SIEB & MEYER

Antriebsverstärker SD2x

Modbus-RTU-Anbindung

D-00000348.2
2022-02-18

SIEB & MEYER weltweit

Bei Fragen zu unseren Produkten oder technischen Rückfragen wenden Sie sich bitte an uns.

SIEB & MEYER AG
Auf dem Schmaarkamp 21
21339 Lüneburg
Deutschland

Tel.: +49 4131 203 0
Fax: +49 4131 203 2000
info@sieb-meyer.de
http://www.sieb-meyer.de

SIEB & MEYER Shenzhen Trading Co. Ltd.
Room A208 2/F,
Internet Innovation and Creation services base Building (2),
No.126, Wanxia road, Shekou, Nanshan district,
Shenzhen City, 518067
P.R. China

Tel.: +86 755 2681 1417 / +86 755 2681 2487
Fax: +86 755 2681 2967
info@sieb-meyer.cn
http://www.sieb-meyer.cn

SIEB & MEYER Asia Co. Ltd.
4 Fl, No. 532, Sec. 1
Min-Sheng N. Road
Kwei-Shan Hsiang
333 Tao-Yuan Hsien
Taiwan

Tel.: +886 3 311 5560
Fax: +886 3 322 1224
info@sieb-meyer.tw

1 Einleitung....4

2 Parametrierung....5
2.1 Antriebssteuerung....5
2.2 Bussystem....6
2.2.1 Baudrate....6
2.2.2 Slave-Adresse....6
2.2.3 Parity-Check....8

3 Funktionscodes....9
3.1 FC 03 Read Input Register / FC 04 Read Holding Register....9
3.2 FC 06 Write Single Register....10
3.3 FC 08 Diagnostics....10
3.4 FC 16 Write Multiple Registers....12
3.5 FC 23 Read/Write Multiple Registers....12
3.6 FC 43 Encapsulated Interface Transport....13
3.7 Fehlercodes für die Kommunikation....16
3.8 Erweiterte Fehlercodes (SDO-Fehler)....16

4 Prozessdaten....18
4.1 Default-RxPDO....18
4.2 Default-TxPDO....18
4.3 Bytereihenfolge der Prozessdaten....18

5 Objektverzeichnis....20
Control Word (Steuerwort)....20
Target Velocity (Drehzahlsollwert)....20
Torque Limit Iq (Strombegrenzung)....21
Watchdog....21
Status Word (Statuswort)....21
Actual Speed (Drehzahlistwert)....22
Actual Apparent Current (Istwert Scheinstrom)....22
Actual Output Power (Istwert Ausgangsleistung)....22
Error Code (Gespeicherte Fehlermeldung)....23
Temperature Power Stage (Temp. Leistungsendstufe)....23
Temperature Motor (Motortemperatur)....23
Output Voltage (Ausgangsspannung)....24
DC Voltage (Zwischenkreisspannung)....24
Hardware ID....24
Hardware Version....25
Software ID....25
Software Version....25
Max Iq Current (Max. Strom)....26
Max Speed (Max. Drehzahl)....26
Speed Scaling (Drehzahlskalierung)....26
Current Scaling (Stromskalierung)....26

6 Beispiele....28
6.1 Beispiel 1: Antrieb verfahren....28
6.2 Beispiel 2: Encapsulated Interface Transport....30
6.3 Beispiel 3: CRC berechnen....32

1 Einleitung

Dieses Dokument beschreibt die Anbindung eines SD2x-Antriebs an das Modbus-RTU-Protokoll. Modbus ASCII ist zurzeit nicht in die Geräteserie SD2x implementiert.

Für die Datenübertragung über Modbus RTU wird die RS485-Schnittstelle des SD2x-Antriebs verwendet. Innerhalb des RS485-Netzwerks kommuniziert der Antrieb als Slave.

Informationen zur Geräteserie SD2x finden Sie in der entsprechenden Hardware- bzw. Softwaredokumentation.

In den folgenden Kapiteln finden Sie Informationen zur Parametrierung des Antriebs, den Funktionscodes für die Kommunikation, den Prozessdatenobjekten und den Antriebsobjekten sowie einigen Beispielen.

Weitere Informationen zu Modbus finden Sie auf der Website www.modbus.org.

Hinweis

Wenn Sie eigene I/O-Daten konfigurieren möchten, muss der Modbus-Master die Funktion „CAN Encapsulation“ unterstützen. damit die Parameter unabhängig vom Prozessabbild gelesen und geschrieben werden können. Sollte der Master diese Funktion nicht unterstützen, können Sie nur die Default-Prozessdatenobjekte mit den Standard-Modbus-Funktionscodes übertragen.

2 Parametrierung

In den folgenden Kapiteln ist beschrieben, wie Sie Ihren SD2x-Antrieb mit Hilfe der Software *drivemaster2* für die Modbus-Kommunikation parametrieren.

2.1 Antriebssteuerung

Aktivieren Sie die Modbus-Kommunikation, indem Sie auf der Parameterseite „Antriebssteuerung“ den Steuerkanal und den Sollwertkanal auf „Modbus“ setzen:

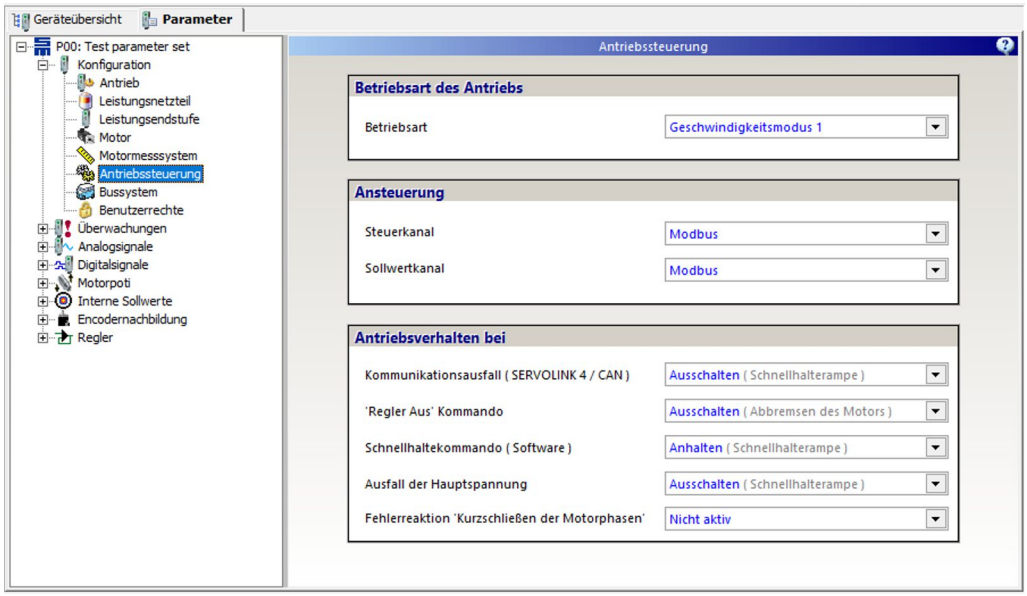

Abb. 1: Parametrierung der Antriebssteuerung

Hinweis

Wenn die Modbus-Kommunikation aktiviert ist, können Sie die RS485-Schnittstelle Ihres SD2x-Antriebs (Anschluss X74) nicht mehr für die Parametrierung des Antriebs nutzen. Diese Schnittstelle dient dann ausschließlich zur Modbus-Kommunikation.

Informationen zur Pinbelegung des Anschlusses X74 finden Sie in der Hardwarebeschreibung.

Nachdem Sie die Modbus-Kommunikation in der Ansteuerung aktiviert haben, können Sie die Baudrate und Parity-Einstellungen der Schnittstelle parametrieren, wie in den nachfolgenden Kapiteln beschrieben.

2.2 Bussystem

Auf der Parameterseite „Bussystem“ können Sie die Modbus-Kommunikation parametrieren:

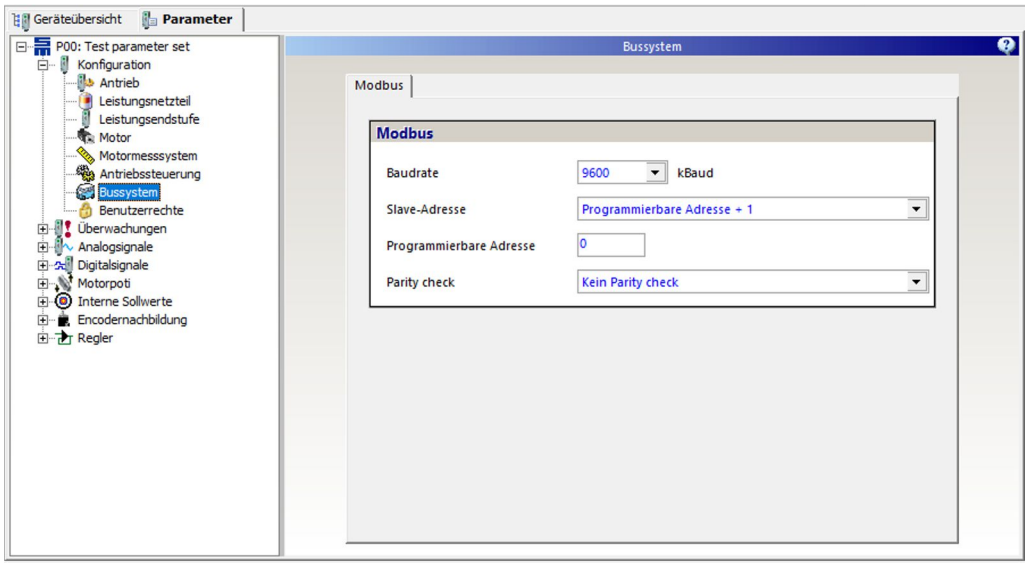

Abb. 2: Parametrierung der Modbus-Kommunikation

2.2.1 Baudrate

Sie können die folgenden Werte für die Baudrate einstellen:

- 9600 kBaud
- 14400 kBaud
- 19200 kBaud
- 28800 kBaud
- 38400 kBaud
- 57600 kBaud
- 115200 kBaud

2.2.2 Slave-Adresse

Die Slave-Adresse ist die Adresse, die dem SD2x-Antrieb im Bussystem zugewiesen ist.

Über den Parameter können Sie einstellen, wie die Slave-Adresse ermittelt werden soll. Die folgenden Einstellungen stehen zur Verfügung:

- Adresswahlschalter + 1
- Adresswahlschalter + programmierbare Adresse + 1
- Programmierbare Adresse + 1

Adresswahlschalter + 1

Die Slave-Adresse wird über den ID-Schalter auf der Frontplatte des Geräts eingestellt. Da sich am ID-Schalter Werte zwischen 0..15 einstellen lassen, im Modbus-Protokoll die 0 aber nicht zulässig ist, wird automatisch eine 1 aufaddiert. Zudem benötigen doppelachsige Geräte der SD2-Serie zwei Adressen – eine Adresse pro Achse. Deshalb wird der Wert des ID-Schalters zuvor mit 2 multipliziert. Einachsige Geräte sowie die A-

Achsen der doppelachsigen Gerät bekommen dann die ungeraden Slave-Adressen *(ID-Schalter × 2 + 1)* und die B-Achsen der doppelachsigen Geräte bekommen die geraden Slave-Adressen *(ID-Schalter × 2 + 2)*.

Die folgende Tabelle zeigt die Modbus-Adressen je nach Einstellung des ID-Schalters:

Einstellung des ID-Schalters	Modbus-Adresse	
	Einachsige Geräte	Doppelachsige Geräte
0	1	1 und 2
1	3	3 und 4
2	5	5 und 6
3	7	7 und 8
4	9	9 und 10
5	11	11 und 12
6	13	13 und 14
7	15	15 und 16
8	17	17 und 18
9	19	19 und 20
A	21	21 und 22
B	23	23 und 24
C	25	25 und 26
D	27	27 und 28
E	29	29 und 30
F	31	31 und 32

Beispiel:

Der ID-Schalter am Gerät ist auf Position 5 eingestellt.

Für ein einachsiges Gerät ergibt sich daraus die Modbus-Adresse 11.

Adresswahlschalter + programmierbare Adresse + 1

Die Slave-Adresse wird aus der Summe des ID-Schalter-Werts und dem in das Feld „Programmierbare Adresse“ eingegebenen Wert berechnet. Um den Wertebereich der Modbus-ID einzuhalten, muss hier ein Wert von 0..253 eingegeben werden. Da im Modbus-Protokoll die 0 nicht zulässig ist, wird automatisch eine 1 auf das Ergebnis aufaddiert.

Beispiel:

Der ID-Schalter am Gerät ist auf Position 5 eingestellt.

Für ein einachsiges Gerät ergibt sich daraus die Modbus-Adresse 11 + 3 = 14.

Programmierbare Adresse + 1

Die Slave-Adresse wird in das Feld „Programmierbare Adresse“ eingegeben. Hier lässt sich ein Wert von 0..253 eingeben. Da im Modbus-Protokoll die 0 nicht zulässig ist, wird automatisch eine 1 auf das Ergebnis aufaddiert. So ergibt sich eine Slave-Adresse im Bereich von 1..254. Der Adresswahlschalter auf der Gerätefrontplatte hat keine Funktion

Beispiel:

Unabhängig davon, ob es sich um ein einachsiges oder ein doppelachsiges Gerät handelt, kommuniziert der Antrieb mit der Slave-Adresse 3 + 1 = 4.

2.2.3 Parity-Check

Der Antrieb unterstützt die folgenden Einstellungen für die Parität:

- Kein Parity-Check (1 Stoppbit)
- Gerade Parität (1 Stoppbit)
- Ungerade Parität (1 Stoppbit)

3 Funktionscodes

Die folgenden Funktionscodes werden unterstützt:

Funktionsgruppe	Name	Funktionscode
Datenzugriff 16 Bit	Read Holding Registers	03 (03_h)
	Read Input Register	04 (04_h)
	Write Single Register	06 (06_h)
	Write Multiple Register	16 (10_h)
	Read/Write Multiple Registers	23 (17_h)
Diagnose	Read Error Counter	08 (08_h)
CAN over Modbus	Encapsulated CAN Message	43 ($2B_h$)

3.1 FC 03 Read Input Register / FC 04 Read Holding Register

Mit diesen beiden Funktionscodes können ein oder mehrere 16-Bit-Register gelesen werden. Sie werden zum Lesen der PDO-Daten vom Antrieb verwendet.

Anfrage:

Name	Länge	Wert
Slave-Adresse	1 Byte	
Funktionscode	1 Byte	03_h / 04_h
Registeradresse	1 Byte high	0000_h bis $FFFF_h$
	1 Byte low	
Anzahl Register	1 Byte high	n = 1 bis 48 (30_h)
	1 Byte low	
CRC	1 Byte low	
	1 Byte high	

Antwort:

Name	Länge	Wert
Slave-Adresse	1 Byte	
Funktionscode	1 Byte	03_h / 04_h
Anzahl Bytes	1 Byte	2 × n
Registerwerte	1 Byte high	Register (1) Wert
	1 Byte low	
	1 Byte high	Register (2) Wert
	1 Byte low	
	⋮	⋮
	1 Byte high	Register (n) Wert
	1 Byte low	
CRC	1 Byte low	
	1 Byte high	

Fehlerrückmeldung:

Name	Länge	Wert
Slave-Adresse	1 Byte	
Funktionscode	1 Byte	83_h / 84_h
Fehlernummer	1 Byte	Siehe Kommunikationsfehler (S. 16)
CRC	1 Byte low	
	1 Byte high	

Input Register sind in der Regel nur lesbar während Holding Register gelesen und geschrieben werden können. Bei den Funktionscodes FC-03 und FC-04 unterscheidet der Antrieb nicht zwischen *read only*- und *read/write*-Registern. Beide Funktionscodes sind aber aus Kompatibilität zum Modbus-Protokoll implementiert und werden identisch behandelt.

3.2 FC 06 Write Single Register

Mit diesem Funktionscode kann ein einzelnes 16-Bit-Register geschrieben werden. Er wird zum Schreiben der PDO-Daten in den Antrieb verwendet.

Anfrage:

Name	Länge	Wert
Slave-Adresse	1 Byte	
Funktionscode	1 Byte	06_h
Registeradresse	1 Byte high	0000_h bis $FFFF_h$
	1 Byte low	
Registerwert	1 Byte high	0000_h bis $FFFF_h$
	1 Byte low	
CRC	1 Byte low	
	1 Byte high	

Antwort:

Name	Länge	Wert
Slave-Adresse	1 Byte	
Funktionscode	1 Byte	06_h
Registeradresse	1 Byte high	0000_h bis $FFFF_h$
	1 Byte low	
Registerwert	1 Byte high	0000_h bis $FFFF_h$
	1 Byte low	
CRC	1 Byte low	
	1 Byte high	

Fehlerrückmeldung:

Name	Länge	Wert
Slave-Adresse	1 Byte	
Funktionscode	1 Byte	86_h
Fehlernummer	1 Byte	Siehe Kommunikationsfehler (S. 16)
CRC	1 Byte low	
	1 Byte high	

3.3 FC 08 Diagnostics

Dieser Funktionscode ermöglicht den Test und die Diagnose der Modbus-Kommunikation. Über die „Diagnosefunktion“ mit einer Länge von 2 Bytes können Diagnosetests ausgelöst oder Werte abgefragt werden. Einige Diagnosefunktionen verwenden Daten im Anfrage/Antwort-Telegramm. Auch wenn diese Daten nicht verwendet werden, müssen diese Bytes mit übertragen werden, damit das Diagnostics-Telegramm immer die gleiche Länge hat.

Anfrage:

Name	Länge	Wert
Slave-Adresse	1 Byte	

Name	Länge	Wert
Funktionscode	1 Byte	08_h
Diagnosefunktion	1 Byte high	0, 10, 11, 12, 13, 14, 15
	1 Byte low	
Daten	1 Byte high	0000_h bis $FFFF_h$
	1 Byte low	
CRC	1 Byte low	
	1 Byte high	

Antwort:

Name	Länge	Wert
Slave-Adresse	1 Byte	
Funktionscode	1 Byte	08_h
Diagnosefunktion	1 Byte high	0, 10, 11, 12, 13, 14, 15
	1 Byte low	
Daten	1 Byte high	0000_h bis $FFFF_h$
	1 Byte low	
CRC	1 Byte low	
	1 Byte high	

Fehlerrückmeldung:

Name	Länge	Wert
Slave-Adresse	1 Byte	
Funktionscode	1 Byte	88_h
Fehlernummer	1 Byte	Siehe Kommunikationsfehler (S. 16)
CRC	1 Byte low	
	1 Byte high	

Diagnosefunktionen

Die folgenden Diagnosefunktionen sind verfügbar:

Funktions-code	Funktion	Anfragedaten	Antwortdaten
0 (00_h)	Return Query Data Das empfangene Telegramm wird zurückgesendet. Dabei werden die im Datenfeld empfangenen Datenbytes an den Sender zurückgeschickt.	Beliebige Werte	Die empfangenen Werte
10 ($0A_h$)	Clear Counters and Diagnostic Register Alle Diagnoseregister werde auf 0 zurückgesetzt.	–	–
11 ($0B_h$)	Return Bus Message Count Gibt die Anzahl der Nachrichten an, die vom System erkannt wurden.	–	Bus Message Counter
12 ($0C_h$)	Return Bus Communication Error Count Gibt die Anzahl der CRC-Fehler an.	–	CRC Error Counter
13 ($0D_h$)	Return Bus Exception Error Count Gibt die Anzahl der Kommunikationsfehler an. die das System gemeldet hat.	–	Exception Error Counter
14 ($0E_h$)	Return Slave Message Count Gibt die Anzahl der vom System gesendeten Nachrichten an.	–	Message Counter
15 ($0F_h$)	Return Slave No Response Count Gibt die Anzahl der vom System gesendeten Nachrichten an, die nicht beantwortet wurden.	–	No Response Counter

3.4 FC 16 Write Multiple Registers

Mit diesem Funktionscode können mehrere 16-Bit-Register geschrieben werden. Er wird zum Schreiben der PDO-Daten in den Antrieb verwendet.

Anfrage:

Name	Länge	Wert
Slave-Adresse	1 Byte	
Funktionscode	1 Byte	10_h
Registeradresse	1 Byte high	0000_h bis $FFFF_h$
	1 Byte low	
Anzahl Register	1 Byte high	n = 1 bis 48 (30_h)
	1 Byte low	
Anzahl Bytes	1 Byte	2 × n
Registerwerte	1 Byte high	Register (1) Wert
	1 Byte low	
	1 Byte high	Register (2) Wert
	1 Byte low	
	⋮	⋮
	1 Byte high	Register (n) Wert
	1 Byte low	
CRC	1 Byte low	
	1 Byte high	

Antwort:

Name	Länge	Wert
Slave-Adresse	1 Byte	
Funktionscode	1 Byte	10_h
Startadresse	1 Byte high	0000_h bis $FFFF_h$
	1 Byte low	
Anzahl Register	1 Byte high	n = 1 bis 48 (30_h)
	1 Byte low	
CRC	1 Byte low	
	1 Byte high	

Fehlerrückmeldung:

Name	Länge	Wert
Slave-Adresse	1 Byte	
Funktionscode	1 Byte	90_h
Fehlernummer	1 Byte	Siehe Kommunikationsfehler (S. 16)
CRC	1 Byte low	
	1 Byte high	

3.5 FC 23 Read/Write Multiple Registers

Mit diesem Funktionscode können ein oder mehrere 16-Bit-Register geschrieben werden, während andere Register gelesen werden. Er wird zum Schreiben und Lesen der PDO-Daten des Antriebs verwendet.

Anfrage:

Name	Länge	Wert
Slave-Adresse	1 Byte	
Funktionscode	1 Byte	17_h

Name	Länge	Wert
Registeradresse lesen	1 Byte high	0000_h bis $FFFF_h$
	1 Byte low	
Anzahl Register lesen	1 Byte high	n = 1 bis 48 (30_h)
	1 Byte low	
Registeradresse schreiben	1 Byte high	0000_h bis $FFFF_h$
	1 Byte low	
Anzahl Register schreiben	1 Byte high	m = 1 bis 48 (30_h)
	1 Byte low	
Anzahl Bytes	1 Byte	2 × m
Registerwerte	1 Byte high	Register (1) Wert
	1 Byte low	
	1 Byte high	Register (2) Wert
	1 Byte low	
	⋮	⋮
	1 Byte high	Register (m) Wert
	1 Byte low	
CRC	1 Byte low	
	1 Byte high	

Antwort:

Name	Länge	Wert
Slave-Adresse	1 Byte	
Funktionscode	1 Byte	17_h
Anzahl Bytes	1 Byte	2 × n
Registerwerte	1 Byte high	Register (1) Wert
	1 Byte low	
	1 Byte high	Register (2) Wert
	1 Byte low	
	⋮	⋮
	1 Byte high	Register (n) Wert
	1 Byte low	
CRC	1 Byte low	
	1 Byte high	

Fehlerrückmeldung:

Name	Länge	Wert
Slave-Adresse	1 Byte	
Funktionscode	1 Byte	97_h
Fehlernummer	1 Byte	Siehe Kommunikationsfehler (S. 16)
CRC	1 Byte low	
	1 Byte high	

3.6 FC 43 Encapsulated Interface Transport

Dieser Funktionscode ermöglicht einen einfachen Zugriff auf das Objektverzeichnis des Antriebs.

Zurzeit ist nur der Zugriff auf 16-Bit- oder 32-Bit-Objekte möglich. Der Zugriff auf Array- oder String-Objekte ist nicht implementiert.

Anfrage:

Name	Länge	Wert
Slave-Adresse	1 Byte	

Name	Länge	Wert
Funktionscode	1 Byte	$2B_h$
MEI type	1 Byte	$0D_h$
Steuercode	1 Byte	00_h / 80_h Lesen / Schreiben
Reserviertes Feld	1 Byte	00_h reserviert
Node-ID	1 Byte	01_h bis $7F_h$
Index	1 Byte high	0000_h bis $FFFF_h$ Objektnummer
	1 Byte low	
Subindex	1 Byte	00_h bis FF_h Objekt-Subindex
Adressoffset	1 Byte high	0000_h Datenoffset
	1 Byte low	
Anzahl Bytes	1 Byte high	n = 0000_h bis 0004_h Bytes
	1 Byte low	
Datenwerte	1 Byte #0	Byte (1) Wert
	1 Byte #1	
	1 Byte #2	
	1 Byte #3	Byte (n) Wert
CRC	1 Byte low	
	1 Byte high	

Antwort:

Name	Länge	Wert
Slave-Adresse	1 Byte	
Funktionscode	1 Byte	$2B_h$
MEI type	1 Byte	$0D_h$
Steuercode	1 Byte	00_h / 80_h Lesen / Schreiben
Reserviertes Feld	1 Byte	00_h reserviert
Node-ID	1 Byte	01_h bis $7F_h$
Index	1 Byte high	0000_h bis $FFFF_h$ Objektnummer
	1 Byte low	
Subindex	1 Byte	00_h bis FF_h Objekt-Subindex
Adressoffset	1 Byte high	0000_h Datenoffset
	1 Byte low	
Anzahl Bytes	1 Byte high	n = 0000_h bis 0004_h Bytes
	1 Byte low	
Datenwerte	1 Byte #0	Byte (1) Wert
	1 Byte #1	
	1 Byte #2	
	1 Byte #3	Byte (n) Wert
CRC	1 Byte low	
	1 Byte high	

Fehlerrückmeldung:

Name	Länge	Wert
Slave-Adresse	1 Byte	
Funktionscode	1 Byte	AB_h
Fehlernummer	1 Byte	01_h, 02_h, 03_h, 04_h Siehe Kommunikationsfehler (S. 16)
CRC	1 Byte low	
	1 Byte high	

Antwort, wenn ein Fehler beim Objektzugriff auftritt:

Name	Länge	Wert
Slave-Adresse	1 Byte	

Name	Länge	Wert
Funktionscode	1 Byte	AB_h
Fehlernummer	1 Byte	FF_h erweiterte Ausnahme (Extended Exception)
Extended Exception Länge	1 Byte high	6
	1 Byte low	
MEI type	1 Byte	$0D_h$
Exception Code	1 Byte	CE_h
SDO-Fehlercode	1 Byte #0	Byte (1) Wert
	1 Byte #1	
	1 Byte #2	
	1 Byte #3	Byte (4) Wert siehe SDO-Fehler (S. 16)
CRC	1 Byte low	
	1 Byte high	

Datenbeschreibung

- **MEI-type:**

  Modbus Encapsulated Interface (MEI) ist 0Dh und selektiert CANopen über Modbus-Transport. Das ist der einzige MEI-type, der vom Antrieb unterstützt wird.

- **Steuercode:**

Bit	Beschreibung	Verwendung
0..6	Nicht verwendet	Auf Null gesetzt
7	Daten lesen oder schreiben	0 = lesen; 1 = schreiben

- **Node-ID:**

  Die Node-ID hat denselben Wert wie die Slave-Adresse.

- **Index:**

  Der Index enthält die Objektnummer aus dem Objektverzeichnis des Antriebs und entspricht dem Index in der Software *drivemaster2*. Der Index startet mit der Nummer 0 und ist fortlaufend. Die Liste der Objekte finden Sie im Objektbrowser der *drivemaster2*-Oberfläche.

  Objekte aus den CiA-Standards DS301 (Objektnummern ab 0x2000) und DS402 (Objektnummern ab 0x6000) werden nicht unterstützt.

- **Subindex:**

  Der Subindex wird zur Auswahl von Daten aus Array- und String-Objekten verwendet. Bei einfachen Objekten wird der Subindex auf Null gesetzt.

- **Adressoffset:**

  Der Adressoffset wird für SD2x-Antriebe nicht verwendet. Setzen Sie den Adressoffset auf Null.

- **Anzahl Bytes:**

  Diese Bytes geben die Anzahl der zu lesenden oder zu schreibenden Bytes an.

- **Datenwerte:**
  - Anfragetelegramm: zu schreibende Daten
  - Antworttelegramm: zu lesende Daten

  Im Gegensatz zu allen anderen Telegrammen im Modbus-Protokoll werden die Daten im Encapsulated Interface Transport im Little-Endian-Format übertagen. Damit ergibt sich für 16-Bit- und 32-Bit-Werte folgende Bytereihenfolge:

  16-Bit-Wert: Beispiel 0x1234

Byte	Byte #0	Byte #1	Byte #2	Byte #3
Wert	0x34	0x12	–	–

32-Bit-Wert: Beispiel 0x12345678

Byte	Byte #0	Byte #1	Byte #2	Byte #3
Wert	0x78	0x56	0x34	0x12

- **Fehlernummer:**
  Die Fehlernummer kann den Wert 01_h, 02_h, 03_h, 04_h und FF_h annehmen. Eine entsprechende Fehlerbeschreibung finden Sie in der Tabelle Kommunikationsfehler (S. 16). Der Wert FF_h bedeutet, dass eine erweiterte Fehlernummer gesendet wird.
- **Exception Code:**
  Zurzeit ist nur der Code CE_h implementiert. Dieser bedeutet, dass ein 4 Byte langer Objektzugriffsfehlercode (SDO-Fehlercode) folgt.
- **SDO-Fehlercode:**
  SDO-Fehlercodes sind 4 Byte lange Objektzugriffsfehlercodes, siehe SDO-Fehler (S. 16).

3.7 Fehlercodes für die Kommunikation

Die folgenden Fehlercodes können übertragen werden:

Fehlercode	Name	Bedeutung
01	Illegal Function	Der Modbus-Funktionscode aus der Anfrage ist nicht implementiert.
02	Illegal Data Address	Die Datenadresse aus der Anfrage ist ungültig.
03	Illegal Data Value	Der Datenwert aus der Anfrage ist außerhalb des Datenbereichs.
04	Slave Device Failure	Als das Gerät versucht hat, die angefragte Aktion auszuführen, ist ein nicht behebbarer Fehler aufgetreten.
FF_h	SDO Error	Beim Objektzugriff ist ein Fehler aufgetreten. Ein erweiterter Fehlercode wurde gesendet, siehe SDO-Fehler (S. 16).

3.8 Erweiterte Fehlercodes (SDO-Fehler)

Die folgenden Tabelle zeigt Objektfehler, die beim Zugriff über die Funktion FC 43 „Encapsulated Interface Transport“ auftreten können:

Fehlercode	Beschreibung
$FFFF0003_h$	Client/Server-Befehl nicht gültig oder unbekannt
$FFFF0007_h$	Kein Speicherplatz mehr frei
$FFFF0008_h$	Nicht erlaubter Zugriff auf ein Objekt
$FFFF0009_h$	Versuch, ein Write-Only-Objekt zu lesen
$FFFF000A_h$	Versuch, ein Read-Only-Objekt zu beschreiben
$FFFF000B_h$	Objekt existiert nicht im Objektverzeichnis
$FFFF000C_h$	Objekt kann nicht in das PDO gemappt werden
$FFFF000D_h$	Anzahl und Länge der zu mappenden Objekte überschreitet PDO-Länge
$FFFF000E_h$	Allgemeine Inkompatibilität der Parameter
$FFFF000F_h$	Allgemeine interne Inkompatibilität im Gerät
$FFFF0010_h$	Zugriff verweigert aufgrund eines Hardwarefehlers
$FFFF0011_h$	falscher Datentyp, Länge des Service-Parameters falsch
$FFFF0012_h$	falscher Datentyp, Länge des Service-Parameters zu groß
$FFFF0013_h$	falscher Datentyp, Länge des Service-Parameters zu klein
$FFFF0014_h$	Subindex nicht vorhanden
$FFFF0015_h$	Wertebereich des Parameters überschritten (nur bei Schreibzugriff)
$FFFF0016_h$	Wert des geschriebenen Parameters zu hoch

Fehlercode	Beschreibung
$FFFF0017_h$	Wert des geschriebenen Parameters zu niedrig
$FFFF0018_h$	Maximalwert kleiner als Minimalwert
$FFFF0019_h$	Allgemeiner Fehler
$FFFF001A_h$	Daten können nicht übertragen oder in der Applikation gespeichert werden
$FFFF001B_h$	Daten können wegen lokaler Steuerung nicht übertragen oder in der Applikation gespeichert werden
$FFFF001C_h$	Daten können wegen aktuellem Gerätestatus nicht übertragen oder in der Applikation gespeichert werden
$FFFF001D_h$	Dynamische Erzeugung des Objektverzeichnisses nicht möglich oder kein Objektverzeichnis vorhanden (z. B. Objektverzeichnis wurde aus einer Datei generiert und die Datei enthält einen Fehler)
$FFFF001E_h$	Angefordertes Objekt ist zu groß für einzelne Nachricht

4 Prozessdaten

Die folgenden Prozessdatenobjekte (PDO) stehen zur Verfügung:

- RxPDO: Sollwerte empfangen
- TxPDO: Istwerte senden

Die Prozessabbilder fangen bei den folgenden Modbus-Registeradressen an:

Modbus Registeradresse	Name	Beschreibung
2000_d	TxPDO (Input Register)	Istwerte
3000_d	RxPDO (Holding Register)	Sollwerte

In den folgenden Kapiteln finden Sie die Default-Konfiguration der Prozessdatenobjekte.

4.1 Default-RxPDO

Modbus-Register	Name	Länge	Beschreibung
3000_d	Control Word	16 Bit	Steuerwort: Start / Stopp
3001_d	Target Speed	16 Bit	Drehzahlsollwert in 0,1 Hz
3002_d	Torque Limit Iq	16 Bit	Strombegrenzung in 0,1 A
3003_d	Watchdog	16 Bit	Watchdog-Zähler

4.2 Default-TxPDO

Modbus-Register	Name	Länge	Beschreibung
2000_d	Status Word	16 Bit	Statuswort: Freigegeben / Fehler
2001_d	Actual Speed	16 Bit	Drehzahl in 0,1 Hz
2002_d	Torque Limit Iq	16 Bit	Strombegrenzung in 0,1 A
2003_d	Act Apparent Current	16 Bit	Strom in 0,1 A
2004_d	Act Output Power	16 Bit	Leistung in 0,1 kW
2005_d	Error Code	16 Bit	Fehlercode
2006_d	Temp Powerstage	16 Bit	Temperatur der Leistungsendstufe in 0,1 °C
2007_d	Temp Motor	16 Bit	Temperatur des Motors in 0,1 °C
2008_d	Output Voltage	16 Bit	Ausgangsspannung in 0,1 V
2009_d	DC Voltage	16 Bit	Zwischenkreisspannung in 0,1 V
2010_d	Hardware ID	32 Bit	Hardware-ID
2012_d	Hardware Version	32 Bit	Hardwareversion
2014_d	Software ID	32 Bit	Software-ID
2016_d	Software Version	32 Bit	Softwareversion
2018_d	Max Iq Current	16 Bit	Max. Strom Iq in 0,1 A
2019_d	Max Speed	16 Bit	Max. Drehzahl in 0,1 Hz
2020_d	Speed Scaling	32 Bit	Drehzahlskalierung in 0,001 1/min
2022_d	Current Scaling	32 Bit	Stromskalierung in 0,1 A

4.3 Bytereihenfolge der Prozessdaten

Die Prozessdaten werden im Big-Endian-Format übertragen.

Die folgende Bytereihenfolge für 16-Bit und 32-Bit-Werte gilt für alle Prozessdaten der Funktionscodes FC 03, FC 04, FC 16 und FC 23.

Hinweis

Beim Funktionscode FC 43 ist die Bytereihenfolge anders, da die Daten im Little-Endian-Format übertragen werden, siehe Kapitel 3.6 „FC 43 Encapsulated Interface Transport“, Seite 13.

16-Bit-Register

Für ein 16-Bit-Register ergibt sich folgende Bytereihenfolge in den seriellen Daten.

Beispiel = 0x1234

	16-Bit-Register			
Byte	**Byte(0)**	**Byte(1)**		
Wert	0x12	0x34		

32-Bit-Register

Bei 32-Bit-Werten werden zwei 16-Bit-Register zu einem 32-Bit-Wert zusammengefasst. Dabei liegt Register(x) an der Adresse x und Register(x+1) liegt an der folgenden Adresse. Es ergibt sich folgende Bytereihenfolge.

Beispiel = 0x12345678

	16-Bit-Register(x)		16-Bit-Register(x+1)	
Byte	**Byte(0)**	**Byte(1)**	**Byte(0)**	**Byte(1)**
Wert	0x12	0x34	0x56	0x78

5 Objektverzeichnis

Control Word (Steuerwort)

Das Control Word entspricht dem Objekt 0x6040 aus dem DS402-Antriebsprofil.

Index	68_d
Name	Control Word
Objektcode	VAR
Datentyp	Unsigned 16

Zugriff	RW
PDO-Zuordnung	möglich
Einheit	Bits codiert nach DS402-Standard
Wertebereich	0 … 65535
Vorgabewert	0
Index nach DS402	0x6040

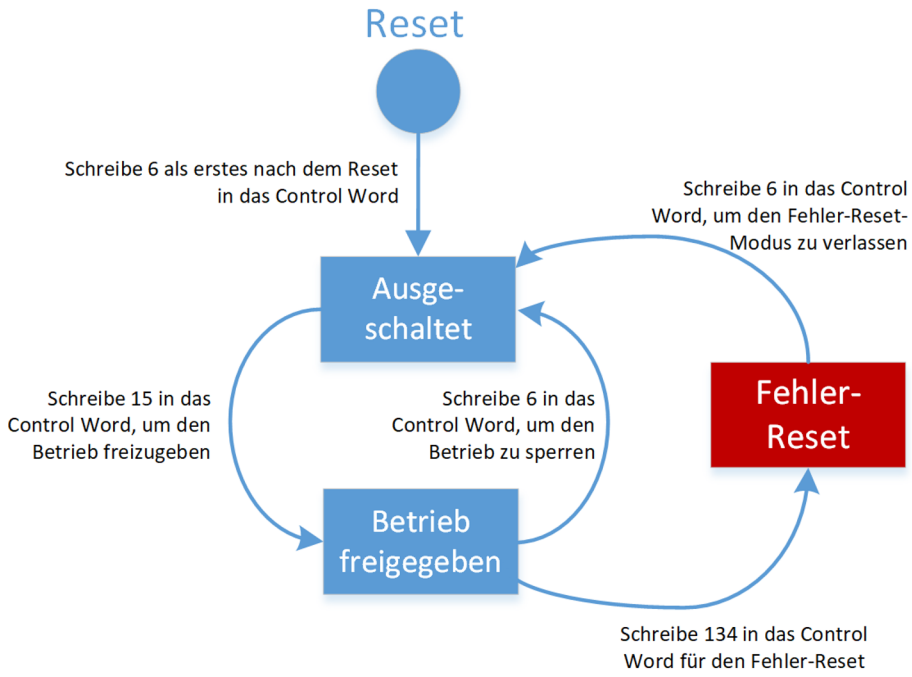

Abb. 3: Antriebsbetrieb über Control Word

Das Steuerwort ist in der Dokumentation „Antriebssystem SD2 – Gerätesteuerung“ beschrieben (siehe Abschnitt „Aufbau des Steuerwortes (Objekt 68_D)“.

Target Velocity (Drehzahlsollwert)

Index	210_d
Name	Target Velocity
Objektcode	VAR
Datentyp	Integer 16

Zugriff	RW
PDO-Zuordnung	möglich
Einheit	0,1 Hz (10 = 1 Hz)

Wertebereich	$(-2^{15}) \ldots (2^{15}-1)$
Vorgabewert	0
Index nach DS402	0x60FF

Der Drehzahlsollwert wird in 0,1 Hz angegeben.

Torque Limit Iq (Strombegrenzung)

Index	209_d
Name	Target Torque Limit
Objektcode	VAR
Datentyp	Integer 16

Zugriff	RW
PDO-Zuordnung	möglich
Einheit	0,1 A (10 = 1 A)
Wertebereich	$(-2^{15}) \ldots (2^{15}-1)$
Vorgabewert	0
Index nach DS402	–

Die Strombegrenzung Iq wird in 0,1 A angegeben.

Die Strombegrenzung wird von der Steuerung vorgegeben. Der Wert ist auf den Antriebsspitzenstrom begrenzt. Das Objekt entspricht dem Objekt 209 aus der *drivemaster2*-Objektliste, das in der Dokumentation „Antriebssystem SD2 – Gerätesteuerung“ beschrieben ist.

Watchdog

Index	–
Name	Watchdog Counter
Objektcode	VAR
Datentyp	Unsigned 16

Zugriff	RW
PDO-Zuordnung	möglich
Einheit	1 ms
Wertebereich	0 … 65535
Vorgabewert	0
Index nach DS402	–

Mit dem Watchdog-Register kann der Antrieb deaktiviert werden, wenn die Kommunikation zwischen Steuerung und Antrieb unterbrochen wird. Dazu tragen Sie in das Register eine Timeout-Zeit ein. Das Register muss dann innerhalb dieser Timeout-Zeit neu beschrieben werden. Läuft die Zeit ab, stoppt der Antrieb mit der Fehlermeldung „Heartbeat/Watchdog“. Wenn Sie den Wert 0 in das Register eintragen, wird die Watchdog-Überwachung deaktiviert.

Status Word (Statuswort)

Das Status Word entspricht dem Objekt 0x6041 aus dem DS402-Antriebsprofil.

Index	67_d
Name	Status Word
Objektcode	VAR
Datentyp	Unsigned 16

Zugriff	RO
PDO-Zuordnung	möglich
Einheit	Bits codiert nach DS402-Standard
Wertebereich	0 … 65535
Vorgabewert	0
Index nach DS402	0x6041

Das Statuswort ist in der Dokumentation „Antriebssystem SD2 – Gerätesteuerung“ beschrieben (siehe Abschnitt „Aufbau des Statuswortes (Objekt 67_D)“.

Die meist verwendeten Bits sind:

Bit	Name	Beschreibung
0	Ready to switch on	Antrieb ist bereit zum Einschalten
2	Operation enabled	Antriebsbetrieb ist freigegeben
3	Fault	Ein Fehler ist aufgetreten

Actual Speed (Drehzahlistwert)

Index	168_d
Name	Actual Speed
Objektcode	VAR
Datentyp	Integer 16

Zugriff	RO
PDO-Zuordnung	möglich
Einheit	0,1 Hz (10 = 1 Hz)
Wertebereich	(-2^{15}) … $(2^{15}-1)$
Vorgabewert	0
Index nach DS402	–

Der Drehzahlistwert wird in 0,1 Hz angegeben.

Actual Apparent Current (Istwert Scheinstrom)

Index	431_d
Name	Actual Apparent Current
Objektcode	VAR
Datentyp	Integer 16

Zugriff	RO
PDO-Zuordnung	möglich
Einheit	0,1 A (10 = 1 A)
Wertebereich	(-2^{15}) … $(2^{15}-1)$
Vorgabewert	0
Index nach DS402	–

Der Istwert für den Scheinstrom wird in 0,1 A angegeben.

Actual Output Power (Istwert Ausgangsleistung)

Index	390_d
Name	Actual Output Power
Objektcode	VAR
Datentyp	Integer 16

Zugriff	RO
PDO-Zuordnung	möglich
Einheit	0,1 kW (10 = 1 kW)
Wertebereich	(-2^{15}) … $(2^{15}-1)$
Vorgabewert	0
Index nach DS402	–

Der Istwert für die Ausgangsleistung wird in 0,1 kW angegeben.

Error Code (Gespeicherte Fehlermeldung)

Index	70_d
Name	Error Code
Objektcode	VAR
Datentyp	Unsigned 16

Zugriff	RO
PDO-Zuordnung	möglich
Einheit	–
Wertebereich	(-2^{15}) ... $(2^{15}-1)$
Vorgabewert	0
Index nach DS402	0x603F

Eine Beschreibung der Fehlercodes ist im Dokument „Antriebssystem SD2 – Gerätesteuerung“ zu finden, Kapitel „Liste der Antriebsfehlermeldungen“.

Temperature Power Stage (Temp. Leistungsendstufe)

Index	41_d
Name	Temperature Power Stage
Objektcode	VAR
Datentyp	Integer 16

Zugriff	RO
PDO-Zuordnung	möglich
Einheit	0,1 °C (10 = 1 °C)
Wertebereich	(-2^{15}) … $(2^{15}-1)$
Vorgabewert	0
Index nach DS402	–

Die Temperatur der Leistungsendstufe wird in 0,1 °C angegeben.

Temperature Motor (Motortemperatur)

Index	63_d
Name	Temperature Motor
Objektcode	VAR
Datentyp	Integer 16

Zugriff	RO
PDO-Zuordnung	möglich
Einheit	0,1 °C (10 = 1 °C)
Wertebereich	(-2^{15}) … $(2^{15}-1)$
Vorgabewert	0

Index nach DS402	–

Die Motortemperatur wird in 0,1 °C angegeben.

Output Voltage (Ausgangsspannung)

Index	434_d
Name	Output Voltage
Objektcode	VAR
Datentyp	Integer 16

Zugriff	RO
PDO-Zuordnung	möglich
Einheit	0,1 V (10 = 1 V)
Wertebereich	(-2^{15}) … $(2^{15}-1)$
Vorgabewert	0
Index nach DS402	–

Die Ausgangsspannung wird in 0,1 V angegeben.

DC Voltage (Zwischenkreisspannung)

Index	33_d
Name	DC Voltage
Objektcode	VAR
Datentyp	Integer 16

Zugriff	RO
PDO-Zuordnung	möglich
Einheit	0,1 V (10 = 1 V)
Wertebereich	(-2^{15}) … $(2^{15}-1)$
Vorgabewert	0
Index nach DS402	–

Die Zwischenkreissspannung wird in 0,1 V angegeben.

Hardware ID

Index	2_d
Name	Hardware ID
Objektcode	VAR
Datentyp	Unsigned 32

Zugriff	RO
PDO-Zuordnung	möglich
Einheit	–
Wertebereich	(-2^{31}) … $(2^{31}-1)$
Vorgabewert	0
Index nach DS402	–

Die Hardware-ID enthält einen Identifikationscode für den Gerätetyp, der als 32-Bit-Wert codiert ist.

Hardware Version

Index	5_d
Name	Hardware Version
Objektcode	VAR
Datentyp	Unsigned 32

Zugriff	RO
PDO-Zuordnung	möglich
Einheit	–
Wertebereich	$(-2^{31}) \dots (2^{31}-1)$
Vorgabewert	0
Index nach DS402	–

Die Hardwareversion ist als 32-Bit-Wert codiert, der sich in einen 16-Bit-Vorkommaanteil und einen 16-Bit-Nachkommaanteil gliedert.

Beispiel: 0x00020010 ist Version 2.16

Software ID

Index	16_d
Name	Software ID
Objektcode	VAR
Datentyp	Unsigned 32

Zugriff	RO
PDO-Zuordnung	möglich
Einheit	–
Wertebereich	$(-2^{31}) \dots (2^{31}-1)$
Vorgabewert	0
Index nach DS402	–

Die Software-ID enthält einen Identifikationscode für die Antriebssoftware, der als 32-Bit-Wert codiert ist.

Software Version

Index	17_d
Name	Software Version
Objektcode	VAR
Datentyp	Unsigned 32

Zugriff	RO
PDO-Zuordnung	möglich
Einheit	–
Wertebereich	$(-2^{31}) \dots (2^{31}-1)$
Vorgabewert	0
Index nach DS402	–

Die Softwareversion ist als 32-Bit-Wert codiert, der sich in einen 16-Bit-Vorkommaanteil und einen 16-Bit-Nachkommaanteil gliedert.

Beispiel: 0x00020010 ist Version 2.16

Max Iq Current (Max. Strom)

Index	171_d
Name	Max Iq Current
Objektcode	VAR
Datentyp	Integer 16

Zugriff	RO
PDO-Zuordnung	möglich
Einheit	0,1 A (10 = 1 A)
Wertebereich	$(-2^{15}) \dots (2^{15}-1)$
Vorgabewert	0
Index nach DS402	–

Der maximale Strom Iq des Antriebs wird in 0,1 A angegeben.

Das Objekt entspricht dem Objekt 171 aus der *drivemaster2*-Objektliste und gibt den aktuell zulässigen Maximalstrom des Stromreglers an. Für diesen Wert werden die in der Parametrierung eingestellten Strombegrenzungen des Antriebs miteinbezogen.

Max Speed (Max. Drehzahl)

Index	120_d
Name	Max Speed
Objektcode	VAR
Datentyp	Integer 16

Zugriff	RO
PDO-Zuordnung	möglich
Einheit	0x3FFF = Speed Scaling
Wertebereich	$(-2^{15}) \dots (2^{15}-1)$
Vorgabewert	0
Index nach DS402	–

Die maximale Drehzahl wird in 0x3FFF = Speed-Scaling-Objekt angegeben.

Speed Scaling (Drehzahlskalierung)

Index	177_d
Name	Speed Scaling
Objektcode	VAR
Datentyp	Integer 32

Zugriff	RO
PDO-Zuordnung	möglich
Einheit	0,001 1/min
Wertebereich	$(-2^{31}) \dots (2^{31}-1)$
Vorgabewert	0
Index nach DS402	–

Die Drehzahlskalierung wird in 0,001 1/min angegeben.

Current Scaling (Stromskalierung)

Index	182_d

Name	Current Scaling
Objektcode	VAR
Datentyp	Integer 32

Zugriff	RO
PDO-Zuordnung	möglich
Einheit	0,001 A
Wertebereich	(-2^{31}) … $(2^{31}-1)$
Vorgabewert	0
Index nach DS402	–

Die Stromskalierung wird in 0,001 A angegeben.

6 Beispiele

6.1 Beispiel 1: Antrieb verfahren

Das folgende Beispiel zeigt, welche Abfolge von Modbus-Protokollen nötig ist, um den Antrieb zu verfahren.

Aufgabe: Nach dem Einschalten des Antriebs soll eine Drehzahl von 100 Hz gefahren werden. Anschließend soll wieder abgebremst werden. Während des Fahrens soll die Funktion des Antriebs überwacht werden. Hierfür werden die Register 3000 bis 3003 des RxPDO beschrieben.

Die Modbusadresse des Antriebs ist auf 1 eingestellt, siehe hierzu Kapitel 2.2.2 „Slave-Adresse“, Seite 6.

Antrieb stillsetzen

Nachdem der Antrieb gebootet ist, wird der Befehl „Shutdown“ zum Stillsetzen des Antriebs in das Steuerwort geschrieben.

Die Steuerung sendet:

Name	Länge	Wert	Beschreibung
Slave-Adresse	1 Byte	01_h	
Funktionscode	1 Byte	10_h	Write Multiple Registers (FC 16)
Registeradresse	1 Byte high	$0B_h$	Register $0BB8_h = 3000_d$
	1 Byte low	$B8_h$	
Anzahl Register	1 Byte high	00_h	4 Register
	1 Byte low	04_h	
Anzahl Bytes	1 Byte	08_h	8 Byte
Control Word	1 Byte high	00_h	Shutdown-Befehl (Stillsetzen)
	1 Byte low	06_h	
Target Speed	1 Byte high	03_h	Drehzahlsollwert = 100 Hz
	1 Byte low	$E8_h$	
Torque Limit Iq	1 Byte high	00_h	Strombegrenzung = 10 A
	1 Byte low	64_h	
Watchdog	1 Byte high	01_h	Watchdog-Timeout = 500 ms
	1 Byte low	$F4_h$	
CRC	1 Byte low	48_h	CRC = 0xFC48
	1 Byte high	FC_h	

Der Antrieb antwortet:

Name	Länge	Wert	Beschreibung
Slave-Adresse	1 Byte	01_h	
Funktionscode	1 Byte	10_h	Write Multiple Registers (FC 16)
Registeradresse	1 Byte high	$0B_h$	Register $0BB8_h = 3000_d$
	1 Byte low	$B8_h$	
Anzahl Register	1 Byte high	00_h	4 Register
	1 Byte low	04_h	
CRC	1 Byte low	43_h	CRC = 0xCB43
	1 Byte high	CB_h	

Der Antrieb meldet nun den Status „Einschaltbereit“. Den Status finden Sie auf den Diagnoseseiten der *drivemaster2*-Software. Alternativ kann auch das Statusregister ausgelesen werden.

Da der Watchdog jetzt gesetzt ist, muss das Watchdog-Register mindestens alle 500 ms beschrieben werden. Anderenfalls wechselt der Antrieb in den Fehler-Zustand. Dies kann durch wiederholtes Senden des obigen Telegramms realisiert werden. Es ist vorteilhaft das Telegramm sogar alle 200 ms zu senden, damit der Watchdog auch dann nicht auslöst, wenn eine einzelne Übertagung fehlerhaft ist und verworfen wird.

Betrieb freigeben

Nun wird der Befehl „Enable Operation“ (Betrieb freigeben) gesendet.

Die Steuerung sendet:

Name	Länge	Wert	Beschreibung
Slave-Adresse	1 Byte	01_h	
Funktionscode	1 Byte	10_h	Write Multiple Registers (FC 16)
Registeradresse	1 Byte high	$0B_h$	Register $0BB8_h = 3000_d$
	1 Byte low	$B8_h$	
Anzahl Register	1 Byte high	00_h	4 Register
	1 Byte low	04_h	
Anzahl Bytes	1 Byte	08_h	8 Byte
Control Word	1 Byte high	00_h	Enable Operation-Befehl (Betrieb freigen)
	1 Byte low	$0F_h$	
Target Speed	1 Byte high	03_h	Drehzahlsollwert = 100 Hz
	1 Byte low	$E8_h$	
Torque Limit Iq	1 Byte high	00_h	Strombegrenzung = 10 A
	1 Byte low	64_h	
Watchdog	1 Byte high	01_h	Watchdog-Timeout = 500 ms
	1 Byte low	$F4_h$	
CRC	1 Byte low	48_h	CRC = 0xFC48
	1 Byte high	FC_h	

Die Antrieb antwortet:

Name	Länge	Wert	Beschreibung
Slave-Adresse	1 Byte	01_h	
Funktionscode	1 Byte	10_h	Write Multiple Registers (FC 16)
Registeradresse	1 Byte high	$0B_h$	Register $0BB8_h = 3000_d$
	1 Byte low	$B8_h$	
Anzahl Register	1 Byte high	00_h	4 Register
	1 Byte low	04_h	
CRC	1 Byte low	43_h	CRC = 0xCE43
	1 Byte high	CE_h	

Der Antrieb verfährt jetzt mit einer Drehzahl von 100 Hz.

6.2 Beispiel 2: Encapsulated Interface Transport

Das folgende Beispiel zeigt, wie Sie die Endstufentemperatur des Antriebs über Modbus auslesen können.

Die Endstufentemperatur wird auch in der *drivemaster2*-Software auf der Diagnose-seite „Antriebsistwerte“ angezeigt:

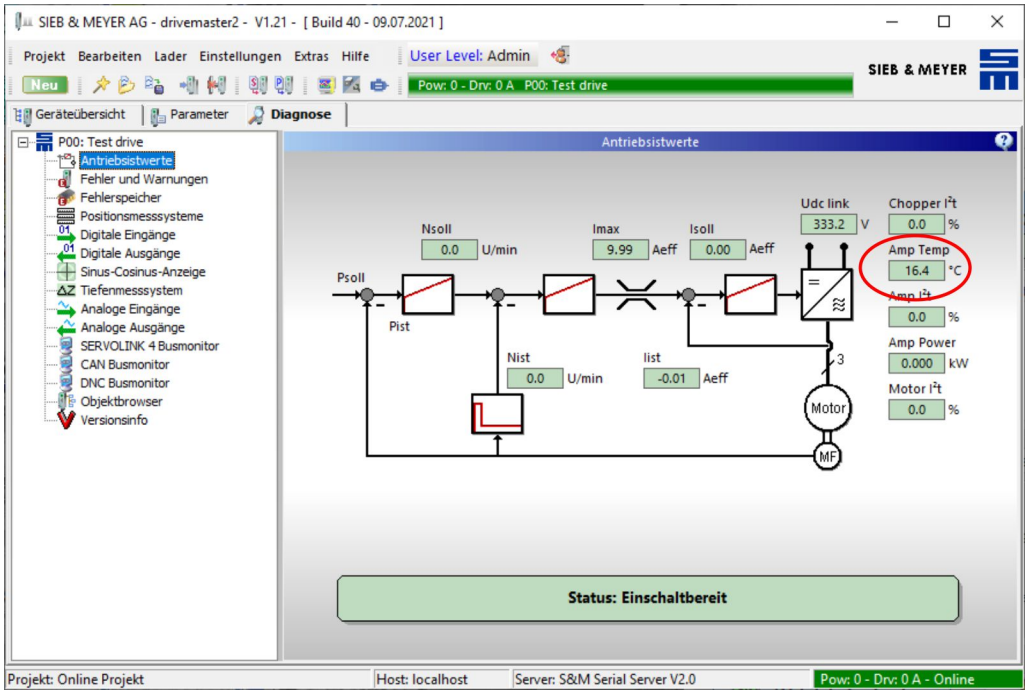

Abb. 4: Endstufentemperatur auf der Seite „Antriebsistwerte“

Zudem finden Sie die Endstufentemperatur im Objektbrowser der *drivemaster2*-Software. Dazu müssen Sie das Objekt „POWER_STAGE_TEMPERATURE_ACTUAL“ im Objektbrowser laden:

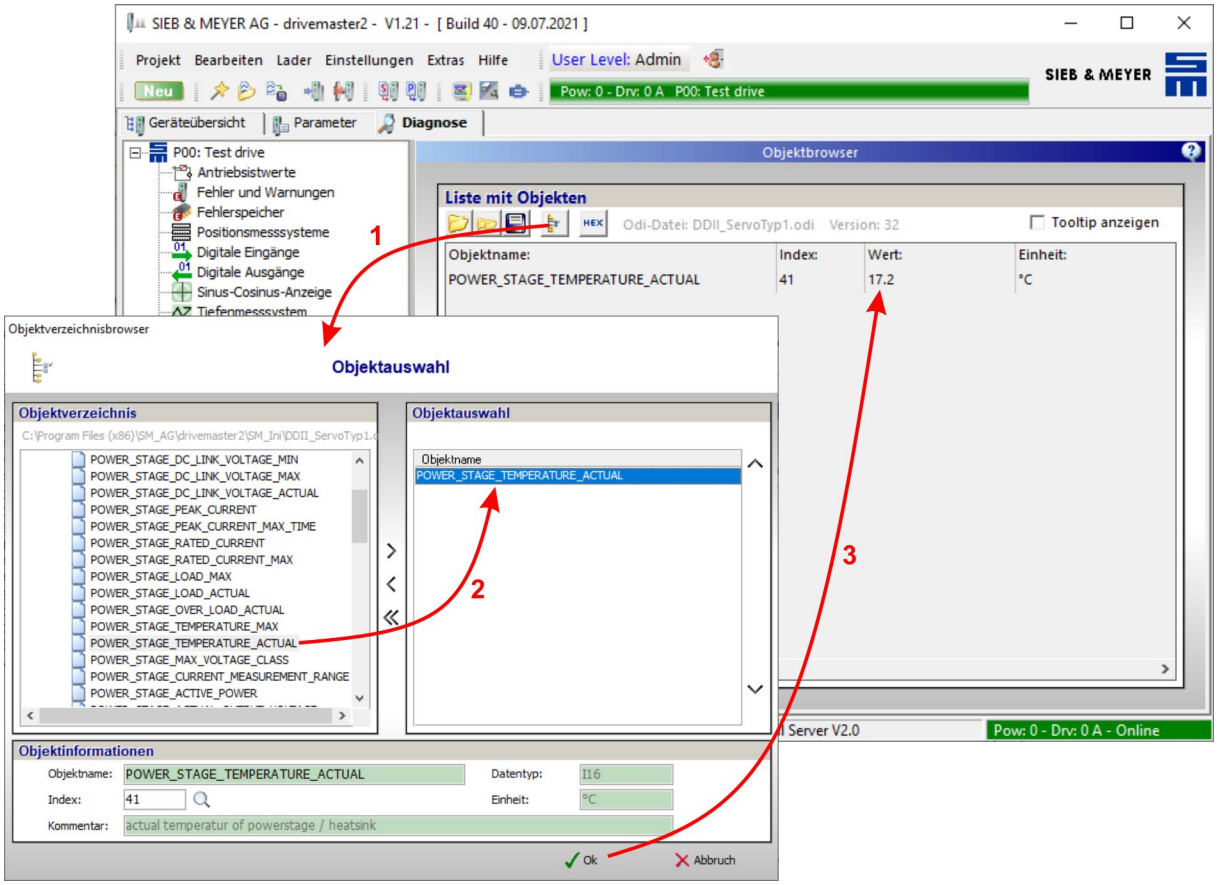

Abb. 5: Endstufentemperatur im Objektbrowser anzeigen

Im Objektbrowser können Sie nun die Eigenschaften des Objekts ermitteln:

- Objektindex: 41
- Datentyp: 16 Bit
- Subindex: 0 (Der Subindex wird nur bei Arrays benutzt, daher können Sie den Subindex auf 0 setzen.)

Aufgabe: Über den Funktionscode 43 „Encapsulated Interface Transport“ soll das Objekt mit den oben aufgeführten Eigenschaften ausgelesen werden.

Die Modbusadresse des Antriebs ist auf 1 eingestellt, siehe hierzu Kapitel 2.2.2 „Slave-Adresse“, Seite 6.

Um das Objekt auszulesen, sendet die Steuerung folgendes Telegramm:

Name	Länge	Wert	Beschreibung
Slave-Adresse	1 Byte	01_h	
Funktionscode	1 Byte	$2B_h$	Encapsulated Interface Transport
MEI type	1 Byte	$0D_h$	Modbus encapsulated Interface-Typ
Steuercode	1 Byte	00_h	Lesen
Node-ID	1 Byte	01_h	Node-ID 1
Index	1 Byte high	00_h	0029_h = Objekt 41_d
	1 Byte low	29_h	
Subindex	1 Byte	00_h	0

Name	Länge	Wert	Beschreibung
Adressoffset	1 Byte high	00_h	0
	1 Byte low	00_h	
Anzahl Bytes	1 Byte high	00_h	Objekt 41_d ist ein 16-Bit-Objekt
	1 Byte low	02_h	
CRC	1 Byte low	$C1_h$	CRC = 0x2BC1
	1 Byte high	$2B_h$	

Der Antrieb antwortet:

Name	Länge	Wert	Beschreibung
Slave-Adresse	1 Byte	01_h	
Funktionscode	1 Byte	$2B_h$	Encapsulated Interface Transport
MEI type	1 Byte	$0D_h$	Modbus encapsulated Interface-Typ
Steuercode	1 Byte	00_h	Lesen
Node-ID	1 Byte	01_h	Node-ID 1
Index	1 Byte high	00_h	0029_h = Objekt 41_d
	1 Byte low	29_h	
Subindex	1 Byte	00_h	0
Adressoffset	1 Byte high	00_h	0
	1 Byte low	00_h	
Anzahl Bytes	1 Byte high	00_h	Objekt 41_d ist ein 16-Bit-Objekt
	1 Byte low	02_h	
Datenwerte	1 Byte #0	00_h	$00AA_h$ = 170_d = 17,0 °C
	1 Byte #1	AA_h	
CRC	1 Byte low	$D1_h$	CRC = 0xDFD1
	1 Byte high	DF_h	

Die Endstufentemperatur 17,0 °C wird zurückgegeben.

6.3 Beispiel 3: CRC berechnen

Im folgenden Beispiel soll über die Funktion „Read Input Register“ das Statusregister des Antriebs ausgelesen werden. Dabei wird gezeigt, wie die CRC-Prüfsumme des Telegramms berechnet wird.

Die Modbusadresse des Antriebs ist auf 1 eingestellt, siehe hierzu Kapitel 2.2.2 „Slave-Adresse“, Seite 6.

Die Steuerung sendet:

Name	Länge	Wert	Beschreibung
Slave-Adresse	1 Byte	01_h	
Funktionscode	1 Byte	03_h	Read Input Register
Registeradresse	1 Byte high	07_h	$07D0_h$ = 2000_d = Statusregister
	1 Byte low	$D0_h$	
Anzahl Register	1 Byte high	00_h	1 Register
	1 Byte low	01_h	
CRC	1 Byte low	84_h	CRC = 0x8784
	1 Byte high	87_h	

Sie können den CRC-Wert für 1 Byte mit der folgenden C-Funktion berechnen. Im Übergabeparameter „input“ wird das Daten-Byte übergeben. Im Parameter „last“ wird der Wert der letzten Berechnung oder der Startwert übergeben. Beim Modbus startet die CRC-Berechnung mit dem Wert 0xFFFF.

```
uint16_t ModbusCRC(uint8_t input, uint16_t last)
{
    uint16_t crc     = last ^ (uint16_t)input;      // crc = Startwert XOR Input
    uint16_t polynom = 0xA001;
    for(int i = 0; i < 8; i++)                      // Schleife wird 8-mal durchlaufen
    {
        if(crc & 0x0001)                            // Wenn Bit 0 gesetzt ist, dann...
        {
            crc = (crc >> 1) ^ polynom;             // crc = (crc shift left 1) XOR polynom
        }
        else                                        // Wenn Bit 0 nicht gesetzt ist, dann...
        {
            crc = (crc >> 1);                       // crc = (crc shift left 1)
        };
     }
     return crc;                                    // Ergebnis nach 8 Durchläufen
}
```

Um den CRC-Wert für das gesamte Telegramm zu berechnen, können Sie die Funktion wie folgt verwenden:

```
    uint16_t crc;
    crc = ModbusCRC(0x01, 0xFFFF);
    crc = ModbusCRC(0x03, crc);
    crc = ModbusCRC(0x07, crc);
    crc = ModbusCRC(0xD0, crc);
    crc = ModbusCRC(0x00, crc);
    crc = ModbusCRC(0x01, crc);
    printf("CRC is 0x%04x\r\n",crc);
```

Die Ausgabe dieses Programms ist: CRC is 0x8784.

Die Antrieb antwortet:

Name	Länge	Wert	Beschreibung
Slave-Adresse	1 Byte	01_h	
Funktionscode	1 Byte	03_h	Read Input Register
Anzahl Bytes	1 Byte	02_h	2 Byte Daten
Registerwerte	1 Byte high	70_h	Status-Register-Wert: 0x7038
	1 Byte low	38_h	
CRC	1 Byte low	$9C_h$	CRC = 0x569C
	1 Byte high	56_h	

Für die Antwort berechnen Sie die CRC-Prüfsumme wir folgt:

```
uint16_t crc;
crc = ModbusCRC(0x01, 0xFFFF);
crc = ModbusCRC(0x03, crc);
crc = ModbusCRC(0x02, crc);
crc = ModbusCRC(0x70, crc);
crc = ModbusCRC(0x38, crc);
printf("CRC is 0x%04x\r\n",crc);
```

Die Ausgabe dieses Programms ist: CRC is 0x569C.