 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 Abschnitt 5.3 *Betriebsart Positionieren (Profile Position Mode)* auf Seite 145 beschrieben: *profile_acceleration*, *profile_deceleration*, *quick_stop_deceleration*.



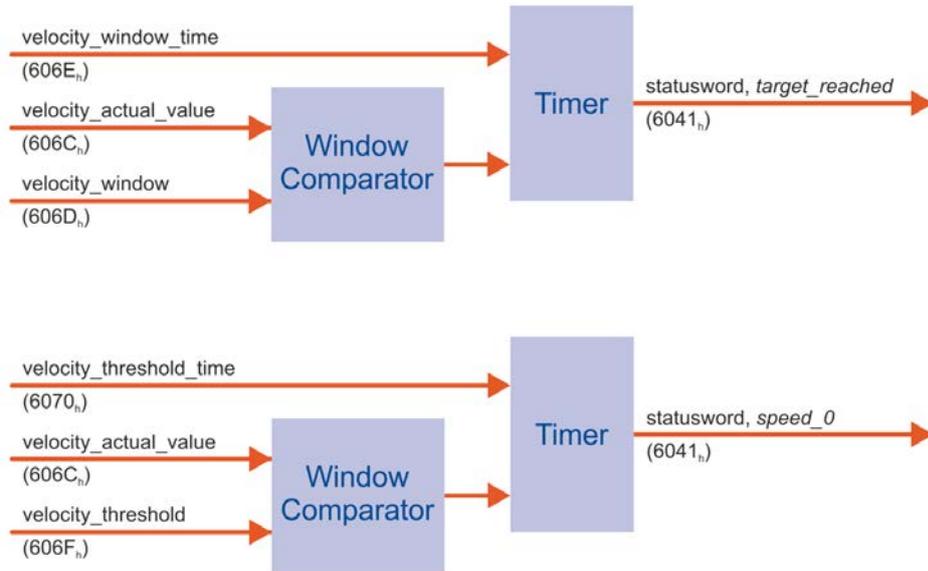


Abbildung 23: Struktur des drehzahlgeregelten Betriebs (Profile Velocity Mode)

5.6.2 Beschreibung der Objekte

5.6.2.1 Wichtige Objekte in anderen Kapiteln

Index	Name	Kapitel	Seite
6040 _h	controlword	<i>Gerätesteuerung (Device Control)</i>	112
6041 _h	statusword		
6064 _h	position_actual_value	<i>Lageregler (Position Control Function)</i>	70
6071 _h	target_torque	<i>Betriebsart Momentenregelung (Profile Torque Mode)</i>	167
6072 _h	max_torque_value		
6083 _h	profile_acceleration	<i>Betriebsart Positionieren (Profile Position Mode)</i>	145
6084 _h	profile_deceleration		
6085 _h	quick_stop_deceleration		
6094 _h	velocity_encoder_factor		

5.6.2.2 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 Metronix Servoreglern 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			
Info	U / 4096 min	ro	PDO	INT32
Value	--			--

5.6.2.3 Objekt 606A_h: sensor_selection_code

Mit diesem Objekt kann der Geschwindigkeitssensor ausgewählt werden. Zur Zeit ist kein separater Geschwindigkeitssensor vorgesehen. Deshalb ist nur der standardmäßige Winkelgeber anwählbar.

Index	606A _h			
Name	sensor_selection_code			
Info	--	rw	PDO	INT16
Value	0		0	

5.6.2.4 Objekt 606B_h: velocity_demand_value

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

Index	606B _h			
Name	velocity_demand_value			
Info	speed_unit	ro	PDO	INT32
Value	--			--

5.6.2.5 Objekt 202E_h: velocity_demand_sync_value

Über dieses Objekt kann die Soll-Drehzahl des Synchronisationsgeber ausgelesen werden. Diese wird durch das Objekt 2022_h `synchronization_encoder_select` (Abschnitt 3.11 *Soll- / Istwertumschaltung* auf Seite 86) definiert. Dieses Objekt wird in benutzerdefinierten Einheiten angegeben.

Index	202E_h			
Name	velocity_demand_sync_value			
Info	speed_unit	ro	PDO	INT32
Value	--		--	

5.6.2.6 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			
Info	speed_unit	ro	PDO	INT32
Value	--		--	

5.6.2.7 Objekt 2074_h: velocity_actual_value_filtered

Über das Objekt `velocity_actual_value_filtered` kann ein gefilterter Drehzahl- Istwert ausgelesen werden, der allerdings nur zu Anzeigezwecken verwendet werden sollte. Im Gegensatz zu `velocity_actual_value` wird `velocity_actual_value_filtered` nicht zur Regelung, wohl aber für den Durchdreheschutz des Servoreglers verwendet. Die Filterzeitkonstante kann über das Objekt 2073_h (`velocity_display_filter_time`) eingestellt werden. Siehe Abschnitt 3.6.2.2 *Objekt 2073h: velocity_display_filter_time* auf Seite 69

Index	2074_h			
Name	velocity_actual_value_filtered			
Info	speed_unit	ro	PDO	INT32
Value			--	

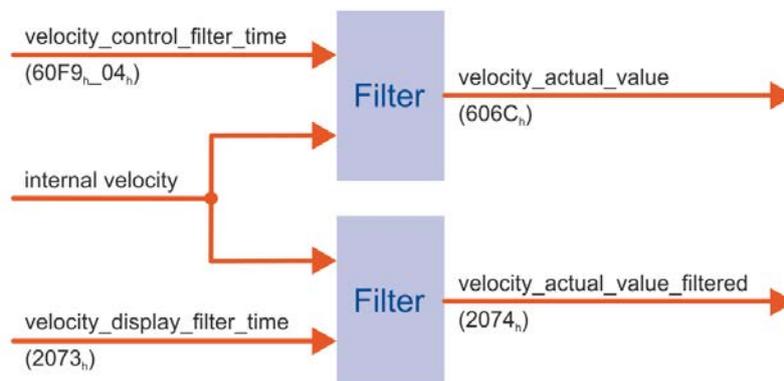


Abbildung 24: Ermittlung von `velocity_actual_value` und `velocity_actual_value_filtered`

5.6.2.8 Objekt 606D_h: velocity_window

Die Objekte `velocity_window_time` und `velocity_window` dienen der Einstellung des Fensterkomparators zum Vergleich des Drehzahl-Istwerts mit der vorgegebenen Endgeschwindigkeit (Objekt 60FF_h: `target_velocity`). Die Drehzahl muss die in `velocity_window_time` spezifizierte Zeit innerhalb des `velocity_window` liegen, damit das Bit 10 `target_reached` im Objekt `statusword` gesetzt wird.

Index	606D_h			
Name	velocity_window			
Info	speed_unit	rw	PDO	UINT16
Value	--			--

5.6.2.9 Objekt 606E_h: velocity_window_time

Die Objekte `velocity_window_time` und `velocity_window` dienen der Einstellung des Fensterkomparators zum Vergleich des Drehzahl-Istwerts mit der vorgegebenen Endgeschwindigkeit (Objekt 60FF_h: `target_velocity`). Die Drehzahl muss die in `velocity_window_time` spezifizierte Zeit innerhalb des `velocity_window` liegen, damit das Bit 10 `target_reached` im Objekt `statusword` gesetzt wird.

Index	606E_h			
Name	velocity_window_time			
Info	ms	rw	PDO	UINT16
Value	0...4999		0	

5.6.2.10 Objekt 606F_h: velocity_threshold

Die Objekte `velocity_threshold` und `velocity_threshold_time` geben an, ab welchem Drehzahl-Istwert der Antrieb als stehend angesehen wird. Wenn der Antrieb die unter `velocity_threshold` vorgegebenen Drehzahl für die `velocity_threshold_time` überschreitet, wird im `statusword` das Bit 12 (`velocity = 0`) gelöscht.

Index	606F_h			
Name	velocity_threshold			
Info	speed_unit	rw	PDO	UINT16
Value	--			--

5.6.2.11 Objekt 6070_h: velocity_threshold_time

Die Objekte `velocity_threshold` und `velocity_threshold_time` geben an, ab welchem Drehzahl-Istwert der Antrieb als stehend angesehen wird. Wenn der Antrieb die unter `velocity_threshold` vorgegebenen Drehzahl für die `velocity_threshold_time` überschreitet, wird im `statusword` das Bit 12 (`velocity = 0`) gelöscht.

Index	6070 _h			
Name	velocity_threshold_time			
Info	ms	rw	PDO	UINT16
Value	0...4999		0	

5.6.2.12 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			
Info	min^{-1}	rw	PDO	UINT16
Value	0...32767		--	

5.6.2.13 Objekt 60FF_h: target_velocity

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

Index	60FF _h			
Name	target_velocity			
Info	speed_unit	rw	PDO	INT32
Value	--		--	

5.6.2.14 Drehzahl-Rampen

Wird als `modes_of_operation Profile Velocity Mode` gewählt, wird grundsätzlich auch die Sollwertrampe aktiviert. Somit ist es möglich über die Objekte `profile_acceleration` und `profile_deceleration` eine sprungförmige Sollwertänderung auf eine bestimmte Drehzahländerungen pro Zeit zu begrenzen.

Der Servoregler ermöglicht es, nicht nur unterschiedliche Beschleunigungen für Bremsen und Beschleunigungen anzugeben, sondern noch zusätzlich nach positiver und negativer Drehzahl zu unterscheiden. Die folgende Abbildung verdeutlicht dieses Verhalten:

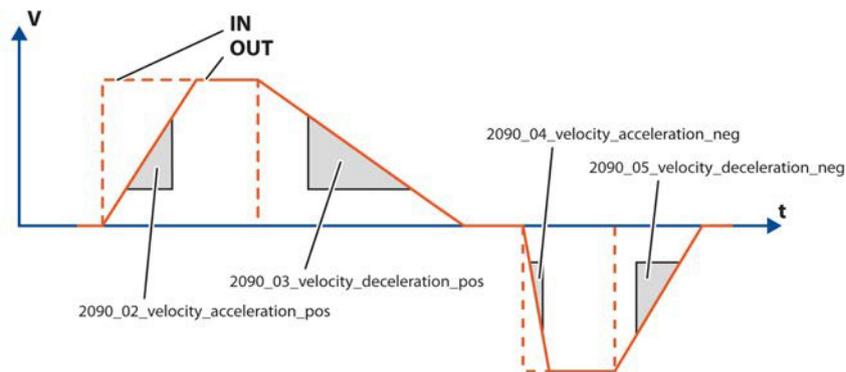


Abbildung 25: Drehzahlrampen

Um diese 4 Beschleunigungen einzeln parametrieren zu können, ist die Objektgruppe `velocity_ramps` vorhanden. Es ist zu beachten, dass die Objekte `profile_acceleration` und `profile_deceleration` die gleichen internen Beschleunigungen verändern, wie die `velocity_ramps`. Wird die `profile_acceleration` geschrieben, werden gemeinsam `velocity_acceleration_pos` und `velocity_acceleration_neg` geändert, wird die `profile_deceleration` geschrieben, werden gemeinsam `velocity_deceleration_pos` und `velocity_deceleration_neg` geändert. Mit dem Objekt `velocity_ramps_enable` lässt sich festlegen, ob die Sollwerte über den Rampengenerator geführt werden, oder nicht.

Index	2090_h		
Name	velocity_ramps		
Type	RECORD		05 _h
Sub-Index	01_h		
Name	velocity_rampe_enable		
Info	--	rw	PBC UINT8
Value	0, 1 (1 = Sollwert über Rampengenerator)	--	
Sub-Index	02_h		
Name	velocity_acceleration_pos		
Info	acceleration_unit	rw	PBC INT32
Value	--	--	

Sub-Index	03_h			
Name	velocity_deceleration_pos			
Info	acceleration_unit	rw	PBO	INT32
Value	--	--		
Sub-Index	04_h			
Name	velocity_acceleration_neg			
Info	acceleration_unit	rw	PBO	INT32
Value	--	--		
Sub-Index	05_h			
Name	velocity_deceleration_neg			
Info	acceleration_unit	rw	PBO	INT32
Value	--	--		

5.7 Betriebsart Momentenregelung (Profile Torque Mode)

5.7.1 Übersicht

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

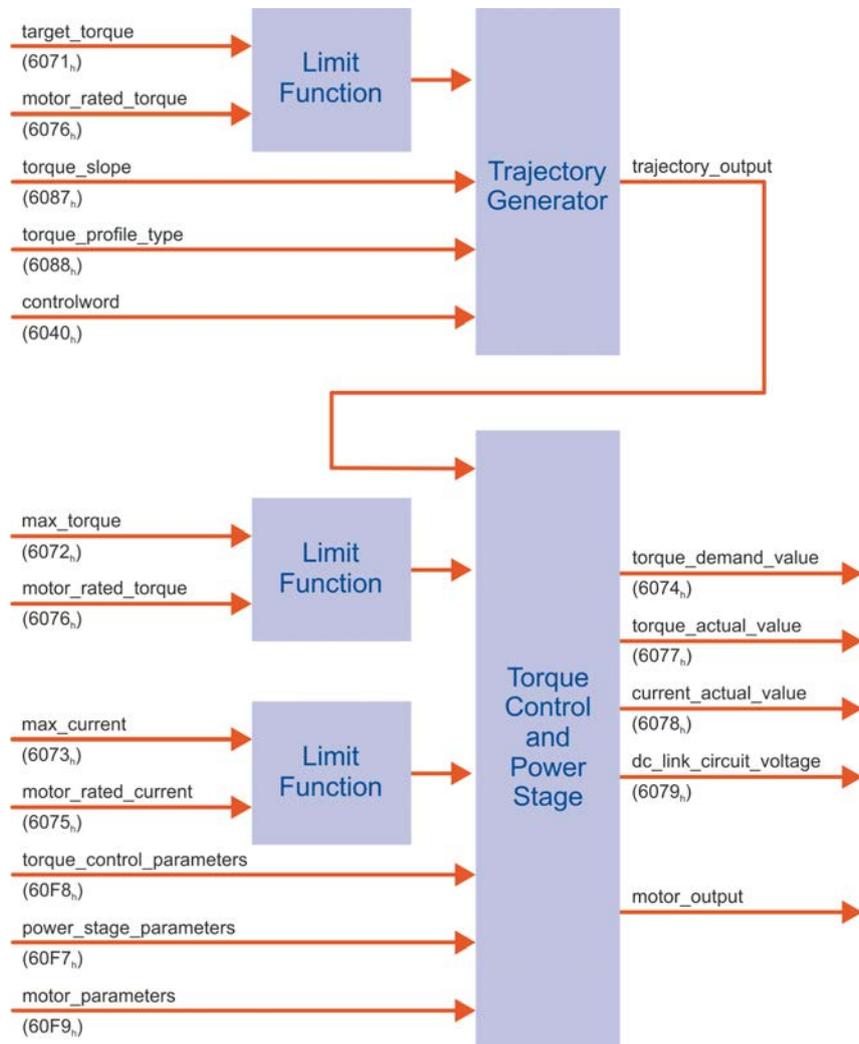


Abbildung 26: : Struktur des drehmomentengeregelten Betriebs

Für den Rampengenerator müssen die Parameter Rampensteilheit `torque_slope` und Rampenform `torque_profile_type` vorgegeben werden.

Wenn im `controlword` das Bit 8 haIt gesetzt wird, senkt der Rampen-Generator das Drehmoment bis auf Null ab. Entsprechend erhöht er es wieder auf das Sollmoment

`target_torque`, wenn das Bit 8 wieder gelöscht wird. In beiden Fällen berücksichtigt der Rampen-Generator die Rampensteilheit `torque_slope` und die Rampenform `torque_profile_type`.

Alle Definitionen innerhalb dieses Kapitels beziehen sich auf rotatorische 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.

5.7.2 Beschreibung der Objekte

5.7.2.1 Wichtige Objekte in anderen Kapiteln

Index	Name	Kapitel	Seite
6040 _h	controlword	<i>Gerätesteuerung (Device Control)</i>	101
60F9 _h	motor_parameters	<i>Stromregler und Motoranpassung</i>	60
6075 _h	motorRatedCurrent		
6073 _h	maxCurrent		

5.7.2.2 Objekt 6071_h: target_torque

Dieser Parameter ist im drehmomentengeregelten Betrieb (Abschnitt 5.7 *Betriebsart Momentenregelung (Profile Torque Mode)* auf Seite 167) der Eingabewert für den Drehmomentenregler. Er wird in Tausendstel des Nennmomentes (Objekt 6076_h) angegeben.

Index	6071_h			
Name	target_torque			
Info	‰ (1000 = motorRatedTorque)	rw	PDO	INT16
Value	--		--	

5.7.2.3 Objekt 6072_h: max_torque

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

HINWEIS Objekt 6072_h korrespondiert mir Objekt 6073_h

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: motorRatedCurrent mit einem gültigen Wert beschrieben wurde.

Index	6072 _h			
Name	max_torque			
Info	‰ (1000 = motorRatedCurrent)	rw	PDO	UINT16
Value	1000...65535		--	

5.7.2.4 Objekt 6074_h: torqueDemandValue

Über dieses Objekt kann das aktuelle Sollmoment in Tausendstel des Nennmoments (6076_h) ausgelesen werden. Berücksichtigt sind hierbei die internen Begrenzungen des Servoreglers (Stromgrenzwerte und I²t-Überwachung).

Index	6074 _h			
Name	torqueDemandValue			
Info	‰ (1000 = motorRatedCurrent)	ro	PDO	INT16
Value	--		--	

5.7.2.5 Objekt 6076_h: motorRatedTorque

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	motorRatedTorque			
Info	0.001 Nm	rw	PDO	UINT32
Value	--		--	

5.7.2.6 Objekt 6077_h: torqueActualValue

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

Index	6077 _h			
Name	torqueActualValue			
Info	‰ (1000 = motorRatedCurrent)	ro	PDO	INT16
Value	--		--	

5.7.2.7 Objekt 6078_h: current_actual_value

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

Index	6078 _h			
Name	current_actual_value			
Info	‰ (1000 = motor Rated current)	ro	PDO	INT16
Value	--		--	

5.7.2.8 Objekt 6079_h: dc_link_circuit_voltage

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

Index	6079 _h			
Name	dc_link_circuit_voltage			
Info	mV	ro	PDO	UINT32
Value	--		--	

5.7.2.9 Objekt 6087_h: torque_slope

Dieser Parameter beschreibt die Änderungsgeschwindigkeit der Sollwertrampe. Diese ist in Tausendstel vom Nennmoment pro Sekunde anzugeben. Beispielsweise wird der Drehmomenten-Sollwert [target_torque](#) von 0 Nm auf den Wert [motor Rated torque](#) erhöht. Wenn der Ausgangswert der zwischengeschalteten Drehmomentenrampe diesen Wert in einer Sekunde erreichen soll, dann ist in diesem Objekt der Wert 1000 einzuschreiben.

Index	6087 _h			
Name	torque_slope			
Info	motor Rated torque / 1000 s	rw	PDO	UINT32
Value	--		--	

5.7.2.10 Objekt 6088_h: torque_profile_type

Mit dem Objekt `torque_profile_type` wird vorgegeben, mit welcher Kurvenform ein Sollwertsprung ausgeführt wird. Zur Zeit ist in diesem Servoregler nur die lineare Rampe implementiert, so dass dieses Objekt nur mit dem Wert 0 beschrieben werden kann.

Index	6088 _h			
Name	torque_profile_type			
Info	--	rw	PDO	INT16
Value	0		0	

Wert	Bedeutung
0	Lineare Rampe

6 Detaillierte Beschreibung des CANopen-Protokolls

6.1 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.

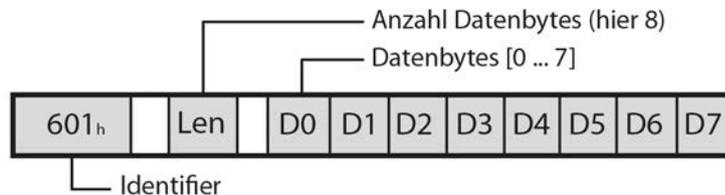
Für den Zugriff auf die CAN-Objekte über den CAN-Bus sind im Wesentlichen zwei Methoden verfügbar: Eine bestätigte Zugriffsart, bei der der Servoregler jeden Parameterzugriff quittiert (über SDOs) und eine unbestätigte Zugriffsart, bei der keine Quittierung erfolgt (über PDOs). In der Regel erfolgt die Parametrierung des Servoreglers über SDOs, während die zyklischen Prozessdaten über PDOs ausgetauscht werden.

Insgesamt sind folgende Kommunikationsobjekte definiert:

SDO	Service Data Object	Werden zur normalen Parametrierung des Servoreglers verwendet.
PDO	Process Data Object	Schneller Austausch von Prozessdaten (z.B. Ist Drehzahl) möglich.
SYNC	Synchronization Message	Synchronisierung mehrerer CAN-Knoten
EMCY	Emergency Message	Übermittlung von Fehlermeldungen.
NMT	Network Management	Netzwerkdienst: Es kann z.B. auf alle CAN-Knoten gleichzeitig eingewirkt werden.
BOOTUP	Error Control Protocol	Einschaltmeldung
HEARTBEAT	Error Control Protocol	Überwachung der Kommunikationsteilnehmer durch regelmäßige Nachrichten.
NODEGUARDING	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:



6.2 SDO-Zugriff

Über die **Service-Data-Objekte (SDO)** kann auf das Objektverzeichnis des Servoreglers zugegriffen werden.

SDO-Zugriffe gehen immer von der übergeordneten Steuerung (Host) aus. Dieser sendet an den Servoregler 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 Quittierung dient.

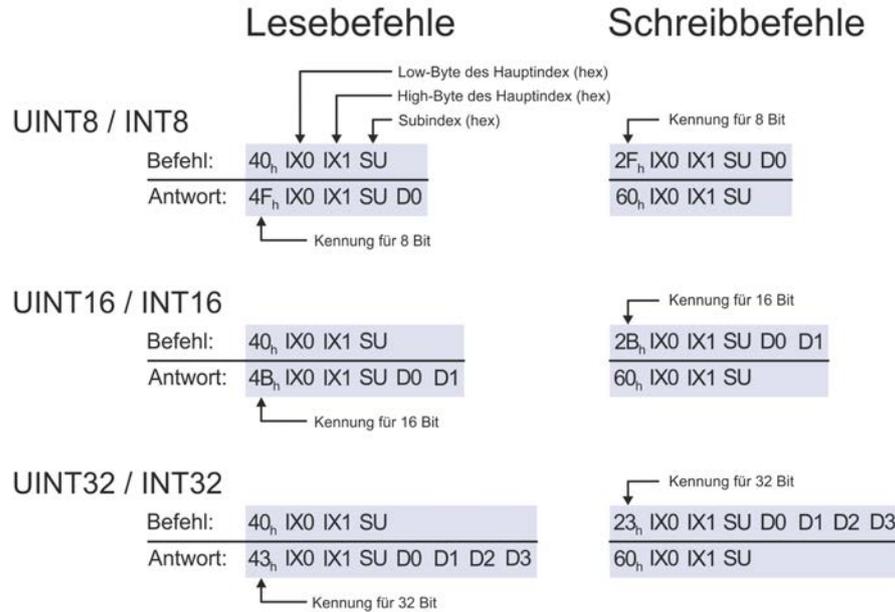
Damit der Servoregler erkennt, dass der Befehl für ihn bestimmt ist, muss der Host den Befehl mit einem bestimmten Identifier senden. **Dieser setzt sich aus der Basis 600_h + Knotennummer des betreffenden Servoreglers zusammen. Der Servoregler antwortet entsprechend mit dem Identifier 580_h + Knotennummer.**

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 ... 65536
INT16	16-Bit-Wert mit Vorzeichen	-32768 ... 32767
UINT32	32-Bit-Wert ohne Vorzeichen	0 ... $(2^{32} - 1)$
INT32	32-Bit-Wert mit Vorzeichen	$-(2^{31})$... $(2^{31} - 1)$
VISSTR	Visible String	---

6.2.1 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 Servoregler 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 Servoregler antwortet je nach zurückgegebenem Datentyp unterschiedlich. Alle Zahlen sind in hexadezimaler Schreibweise gehalten.



BEISPIEL

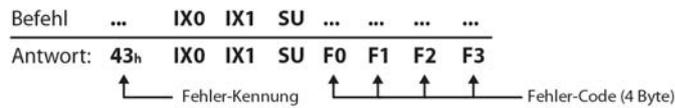
<p>UINT8 / INT8</p> <p>Lesen von Objekt 6061_h00_h Rückgabe-Daten: 01_h</p> <p>Befehl: <code>40_h 61_h 60_h 00_h</code> Antwort: <code>4F_h 61_h 60_h 00_h 01_h</code></p>	<p>Schreiben von Objekt 1401_h02_h Daten: EF_h</p> <p><code>2F_h 01_h 14_h 02_h EF_h</code> <code>60_h 01_h 14_h 02_h</code></p>
<p>UINT16 / INT16</p> <p>Lesen von Objekt 6041_h00_h Rückgabe-Daten: 1234_h</p> <p>Befehl: <code>40_h 41_h 60_h 00_h</code> Antwort: <code>4B_h 41_h 60_h 00_h 34_h 12_h</code></p>	<p>Schreiben von Objekt 6040_h00_h Daten: 03E8_h</p> <p><code>2B_h 40_h 60_h 00_h E8_h 03_h</code> <code>60_h 40_h 60_h 00_h</code></p>
<p>UINT32 / INT32</p> <p>Lesen von Objekt 6093_h01_h Rückgabe-Daten: 12345678_h</p> <p>Befehl: <code>40_h 93_h 60_h 01_h</code> Antwort: <code>43_h 93_h 60_h 01_h 78_h 56_h 34_h 12_h</code></p>	<p>Schreiben von Objekt 6093_h01_h Daten: 12345678_h</p> <p><code>23_h 93_h 60_h 01_h 78_h 56_h 34_h 12_h</code> <code>60_h 93_h 60_h 01_h</code></p>

HINWEIS Die Quittierung vom Servoregler muss abgewartet werden!

Erst wenn der Servoregler die Anforderung quittiert hat, dürfen weitere Anforderungen gesendet werden.

6.2.2 SDO-Fehlermeldungen (abort codes)

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



Fehlercode F3 F2 F1 F0	Bedeutung
05 03 00 00 _h	Protokollfehler: Toggle Bit wurde nicht geändert
05 04 00 01 _h	Protokollfehler: client / server command specifier ungültig oder unbekannt
06 01 00 00 _h	Zugriffsart wird nicht unterstützt.
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 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 (z.B. ro- Objekt in RPDO)
06 04 00 42 _h	Die Länge der in das PDO eingetragenen Objekte überschreitet die PDO-Länge
06 04 00 43 _h	Allgemeiner Parameterfehler
06 04 00 47 _h	Überlauf einer internen Größe / Genereller Fehler
06 06 00 00 _h	Zugriff fehlerhaft aufgrund eine Hardware-Problems *1)
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 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
06 09 00 36 _h	Obere Grenze ist kleiner als untere Grenze
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 Servoregler lokal arbeitet
08 00 00 22 _h	Daten können nicht übertragen oder gespeichert werden, da sich der Servoregler dafür nicht im richtigen Zustand befindet *3)
08 00 00 23 _h	Es ist kein Object Dictionary vorhanden *2)

*1) Werden gemäß DS301 bei fehlerhaftem Zugriff auf [store_parameters](#) / [restore_parameters](#) zurückgegeben.

*2) Dieser Fehler wird z.B. zurückgegeben, wenn ein anderes Bussystem den Servoregler kontrolliert oder der Parameterzugriff nicht erlaubt ist.

*3) „Zustand“ ist hier allgemein zu verstehen: Es kann sich dabei sowohl um die falsche Betriebsart handeln, als auch um ein nicht vorhandenes Technologie-Modul o.ä.

6.2.3 Simulation von SDO-Zugriffen

Die Firmware der Servoregler bietet die Möglichkeit, SDO-Zugriffe über die Parametrierschnittstelle (z.B. das Transfer-Fenster des Metronix ServoCommander®) zu simulieren. So können Objekte, die über den CAN-Bus geschrieben wurden, über die Parametrierschnittstelle gelesen und kontrolliert werden.

Die Syntax der Befehle lautet:



¹⁾ Die Antwort ist im Fehlerfall für alle 3 Schreibbefehle (8, 16, 32 Bit) gleich aufgebaut. Die Befehle werden als Zeichen ohne jegliche Leerzeichen eingegeben.

HINWEIS Testbefehle sind nicht echtzeitfähig

Der Zugriff über die Parametrierschnittstelle ist nicht für eine echtzeitfähige Kommunikation geeignet.

6.3 PDO-Message

Mit **Process-Data-Objekten** (PDOs) können Daten ereignisgesteuert übertragen werden. Das PDO überträgt dabei ausschließlich Nutzdaten. Welche Parameter übertragen werden, wird vorab zwischen Host und Servo festgelegt. Anders als bei einem SDO erfolgt bei der Übertragung eines PDOs keine Quittierung.

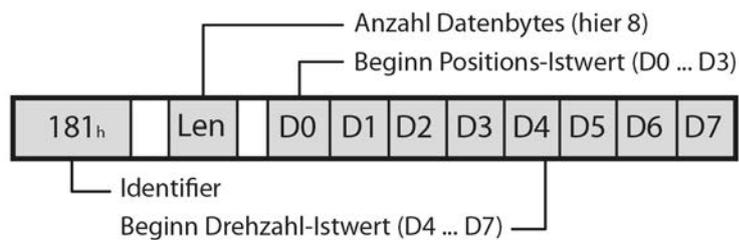
Folgende Typen von PDOs werden unterschieden:

Transmit-PDO (TPDO)	Servo → Host	Servoregler sendet PDO bei Auftreten eines bestimmten Ereignisses
Receive-PDO (RPDO)	Host → Servo	Servoregler wertet PDO bei Auftreten eines bestimmten Ereignisses aus

Der Servoregler verfügt über vier Transmit- und vier 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.

Im 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.

6.3.1 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.

Die COB-ID darf nur geändert werden, wenn das PDO deaktiviert, d.h. Bit 31 gesetzt ist. Ein anderer Identifier als aktuell im Servoregler eingestellt darf daher nur geschrieben werden, wenn gleichzeitig Bit 31 gesetzt ist.

Das gesetzte Bit 30 beim Lesen des Identifiers zeigt an, dass das Objekt nicht durch ein Remoteframe abgefragt werden kann. Dieses Bit wird beim Schreiben ignoriert und ist beim Lesen immer gesetzt.

› 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 Servoregler 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.

Der Mapping-Eintrag wird folgendermaßen zusammengesetzt:

Index (16 Bit), Subindex (8 Bit), Länge (8Bit)

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
01 _h –F0 _h	SYNC-Message Der Zahlenwert gibt an, wie viele SYNC-Nachrichten eingetroffen sein müssen, bevor das PDO <ul style="list-style-type: none"> • gesendet (T-PDO) bzw. • ausgewertet (R-PDO) wird. 	TPDOs RPDOs
FE _h	Zyklisch Das Transmit-PDO wird vom Servoregler zyklisch aktualisiert und gesendet. Die Zeitspanne wird durch das Objekt <code>inhibit_time</code> in 100µs-Schritten festgelegt. Receive-PDOs werden hingegen unmittelbar nach Empfang ausgewertet.	TPDOs (RPDOs)
FF _h	Änderung Das Transmit-PDO wird gesendet, wenn sich in den Daten des PDOs mindestens 1 Bit geändert hat. Dieser <code>transmission_type</code> ist auch für Receive-PDOs zulässig. Mit <code>inhibit_time</code> kann zusätzlich der minimale Abstand zwischen dem Absenden zweier PDOs in 100µs-Schritten festgelegt werden.	TPDOs

> 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:

Index_Subindex	Länge	Name des Objekts
6041 _h _00 _h	10 _h	statusword
6061 _h _00 _h	08 _h	modes_of_operation_display
60FD _h _00 _h	20 _h	digital_inputs

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

- PDO deaktivieren** `cob_id_used_by_pdo = C0000187h`
 Falls das PDO aktiv ist, muss es zuerst deaktiviert werden, d.h. der Identifier muss mit gesetztem Bit 31 (PDO ist deaktiviert) geschrieben werden:
- Anzahl der Objekte löschen** `number_of_mapped_objects = 0`
 Damit das Objektmapping geändert werden darf, muss die Anzahl der Objekte auf Null gesetzt werden.
- Objekte parametrieren** `first_mapped_object = 60410010h`
`second_mapped_object = 60610008h`
`third_mapped_object = 60FD0020h`
 Index und Subindex der oben aufgeführten Objekte müssen jeweils zu einem 32 Bit-Wert zusammengesetzt werden:
- Anzahl der Objekte parametrieren** `number_of_mapped_objects = 3`
 Es sollen 3 Objekte im PDO übertragen werden.
- Übertragungsart parametrieren** `transmission_type = FFh`

`transmit_mask_low = 000000FFh`
`transmit_mask_high = FFFFFFF00h`
 Das PDO soll bei Änderung der digitalen Eingänge gesendet werden.
 Damit nur die Änderung der digitalen Eingänge zum Senden führt, wird das PDO maskiert.
 Das PDO soll höchstens alle 10 ms (100x100µs) gesendet werden. `inhibit_time = 64h`
- Identifier parametrieren** `cob_id_used_by_pdo = 40000187h`
 Das PDO soll mit Identifier 187_h gesendet werden: Schreiben des Identifiers mit gelöschtem Bit 31:

HINWEIS Parametrierung der PDOs

Beachten Sie, dass die Parametrierung der PDOs generell nur geändert werden darf, wenn der Netzwerkstatus (NMT) nicht **Operational** ist. Siehe hierzu auch Abschnitt 6.6 *Netzwerkmanagement (NMT-Service)* auf Seite 190.

6.3.2 Objekte zur PDO-Parametrierung

Die einzelnen Objekte um PDOs zu parametrieren sind jeweils für alle 4 TPDOs und alle 4 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		
Type	RECORD		03 _h
Sub-Index	01_h		
Name	cob_id_used_by_pdo_tpdo1		
Info	--	rw	PDO UINT32
Value	181 _h ...1FF _h , Bit 30 und 31 dürfen gesetzt sein		C0000181 _h
Sub-Index	02_h		
Name	transmission_type_tpdo1		
Info	--	rw	PDO UINT8
Value	0...8C _h , FE _h , FF _h		FF _h
Sub-Index	03_h		
Name	inhibit_time_tpdo1		
Info	100µs (10 = 1ms)	rw	PDO UINT16
Value	--		0000 _h

Index	1A00_h		
Name	transmit_pdo_mapping_tpdo1		
Type	RECORD		04 _h
Sub-Index	00_h		
Name	number_of_mapped_objects_tpdo1		
Info	--	rw	PDO UINT8
Value	0...4		siehe Tabelle
Sub-Index	01_h		
Name	first_mapped_object_tpdo1		
Info	--	rw	PDO UINT32
Value	--		siehe Tabelle
Sub-Index	02_h		
Name	second_mapped_object_tpdo1		
Info	--	rw	PDO UINT32
Value	--		siehe Tabelle

Sub-Index	03_h			
Name	third_mapped_object_tpdo1			
Info	--	rw	PDO	UINT32
Value	--	siehe Tabelle		
Sub-Index	04_h			
Name	fourth_mapped_object_tpdo1			
Info	--	rw	PDO	UINT32
Value	--	siehe Tabelle		

HINWEIS Vor der Parametrierung muss PDO deaktiviert werden

Beachten Sie, dass die Objekt-Gruppen `transmit_pdo_parameter_xxx` und `transmit_pdo_mapping_xxx` nur beschrieben werden können, wenn das PDO deaktiviert ist (Bit 31 in `cob_id_used_by_pdo_xxx` gesetzt)

1. Transmit-PDO

Index	Comment	Type	Acc.	Default Value
1800 _{h_00h}	number of entries	UINT8	ro	03 _h
1800 _{h_01h}	COB-ID used by PDO	UINT32	rw	C0000181 _h
1800 _{h_02h}	transmission type	UINT8	rw	FF _h
1800 _{h_03h}	inhibit time (100 µs)	UINT16	rw	0000 _h
1A00 _{h_00h}	number of mapped objects	UINT8	rw	01 _h
1A00 _{h_01h}	first mapped object	UINT32	rw	60410010 _h
1A00 _{h_02h}	second mapped object	UINT32	rw	00000000 _h
1A00 _{h_04h}	fourth mapped object	UINT32	rw	00000000 _h

tpdo_1_transmit_mask

Index	Comment	Type	Acc.	Default Value
2014 _{h_00h}	number of entries	UINT8	ro	02 _h
2014 _{h_01h}	tpdo_1_transmit_mask_low	UINT32	rw	FFFFFFFF _h
2014 _{h_02h}	tpdo_1_transmit_mask_high	UINT32	rw	FFFFFFFF _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_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

3. Transmit-PDO

Index	Comment	Type	Acc.	Default Value
1802 _h _00 _h	number of entries	UINT8	ro	03 _h
1802 _h _01 _h	COB-ID used by PDO	UINT32	rw	C0000381 _h
1802 _h _02 _h	transmission type	UINT8	rw	FF _h
1802 _h _03 _h	inhibit time (100 μs)	UINT16	rw	0000 _h
1A02 _h _00 _h	number of mapped objects	UINT8	rw	02 _h
1A02 _h _01 _h	first mapped object	UINT32	rw	60410010 _h
1A02 _h _02 _h	second mapped object	UINT32	rw	60640020 _h
1A02 _h _03 _h	third mapped object	UINT32	rw	00000000 _h
1A02 _h _04 _h	fourth mapped object	UINT32	rw	00000000 _h

tpdo_3_transmit_mask

Index	Comment	Type	Acc.	Default Value
2016 _h _00 _h	number of entries	UINT8	ro	02 _h
2016 _h _01 _h	tpdo_3_transmit_mask_low	UINT32	rw	FFFFFFFF _h
2016 _h _02 _h	tpdo_3_transmit_mask_high	UINT32	rw	FFFFFFFF _h

4. Transmit-PDO

Index	Comment	Type	Acc.	Default Value
1803 _h _00 _h	number of entries	UINT8	ro	03 _h
1803 _h _01 _h	COB-ID used by PDO	UINT32	rw	C0000481 _h
1803 _h _02 _h	transmission type	UINT8	rw	FF _h
1803 _h _03 _h	inhibit time (100 µs)	UINT16	rw	0000 _h
1A03 _h _00 _h	number of mapped objects	UINT8	rw	02 _h
1A03 _h _01 _h	first mapped object	UINT32	rw	60410010 _h
1A03 _h _02 _h	second mapped object	UINT32	rw	606C0020 _h
1A03 _h _03 _h	third mapped object	UINT32	rw	00000000 _h
1A03 _h _04 _h	fourth mapped object	UINT32	rw	00000000 _h

tpdo_4_transmit_mask

Index	Comment	Type	Acc.	Default Value
2017 _h _00 _h	number of entries	UINT8	ro	02 _h
2017 _h _01 _h	tpdo_4_transmit_mask_low	UINT32	rw	FFFFFFFF _h
2017 _h _02 _h	tpdo_4_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

3. Receive PDO

Index	Comment	Type	Acc.	Default Value
1402 _h _00 _h	number of entries	UINT8	ro	02 _h
1402 _h _01 _h	COB-ID used by PDO	UINT32	rw	C0000401 _h
1402 _h _02 _h	transmission type	UINT8	rw	FF _h
1602 _h _00 _h	number of mapped objects	UINT8	rw	02 _h
1602 _h _01 _h	first mapped object	UINT32	rw	60400010 _h
1602 _h _02 _h	second mapped object	UINT32	rw	607A0020 _h
1602 _h _03 _h	third mapped object	UINT32	rw	00000000 _h
1602 _h _04 _h	fourth mapped object	UINT32	rw	00000000 _h

4. Receive PDO

Index	Comment	Type	Acc.	Default Value
1403 _{n_00} _h	number of entries	UINT8	ro	02 _h
1403 _{n_01} _h	COB-ID used by PDO	UINT32	rw	C0000501 _h
1403 _{n_02} _h	transmission type	UINT8	rw	FF _h
1603 _{n_00} _h	number of mapped objects	UINT8	rw	02 _h
1603 _{n_01} _h	first mapped object	UINT32	rw	60400010 _h
1603 _{n_02} _h	second mapped object	UINT32	rw	60FF0020 _h
1603 _{n_03} _h	third mapped object	UINT32	rw	00000000 _h
1603 _{n_04} _h	fourth mapped object	UINT32	rw	00000000 _h

6.3.3 PDOs aktivieren

Damit der Servoregler PDOs **sendet oder auswertet** müssen folgende Punkte erfüllt sein:

- Das Objekt `number_of_mapped_objects` muss ungleich Null sein.
- Im Objekt `cob_id_used_for_pdos` muss das Bit 31 gelöscht sein.
- Der Kommunikationsstatus des Servoreglers muss **Operational** sein (siehe Abschnitt 6.6 *Netzwerkmanagement (NMT-Service)* auf Seite 190)

Damit PDOs **parametriert** werden können, darf der Kommunikationsstatus des Servoreglers nicht **Operational** sein.

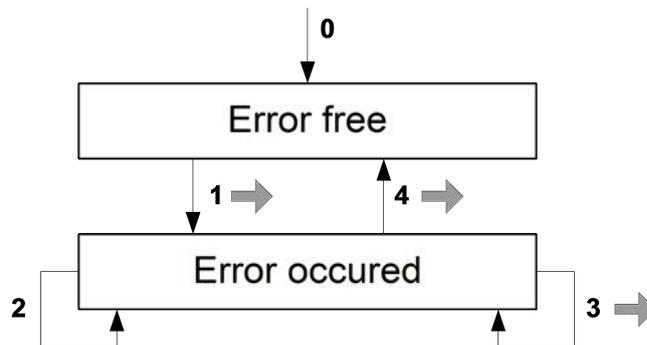
6.4 EMERGENCY-Message

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

Beim Auftreten eines Fehlers wird in der Anzeige des Servoreglers die Fehlernummer angezeigt. Wenn mehrere Fehlermeldungen gleichzeitig auftreten, so wird in der Anzeige immer die Nachricht mit der höchsten Priorität (der niedrigsten Nummer) angezeigt.

6.4.1 Übersicht

Der Servoregler sendet beim Auftreten eines Fehlers oder wenn eine Fehlerquittierung durchgeführt wird, eine EMERGENCY-Message. Der Identifier dieser Nachricht wird aus dem Identifier 80_n und der Knotennummer des betroffenen Servoreglers zusammengesetzt.

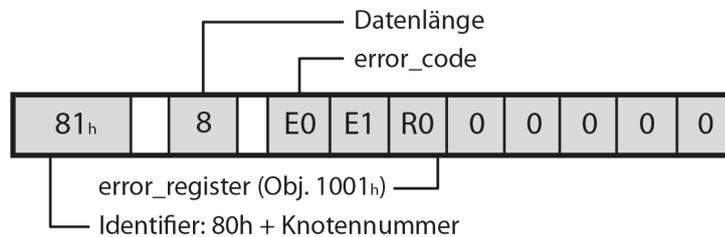


Nach einem Reset befindet sich der Servoregler im Zustand **Error free** (den er ggf. sofort wieder verlässt, weil von Anfang an ein Fehler vorhanden ist). Folgende Zustandsübergänge sind möglich:

Nr.	Ursache	Bedeutung
0	Initialisierung abgeschlossen	
1	Fehler tritt auf	Es lag kein Fehler vor und ein Fehler tritt auf. Ein EMERGENCY- Telegramm mit dem Fehlercode des aufgetretenen Fehlers wird gesendet
2	Fehlerquittierung	Eine Fehlerquittierung (siehe Abschnitt 4.3 <i>controlword (Steuernwort)</i> auf Seite 118) wird versucht, aber nicht alle Ursachen sind behoben.
3	Fehler tritt auf	Es liegt schon ein Fehler vor und ein weiterer Fehler tritt auf. Ein EMERGENCY- Telegramm mit dem Fehlercode des neuen Fehlers wird gesendet.
4	Fehlerquittierung	Eine Fehlerquittierung wird versucht und alle Ursachen sind behoben. Es wird ein EMERGENCY- Telegramm mit dem Fehlercode 0000 gesendet.

6.4.2 Aufbau der EMERGENCY-Message

Die EMERGENCY-Message besteht aus acht Datenbytes, wobei in den ersten beiden Bytes ein `error_code` steht. Im dritten Byte steht ein weiterer Fehlercode (Objekt 1001_h), der allerdings bei Metronix Servoreglern keine relevante Information enthält. Die restlichen fünf Bytes enthalten Nullen.



Eine Übersicht aller Fehlercodes, die auftreten können, finden Sie im Abschnitt 7.3 *Fehlercodes der EMERGENCY-Message* auf Seite 199

6.4.3 Beschreibung der Objekte

Objekt 1003_h: pre_defined_error_field

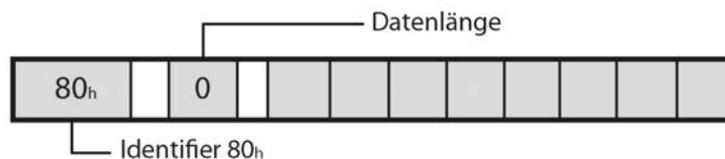
Der jeweilige `error_code` der Fehlermeldungen wird zusätzlich in einem vierstufigen Fehlerspeicher abgelegt. Dieser ist wie ein Schieberegister strukturiert, so dass immer der zuletzt aufgetretene Fehler im Objekt 1003_h01_h (`standard_error_field_0`) abgelegt ist. Durch einen Lesezugriff auf das Objekt 1003_h00_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 00h in das Objekt 1003_h00_h (`pre_defined_error_field`) gelöscht. Um nach einem Fehler die Endstufe des Servoreglers wieder aktivieren zu können, muss zusätzlich eine Fehlerquittierung (`reset_fault`, siehe Abschnitt 4.3 *controlword (Steuernwort)* auf Seite 118) durchgeführt werden.

Index	1003 _h		
Name	pre_defined_error_field		
Type	ARRAY		04 _h
Sub-Index	01 _h		
Name	standard_error_field_0		
Info	--	ro	DD UINT32
Value	--		00000000 _h
Sub-Index	02 _h		
Name	standard_error_field_1		
Info	--	ro	DD UINT32
Value	--		00000000 _h

Sub-Index	03_h		
Name	standard_error_field_2		
Info	--	ro	PDO UINT32
Value	--	00000000 _h	
Sub-Index	04_h		
Name	standard_error_field_3		
Info	--	ro	PDO UINT32
Value	--	00000000 _h	

6.5 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 Servoregler empfangen diese Nachrichten und verwenden sie für die Behandlung der PDOs (siehe Abschnitt 6.3 *PDO-Message* auf Seite 177).



Der Identifier, auf dem der Servoregler die SYNC-Message empfängt, ist fest auf 080h eingestellt. Der Identifier kann über das Objekt [cob_id_sync](#) ausgelesen werden.

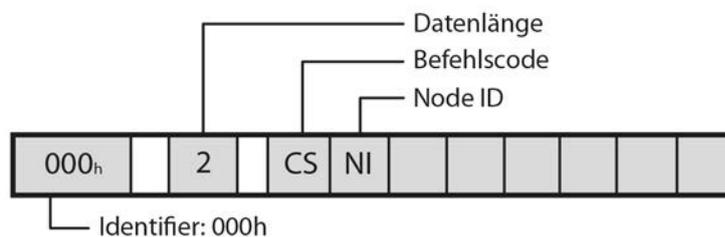
Index	1005_h		
Name	cob_id_sync		
Info	--	rw	PDO UINT32
Value	80 _h	80 _h	

6.6 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 Servoregler 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 Servoreglers beinhaltet. Wird Null als Knotenadresse angegeben, werden alle im Netzwerk befindlichen Knoten adressiert werden (Broadcast). Es ist somit möglich, dass z.B. in allen Geräten gleichzeitig ein Reset ausgelöst wird. Die Servoregler 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.

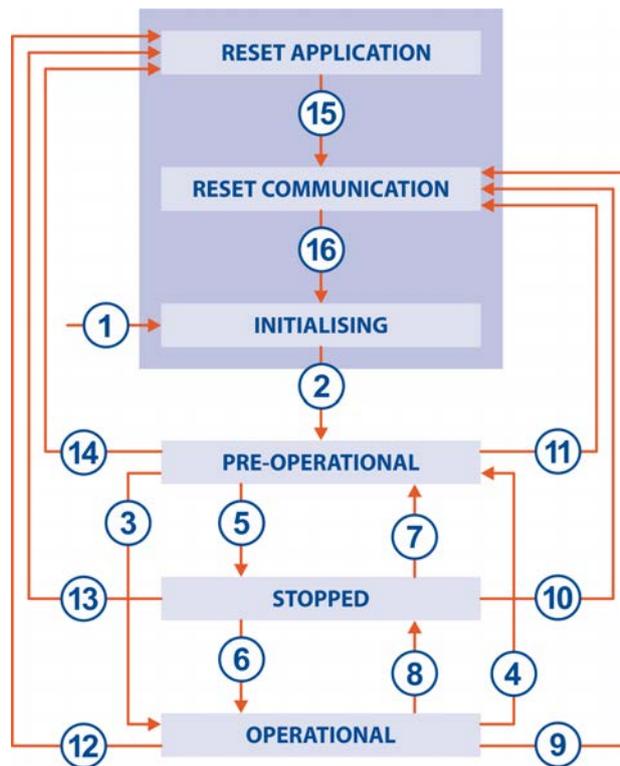


Abbildung 27: NMT-State machine

Übergang	Name	CS	Ziel-Zustand	NMT-Status
1	Power on			
2	Bootup		Pre-Operational	7F _h
3	Start Remote Node	01 _h	Operational	05 _h
4	Enter Pre-Operational	80 _h	Pre-Operational	7F _h
5	Stop Remote Node	02 _h	Stopped	04 _h
6	Start Remote Node	01 _h	Operational	05 _h
7	Enter Pre-Operational	80 _h	Pre-Operational	7F _h
8	Stop Remote Node	02 _h	Stopped	04 _h
9	Reset Communication	82 _h	Pre-Operational	7F _h
10	Reset Communication	82 _h	Pre-Operational	7F _h
11	Reset Communication	82 _h	Pre-Operational	7F _h
12	Reset Application	81 _h	Pre-Operational	7F _h
13	Reset Application	81 _h	Pre-Operational	7F _h
14	Reset Application	81 _h	Pre-Operational	7F _h

Die Zustands-Übergänge 2, 15 und 16 werden vom Servoregler selbsttätig ausgeführt, wenn die Initialisierung abgeschlossen ist.

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 Servoregler PDOs sendet.

Zustand	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

HINWEIS Beachten Sie die folgenden Hinweise

- NMT- Telegramme dürfen nicht in einem Burst (unmittelbar hintereinander) gesendet werden.
- Zwischen zwei aufeinanderfolgenden NMT- Nachrichten auf dem Bus (auch für verschiedene Knoten!) muss mindestens die doppelte Lagereglerzykluszeit liegen, damit der Servoregler die NMT- Nachrichten korrekt verarbeitet.
- Der NMT Befehl „Reset Application“ wird gegebenenfalls so lange verzögert, bis ein laufender Speichervorgang abgeschlossen ist, da ansonsten der Speichervorgang unvollständig bleiben würde (Defekter Parametersatz). Die Verzögerung kann im Bereich einiger Sekunden liegen.
- Der Kommunikationsstatus muss auf **Operational** eingestellt werden, damit der Servoregler PDOs sendet und empfängt.

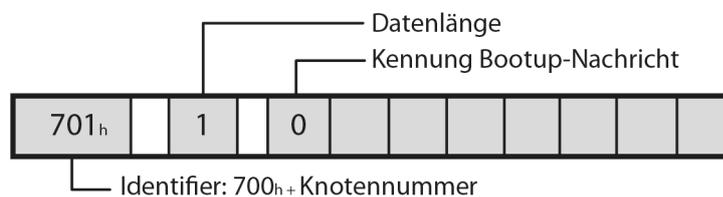
6.7 Bootup

6.7.1 Übersicht

Nach dem Einschalten der Spannungsversorgung oder nach einem Reset, meldet der Servoregler über eine Bootup-Nachricht, dass die Initialisierungsphase beendet ist. Der Servoregler ist dann im NMT-Status [Pre-Operational](#).

6.7.2 Aufbau der Bootup- Nachricht

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



6.8 Heartbeat (Error Control Protocol)

6.8.1 Übersicht

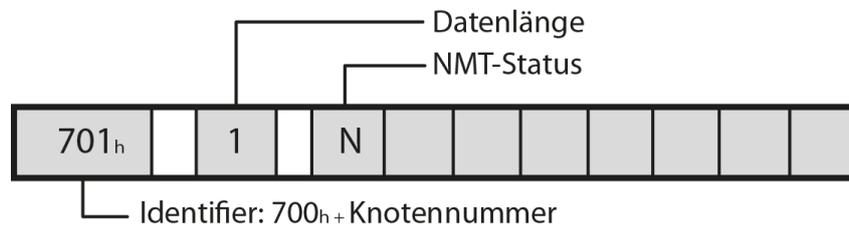
Zur Überwachung der Kommunikation zwischen Slave (Antrieb) und Master kann das sogenannte Heartbeat-Protokoll aktiviert werden: Hierbei sendet der Antrieb zyklisch Nachrichten an den Master. Der Master kann das zyklische Auftreten dieser Nachrichten überprüfen und entsprechende Maßnahmen einleiten, wenn diese ausbleiben.

Da sowohl Heartbeat- als auch Nodeguarding- Telegramme (siehe Abschnitt 6.9 *Nodeguarding (Error Control Protocol)* auf Seite 195) mit dem Identifier

700_h + Knotennummer gesendet werden, können nicht beide Protokolle gleichzeitig aktiv sein. Werden beide Protokolle gleichzeitig aktiviert, ist nur das Heartbeat- Protokoll aktiv.

6.8.2 Aufbau der Heartbeat- Nachricht

Das Heartbeat-Telegramm wird mit dem Identifier **700_h + Knotennummer** gesendet. Es enthält nur 1 Byte Nutzdaten, den NMT-Status des Servoreglers (siehe Abschnitt 6.6 *Netzwerkmanagement (NMT-Service)* auf Seite 190).



NMT-Status	Zustand
04 _h	Stopped
05 _h	Operational
7F _h	Pre-Operational

6.8.3 Beschreibung der Objekte

Objekt 1017_h: producer_heartbeat_time

Zur Aktivierung der Heartbeat- Funktionalität kann die Zeit zwischen zwei Heartbeat-Telegrammen über das Objekt `producer_heartbeat_time` festgelegt werden.

Index	1017_h		
Name	producer_heartbeat_time		
Info	ms	rw	DD UINT16
Value	0...65536		0

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

Der Servoregler kann nur als Heartbeat Producer verwendet werden. Das Objekt 1016_h (`consumer_heartbeat_time`) ist daher nur aus Kompatibilitätsgründen implementiert und liefert immer 0 zurück.

6.9 Nodeguarding (Error Control Protocol)

6.9.1 Übersicht

Ebenfalls zur Überwachung der Kommunikation zwischen Slave (Antrieb) und Master kann das sogenannte Nodeguarding-Protokoll verwendet werden. Im Gegensatz zum Heartbeat-Protokoll überwachen sich hierbei Master und Slave gegenseitig:

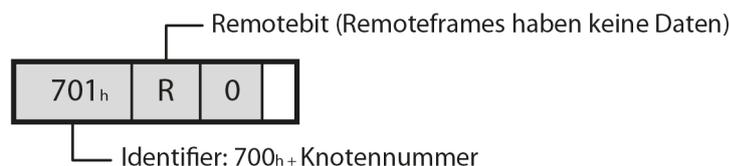
Der Master fragt den Antrieb zyklisch nach seinem NMT-Status. Dabei wird in jeder Antwort des Servoreglers ein bestimmtes Bit invertiert (getoggelt). Bleiben diese Antworten aus oder antwortet der Servoregler immer mit dem gleichen Togglebit, kann der Master entsprechend reagieren.

Ebenso überwacht der Antrieb das regelmäßige Eintreffen der Nodeguarding-Anfragen des Masters: Bleiben die Nachrichten über einen bestimmten Zeitraum aus, löst der Servoregler Fehler 12-4 aus.

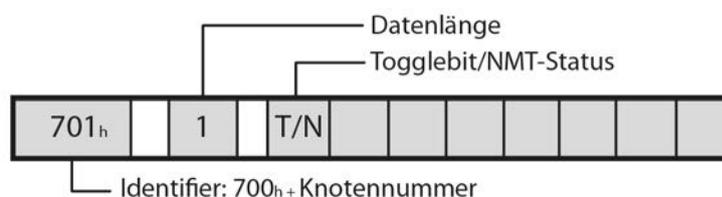
Da sowohl Heartbeat- als auch Nodeguarding-Telegramme (siehe Abschnitt 6.8 *Heartbeat (Error Control Protocol)* auf Seite 193) mit dem Identifier **700_h + Knotennummer** gesendet werden, können nicht beide Protokolle gleichzeitig aktiv sein. Werden beide Protokolle gleichzeitig aktiviert, ist nur das Heartbeat-Protokoll aktiv.

6.9.2 Aufbau der Nodeguarding-Nachrichten

Die Anfrage des Masters muss als Remoteframe mit dem Identifier **700_h + Knotennummer** gesendet werden. Bei einem Remoteframe ist zusätzlich ein spezielles Bit im Telegramm gesetzt, das Remotebit. Remoteframes haben grundsätzlich keine Daten.



Die Antwort des Servoreglers ist analog zur Heartbeat-Nachricht aufgebaut. Sie enthält nur 1 Byte Nutzdaten, das Togglebit und den NMT-Status des Servoreglers (siehe Abschnitt 6.6 *Netzwerkmanagement (NMT-Service)* auf Seite 190).



Die Überwachungszeit für Anfragen des Masters ist parametrierbar. Die Überwachung beginnt mit der ersten empfangenen Remoteabfrage des Masters. Ab diesem Zeitpunkt

müssen die Remoteabfragen vor Ablauf der eingestellten Überwachungszeit eintreffen, da anderenfalls Fehler 12-4 ausgelöst wird.

Das Togglebit wird durch das NMT-Kommando [Reset Communication](#) zurückgesetzt. Es ist daher in der ersten Antwort des Servoreglers gelöscht.

6.9.3 Beschreibung der Objekte

6.9.3.1 Objekt 100C_h: guard_time

Zur Aktivierung der Nodeguarding-Überwachung wird die Maximalzeit zwischen zwei Remoteabfragen des Masters parametrisiert. Diese Zeit wird im Servoregler aus dem Produkt von [guard_time](#) (100C_h) und [life_time_factor](#) (100D_h) bestimmt. Es empfiehlt sich daher den [life_time_factor](#) mit 1 zu beschreiben und die Zeit dann direkt über die [guard_time](#) in Millisekunden vorzugeben.

Index	100C_h			
Name	guard_time			
Info	ms	rw	PBO	UINT16
Value	0...65536		0	

6.9.3.2 Objekt 100D_h: life_time_factor

Der [life_time_factor](#) sollte mit 1 beschrieben werden um die [guard_time](#) direkt vorzugeben.

Index	100D_h			
Name	life_time_factor			
Info	--	rw	PBO	UINT8
Value	0...1		0	

6.10 Tabelle der Identifier

Die folgende Tabelle gibt eine Übersicht über die verwendeten Identifier:

Objekt-Typ	Identifier (hexadezimal)	Bemerkung
SDO (Host an Servo)	600 _h + Knotennummer	
SDO (Servo an Host)	580 _h + Knotennummer	
TPDO1	181 _h / 180 _h + Knotennummer	Angegeben sind die Defaultwerte. Die Knotennummer kann automatisch addiert werden, wenn die entsprechende Option gesetzt ist (siehe Abschnitt 2.1.5 <i>CANopen aktivieren</i> auf Seite 20).
TPDO2	281 _h / 280 _h + Knotennummer	
TPDO3	381 _h / 380 _h + Knotennummer	
TPDO4	481 _h / 480 _h + Knotennummer	
RPDO1	201 _h / 200 _h + Knotennummer	
RPDO2	301 _h / 300 _h + Knotennummer	
RPDO3	401 _h / 400 _h + Knotennummer	
RPDO4	501 _h / 500 _h + Knotennummer	
SYNC	080 _h	
EMCY	080 _h + Knotennummer	
HEARTBEAT	700 _h + Knotennummer	
NODEGUARDING	700 _h + Knotennummer	
BOOTUP	700 _h + Knotennummer	
NMT	000 _h	

7 Anhang

7.1 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 die Metronix Servoreglerfamilien benötigten Elemente des Objektverzeichnisses und die zugehörigen Zugriffsmethoden sind im vorliegenden CANopen-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 Servoreglern. 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)
Kontumazgarten 3
DE-90429 Nürnberg
Tel.: +49-911-928819-0
Fax: +49-911-928819-79

[headquarters\(at\)can-cia.org](mailto:headquarters(at)can-cia.org)
www.can-cia.de

Der CANopen- Implementierung des Servoreglers liegen folgende Normen zugrunde:

- CiA Draft Standard 301, Version 4.02, 13. Februar 2002
- CiA Draft Standard Proposal 402, Version 2.0, 26. Juli 2002

7.2 Kenndaten des CAN-Interface

Das CAN-Interface besitzt folgende Leistungsmerkmale:

- CAN-Spezifikation V2.0 Teil A (Teil B passiv, d. h. Nachrichten dieser Art werden toleriert, aber nicht verarbeitet)
- Physical layer: ISO 11898

7.3 Fehlercodes der EMERGENCY-Message

error_code	Anzeige	Bedeutung
2300 _h	31-x	Gruppe 31: I²t
2311 _h	31-1	I²t-Servoregler
2312 _h	31-0	I²t-Motor
2313 _h	31-2	I²t-PFC
2314 _h	31-3	I²t-Bremswiderstand
2320 _h	6-x	Gruppe 6: Kurzschluss Endstufe
3200 _h	32-x	Gruppe 32: PFC
3210 _h	7-x	Gruppe 7: Überspannung
3220 _h	2-x	Gruppe 2: Unterspannung im Zwischenkreis
3280 _h	32-0	Ladezeit Zwischenkreis überschritten
3281 _h	32-1	Unterspannung für aktive PFC
3282 _h	32-5	Überlast Bremschopper. Zwischenkreis konnte nicht entladen werden
3283 _h	32-6	Entladezeit Zwischenkreis überschritten
3284 _h	32-7	Leistungsversorgung für Reglerfreigabe fehlt
3285 _h	32-8	Ausfall Leistungsversorgung bei freigegebenem Servoregler
3286 _h	32-9	Phasenausfall
4200 _h	4-x	Gruppe 4: Übertemperatur
4210 _h	4-0	Übertemperatur Leistungsteil
4280 _h	4-1	Übertemperatur Zwischenkreis
4310 _h	3-x	Gruppe 3: Übertemperatur Motor
5080 _h	90-x	Gruppe 90: HW-Initialisierung
5110 _h	5-x	Gruppe 5: Interne Spannungsversorgung
5114 _h	5-0	Ausfall interne Spannung 1
5115 _h	5-1	Ausfall interne Spannung 2
5116 _h	5-2	Ausfall Treiberversorgung
5200 _h	21-x	Gruppe 21: Strommessung
5220 _h	16-4	Unerwarteter Hardware-Fehler
5280 _h	21-0	Fehler 1 Strommessung U
5281 _h	21-1	Fehler 1 Strommessung V
5282 _h	21-2	Fehler 2 Strommessung U
5283 _h	21-3	Fehler 2 Strommessung V
5410 _h	5-3	Unterspannung digitale I/Os
5410 _h	5-4	Überstrom digitale I/Os

error_code	Anzeige	Bedeutung
5430 _h	24-x	Gruppe 24: Überwachung Analogeingang
5500 _h	26-x	Gruppe 26: Flash
5580 _h	26-0	Fehlender User-Parametersatz
5581 _h	26-1	Checksummenfehler
5582 _h	26-2	Flash: Fehler beim Schreiben
5583 _h	26-3	Flash: Fehler beim Löschen
5584 _h	26-4	Flash: Fehler im internen Flash
5585 _h	26-5	Fehlende Kalibrierdaten
5586 _h	26-6	Fehlende User-Positionsdatensätze
6000 _h	25-x	Gruppe 25: Ungültiger Gerätetyp
6000 _h	91-x	Gruppe 91: SW-Initialisierung
6080 _h	25-0	Ungültiger Gerätetyp
6081 _h	25-1	Gerätetyp nicht unterstützt
6082 _h	25-2	Hardware-Revision nicht unterstützt
6083 _h	25-3	Gerätefunktion beschränkt
6100 _h	16-x	Gruppe 16: Programmablauf
6180 _h	1-x	Gruppe 1: Stacküberlauf
6181 _h	16-0	Programmausführung fehlerhaft
6182 _h	16-1	Illegaler Interrupt
6183 _h	16-3	Unerwarteter Zustand
6184 _h	15-x	Gruppe 15: Mathematik
6185 _h	15-0	Division durch Null
6186 _h	15-1	Bereichsüberschreitung
6187 _h	16-2	Initialisierungsfehler
6188 _h	82-x	Gruppe 82: Interne Ablaufsteuerung
6320 _h	36-x	Gruppe 36: Parameter
6380 _h	30-x	Gruppe 30: Interne Berechnungen
7122 _h	14-x	Gruppe 14: Motor- und Winkelgeber-Identifikation
7300 _h	8-x	Gruppe 8: Winkelgeber
7380 _h	8-0	Winkelgeberfehler Resolver/Hallgeber
7382 _h	8-2	Fehler Spursignale Z0 Inkrementalgeber
7383 _h	8-3	Fehler Spursignale Z1 Inkrementalgeber
7384 _h	8-4	Fehler Spursignale digitaler Inkrementalgeber
7385 _h	8-5	Fehler Hallgebersignale Inkrementalgeber
7386 _h	8-6	Kommunikationsfehler Winkelgeber

error_code	Anzeige	Bedeutung
7387 _h	8-7	Leitfrequenzeingang: Signalamplitude Inkrementalspur fehlerhaft
7388 _h	8-8	Interner Winkelgeberfehler
7389 _h	8-9	Winkelgeber an [X2B/X6] wird nicht unterstützt
73A0 _h	9-x	Gruppe 9: Winkelgeber-Parametersatz
73A1 _h	9-0	Winkelgeber-Parametersatz: veraltetes Format
73A2 _h	9-1	Winkelgeber-Parametersatz kann nicht dekodiert werden
73A3 _h	9-2	Winkelgeber-Parametersatz: unbekannte Version
73A4 _h	9-3	Winkelgeber-Parametersatz: defekte Datenstruktur
73A5 _h	9-7	Schreibgeschütztes EEPROM Winkelgeber
73A6 _h	9-9	EEPROM Winkelgeber zu klein
7580 _h	60-x	Gruppe 60: Ethernet
7581 _h	61-x	Gruppe 61: Ethernet
8000 _h	45-x	Gruppe 45: Treiberversorgung IGBT
8080 _h	43-x	Gruppe 43: HW-Endschalter
8081 _h	43-0	Endschalter: Negativer Sollwert gesperrt
8082 _h	43-1	Endschalter: Positiver Sollwert gesperrt
8083 _h	43-2	Endschalter: Positionierung unterdrückt
8084 _h	45-0	Treiberversorgung nicht abschaltbar
8085 _h	45-1	Treiberversorgung nicht aktivierbar
8086 _h	45-2	Treiberversorgung wurde aktiviert
8090 _h	51-x	Gruppe 51: FSM 2.0
8091 _h	51-0	Kein / unbekanntes FSM-Modul oder Treiberversorgung fehlerhaft
8093 _h	51-2	FSM: Ungleicher Modultyp
8094 _h	51-3	FSM: Ungleiche Modulversion
8095 _h	51-4	FSM: Fehler in der SSIO-Kommunikation
8096 _h	51-5	FSM: Fehler in der Bremsenansteuerung
8097 _h	51-6	FSM: Ungleiche Modul-Seriennummer
8098 _h	52-x	Gruppe 52: FSM 2.0 STO
8099 _h	52-1	FSM: Diskrepanzzeit abgelaufen
809A _h	52-2	FSM: Ausfall STOA/STOB bei freigegebener Endstufe
809B _h	52-3	FSM: Fehler in den Begrenzungen
80A0 _h	53-x	Gruppe 53: FSM: Verletzung von Sicherheitsbedingungen
80A1 _h	53-0	USF0: Sicherheitsbedingung verletzt
80A2 _h	53-1	USF1: Sicherheitsbedingung verletzt
80A3 _h	53-2	USF2: Sicherheitsbedingung verletzt

error_code	Anzeige	Bedeutung
80A4 _h	53-3	USF3: Sicherheitsbedingung verletzt
80A9 _h	54-x	Gruppe 54: FSM: Verletzung von Sicherheitsbedingungen
80AA _h	54-0	SBC: Sicherheitsbedingung verletzt
80AC _h	54-2	SS2: Sicherheitsbedingung verletzt
80AD _h	54-3	SOS: Sicherheitsbedingung verletzt
80AE _h	54-4	SS1: Sicherheitsbedingung verletzt
80AF _h	54-5	STO: Sicherheitsbedingung verletzt
80B0 _h	54-6	SBC: Bremse > 24 h nicht gelüftet
80B1 _h	54-7	SOS: SOS > 24 h angefordert
80C0 _h	55-x	Gruppe 55: FSM: Istwerterfassung 1
80C1 _h	55-0	FSM: Kein Drehzahl-/Positionsistwert verfügbar oder Stillstand > 24 h
80C2 _h	55-1	FSM: SINCOS-Geber [X2B] - Fehler Spursignale
80C3 _h	55-2	FSM: SINCOS-Geber [X2B] - Stillstand > 24 h
80C4 _h	55-3	FSM: Resolver [X2A] - Signalfehler
80C6 _h	55-7	FSM: Sonstiger Geber [X2B] - Fehlerhafte Winkelinformation
80C7 _h	55-8	FSM: Unzulässige Beschleunigung detektiert
80D0 _h	56-x	Gruppe 56: FSM: Istwerterfassung 2
80D1 _h	56-8	FSM: Drehzahl- / Winkeldifferenz Geber 1 - 2
80D2 _h	56-9	FSM: Fehler Kreuzvergleich Geberauswertung
80E0 _h	57-x	Gruppe 57: FSM: Ein-/Ausgänge
80E1 _h	57-0	FSM: E/A - Fehler Selbsttest (intern/extern)
80E2 _h	57-1	FSM: Digitale Eingänge - Fehler Signalpegel
80E3 _h	57-2	FSM: Digitale Eingänge - Fehler Testimpuls
80E7 _h	57-6	FSM: Übertemperatur
80E8 _h	58-x	Gruppe 58: FSM: Kommunikation / Parametrierung
80E9 _h	58-0	FSM: Plausibilitätsprüfung Parameter
80EA _h	58-1	FSM: Allgemeiner Fehler Parametrierung
80ED _h	58-4	FSM: Puffer interne Kommunikation
80EE _h	58-5	FSM: Kommunikation Sicherheitsmodul - Servoregler
80EF _h	58-6	FSM: Fehler Kreuzvergleich Prozessoren 1 - 2
80F0 _h	59-x	Gruppe 59: FSM: Interne Fehler
80F1 _h	59-1	FSM: Failsafe-Versorgung / sichere Impulssperre
80F2 _h	59-2	FSM: Fehler externe Spannungsversorgung
80F3 _h	59-3	FSM: Fehler interne Spannungsversorgung

error_code	Anzeige	Bedeutung
80F4 _h	59-4	FSM: Fehlermanagement: Zu viele Fehler
80F5 _h	59-5	FSM: Fehler beim Schreiben in den permanenten Ereignisspeicher
80F6 _h	59-6	FSM: Fehler beim Speichern des Parametersatzes
80F7 _h	59-7	FSM: Flash-Checksummenfehler
80F8 _h	59-8	FSM: Interne Überwachung Prozessor 1 - 2
80F9 _h	59-9	FSM: Sonstiger unerwarteter Fehler
8100 _h	12-x	Gruppe 12: CAN-Kommunikation
8100 _h	13-x	Gruppe 13: Timeout CAN-Bus
8120 _h	12-1	CAN: Kommunikationsfehler, Bus AUS
8130 _h	12-4	CAN: Node Guarding
8180 _h	12-0	CAN: Knotennummer doppelt
8181 _h	12-2	CAN: Kommunikationsfehler beim Senden
8182 _h	12-3	CAN: Kommunikationsfehler beim Empfangen
8183 _h	12-9	CAN: Protokollfehler
8184 _h	13-0	Timeout CAN-Bus
8200 _h	50-x	Gruppe 50: CAN-Kommunikation
8210 _h	12-5	CAN: RPDO zu kurz
8480 _h	35-x	Gruppe 35: Linearmotor
8600 _h	42-x	Gruppe 42: Positionierung
8611 _h	17-x	Gruppe 17: Überschreitung Grenzwert Schleppfehler
8611 _h	27-x	Gruppe 27: Schleppfehlerüberwachung
8612 _h	40-x	Gruppe 40: SW-Endschalter
8680 _h	42-0	Positionierung: Fehlende Anschlusspositionierung: Stopp
8681 _h	42-1	Positionierung: Drehrichtungsumkehr nicht erlaubt: Stopp
8682 _h	42-2	Positionierung: Drehrichtungsumkehr nach Halt nicht erlaubt
8700 _h	34-x	Gruppe 34: Feldbus
8780 _h	34-0	Keine Synchronisation über Feldbus
8781 _h	34-1	Synchronisationsfehler Feldbus
8A00 _h	11-x	Gruppe 11: Referenzfahrt
8A00 _h	33-x	Gruppe 33: Schleppfehler Encoderemulation
8A80 _h	11-0	Fehler beim Start der Referenzfahrt
8A81 _h	11-1	Fehler während der Referenzfahrt
8A82 _h	11-2	Referenzfahrt: Kein gültiger Nullimpuls
8A83 _h	11-3	Referenzfahrt: Zeitüberschreitung
8A84 _h	11-4	Referenzfahrt: falscher / ungültiger Endschalter

error_code	Anzeige	Bedeutung
8A85 _h	11-5	Referenzfahrt: I ^t / Schleppfehler
8A86 _h	11-6	Referenzfahrt: Ende der Suchstrecke
8A87 _h	33-0	Schleppfehler Encoderemulation
F000 _h	80-x	Gruppe 80: IRQ_0_3
F080 _h	80-0	Überlauf Stromregler IRQ
F081 _h	80-1	Überlauf Drehzahlregler IRQ
F082 _h	80-2	Überlauf Lageregler IRQ
F083 _h	80-3	Überlauf Interpolator IRQ
F084 _h	81-4	Überlauf Low-Level IRQ
F085 _h	81-5	Überlauf MDC IRQ
FF00 _h	28-x	Gruppe 28: Betriebsstundenzähler
FF01 _h	28-0	Betriebsstundenzähler fehlt
FF02 _h	28-1	Betriebsstundenzähler: Schreibfehler
FF03 _h	28-2	Betriebsstundenzähler korrigiert
FF04 _h	28-3	Betriebsstundenzähler konvertiert