SIEB & MEYER

Antriebssystem SD2

DNC-Objektzugriff

P-TD-0000345.2
2016-02-03

Copyright

Originalbetriebsanleitung, Copyright © 2016 SIEB & MEYER AG

Alle Rechte vorbehalten.

Diese Anleitung darf nur mit einer ausdrücklichen schriftlichen Genehmigung der SIEB & MEYER AG kopiert werden. Das gilt auch für Auszüge.

Marken

Alle in dieser Anleitung aufgeführten Produkt-, Schrift- und Firmennamen und Logos sind gegebenenfalls Marken oder eingetragene Marken der jeweiligen Firmen.

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
support@sieb-meyer.de
http://www.sieb-meyer.de

SIEB & MEYER Shenzhen Trading Co. Ltd.
Room 306, 3rd Floor, Building A1,
Dongjiaotou Industrial Area , Houhai Dadao,
Shekou, Nanshan District,
Shenzhen City, 518067
P.R. China

Tel.: +86 755 2681 1417 / +86 755 2681 2487
Fax: +86 755 2681 2967
sm.china.support@gmail.com
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
smasia@ms42.hinet.net
http://www.sieb-meyer.com

SIEB & MEYER USA
3975 Port Union Road
Fairfield, OH 45014
USA

Tel.: +1 513 563 0860
Fax: +1 513 563 7576
info@sieb-meyerusa.com
http://www.sieb-meyer.com

1 Allgemeines

Das DNC-Objektzugriffsprotokoll dient zur Parametrierung und Diagnose von SD2-Antrieben. Ein PC kann zur Diagnose und als Parametrierschnittstelle dienen und Daten über das DNC-Objektzugriffsprotokoll mit dem Antrieb austauschen.

Zur physikalischen Ankopplung des Antriebs werden folgende Bussysteme unterstützt:
- RS232
- RS485

Die Kommunikation mit dem SD2-Antrieb erfolgt nach dem Master/Slave-Verfahren. Der PC dient als Master, während der Antrieb als Slave agiert. Aufgrund dieses Verfahrens wird der Antrieb immer nur nach Aufforderung des Masters aktiv.

Der Austausch der Daten erfolgt in Form von Datentelegrammen. Der Master sendet ein Kommandotelegramm an den Slave, woraufhin der Slave ein Antworttelegramm an den Master zurücksendet. Der Datenaustausch über das Bussystem wird nur dann durchgeführt, wenn der Datenaustausch durch den Master eingeleitet wird.

Das DNC-Objektzugriffsprotokoll hält keine fest definierten Antwortzeiten ein. Aus diesem Grund sollte es nur zu Parametrierungs- und Diagnosezwecken des Antriebs eingesetzt werden.

1

2 Besonderheiten der RS232-Kommunikation

Beachten Sie die folgenden Erläuterungen bei der seriellen Kommunikation mit dem SD2.

2.1 Anschluss der seriellen Verbindung

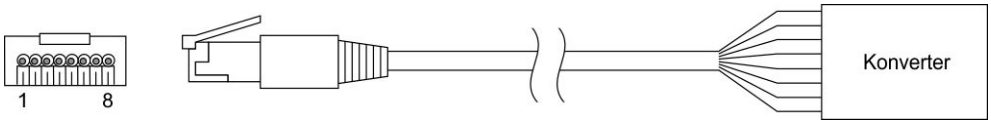
Mit Hilfe eines handelsüblichen PCs können Daten über die serielle Schnittstelle mit dem Antrieb ausgetauscht werden. Hierzu muss eine freie serielle Schnittstelle des PCs mit der RS232/RS485-Schnittstelle des Antriebs verbunden werden. Die Verbindung ist abhängig von dem jeweils verwendeten Antrieb.

2.1.1 RS232-Verbindung mit SD2

Verbinden Sie die RJ45-Buchse X3 an der Frontseite des SD2-Antriebs über einen RS232-RS485-Konverter mit einer freien seriellen Schnittstelle des PCs.

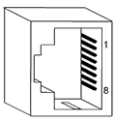
Anschlusskabel

- abgeschirmtes Rundkabel
- paarig verdrillt
- 8-poliger RJ45-Stecker ↔ offenes Ende



Pinbelegung auf dem SD2

- 8-polige RJ45-Buchse

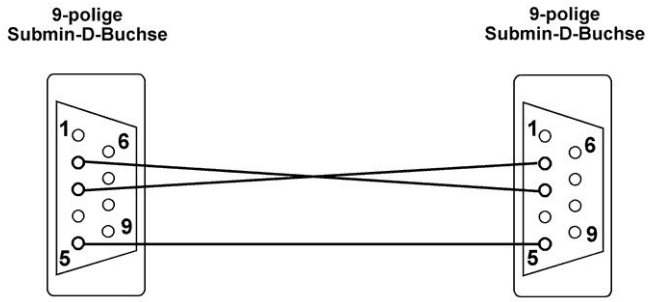
	Pin	E/A	Name	Bedeutung
	1		–	
	2		–	
	3	E/A	D+	RS485-Schnittstelle
	4		–	
	5		–	
	6	E/A	D−	RS485-Schnittstelle
	7	E/A	GND	Masse
	8	E/A	GND	Masse

2.1.2 RS232-Verbindung mit SD2S

Verbinden Sie den Stecker X19 (9-poliger Submin-D-Stecker) an der Frontseite des SD2S-Antriebs mit einer freien seriellen Schnittstelle des PCs (9-poliger Submin-D-Stecker).

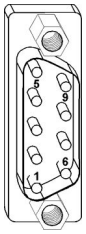
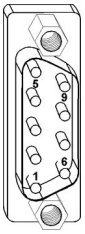
Anschlusskabel

- abgeschirmtes Rundkabel
- paarig verdrillt
- 9-polige Submin-D-Buchse ↔ 9-polige Submin-D-Buchse



Pinbelegung auf dem Gerät

- 9-poliger Submin-D-Stecker ↔ 9-poliger Submin-D-Stecker

SD2S			Name	Beschreibung	PC		
	Pin	E/A			E/A	Pin	
	1		–				
	2	E	RxD	Daten vom PC empfangen	A	3	
	3	A	TxD	Daten zum PC senden	E	2	
	4		–				
	5	E/A	GND	Masse	E/A	5	
	6		–				
	7		–				
	8		–				
	9		–				

2.2 Parametrierung der seriellen Schnittstelle

Folgende Einstellungen sind für die serielle Schnittstelle notwendig:

- Baudrate: 57600 Bit/s
- Datenbits: 8
- Parität: Keine
- Stoppbits: Eins

2.3 Timingverhalten

Nach Senden eines DNC-Kommandoblocks antwortet der Antrieb spätestens 250 µs nachdem er das letzte Byte des Kommandoblocks empfangen hat.

Beachten Sie, dass die Übertragungszeit zusätzlich von der zu übertragenden Datenmenge abhängig ist. Die PC-Software *drivemaster2* benutzt daher ein Timeout von 1 s, um sicher zu gehen, dass die serielle Verbindung wirklich unterbrochen ist.

3 Datentypen

Die Kommunikation mit dem SD2 erfolgt über DNC-Kommandos. Mit Hilfe dieser Kommandos können unter Verwendung von Speicherblöcken Daten mit dem Gerät ausgetauscht werden.

Ein Speicherblock ist in Abhängigkeit des DNC-Kommandos byteweise, wortweise oder 3-byteweise organisiert. Bei einem Speicherblock der Länge n gilt folgendes:

- <Byte 0, Wort 0, drei Byte 0> = Byte mit der niedrigsten Adresse
- <Byte n-1, Wort n-1, drei Byte n-1 > = Byte mit der höchsten Adresse

Somit muss bei der sequentiellen Übertragung der Blöcke das <Byte 0> zuerst und das <Byte n-1> zuletzt übertragen werden.

3.1 1-Byte-Datentypen

Ein Byte ist das kleinste Datenformat, das über ein DNC-Kommando übertragen werden kann. Es wird zwischen vorzeichenlosen und vorzeichenbehafteten 1-Byte-Datentypen unterschieden:

T36_S36_SHORT-CARD	vorzeichenlose 8-Bit-Zahl (0 … 255)
T36_S36_SHORTINT	vorzeichenbehaftete 8-Bit-Zahl (-128 … 127)

3.2 2-Byte-Datentypen

2-Byte-Datentypen werden wie folgt kodiert:

Byte 0	Byte 1						
Bit 0 … 7	Bit 8 … 15						

Es wird zwischen vorzeichenlosen und vorzeichenbehafteten Datentypen unterschieden:

T36_S36_CARDINAL	vorzeichenlose 16-Bit-Zahl (0 … 65535)
T36_S36_INTEGER	vorzeichenbehaftete 16-Bit-Zahl (-32768 … 32767)

3.3 4-Byte-Datentypen

4-Byte-Datentypen werden wie folgt kodiert:

Byte 0	Byte 1	Byte 2	Byte 3				
Bit 0 … 7	Bit 8 … 15	Bit 16 … 23	Bit 24 … 31				

Es wird zwischen vorzeichenlosen und vorzeichenbehafteten Datentypen unterschieden:

T36_S36_LONG-CARD	vorzeichenlose 32-Bit-Zahl (0 … 4.294.967.295)
T36_S36_LONGINT	vorzeichenbehaftete 32-Bit-Zahl (-2.147.483.647 … 2.147.483.647)

3.4 3-Byte-Datentypen

Die Geräte der Serie SD2 arbeiten intern zum Teil mit 3-Byte-Datenworten.

Byte 0	Byte 1	Byte 2					
Bit 0 … 7	Bit 8 … 15	Bit 16 … 23					

Da der PC diesen Datentyp nicht kennt, er jedoch von einigen DNC-Kommandos verwendet wird, muss er durch die bestehenden Datentypen abgebildet werden. Ein 3-Byte-Datenwort kann somit aus 3 Bytes vom Typ *T36_S36_SHORTCARD* bestehen. Zwei 3-Byte-Datenworte können durch drei Worte des Typs *T36_S36_CARDINAL* ersetzt werden. Und vier 3-Byte-Datenworte können durch drei Doppelworte des Typs *T36_S36_LONGCARD* abgebildet werden.

4 DNC-Kommandos

Die DNC-Kommunikation mit dem Antrieb besteht im Allgemeinen aus dem Austausch von Datenblöcken zwischen dem DNC-Master (PC oder SPS) und dem Slave (Antrieb).

Ein DNC-Kommando setzt sich wie folgt zusammen:
- Kommandoblock (wird vom Master an den Antrieb gesendet)
- Antwortblock (wird vom Antrieb an den Master zurückgesendet)

Ein DNC-Kommando wird immer vom Master eingeleitet. Der Antrieb kann nur auf das DNC-Kommando reagieren.

Für jedes DNC-Kommando haben die Kommando- und Antwortblöcke eine eigene Struktur und Bedeutung. Der Kommandoblock beschreibt dabei das DNC-Kommando und kann sogenannte Unterkommandos enthalten.

4.1 Allgemeiner Aufbau der Kommandoschnittstelle

Im folgenden wird der allgemeine Aufbau des Kommando- und des Antworttelegramms beschrieben.

4.1.1 Kommandoblock

Offset	Typ	Name	Beschreibung
0x00	T36_S36_SHORTCARD	zero	Startsignal für die Übertragung eines Kommandos: Wird immer auf Null gesetzt.
0x01	T36_S36_SHORTCARD	length	Länge des Protokolls in Bytes: Sie berechnet sich aus der Anzahl der übertragenden Daten, aber ausschließlich der führenden Null, der Prüfsumme und der Längenangabe. Die kleinste Länge ist 3.
0x02	T36_S36_SHORTCARD	dest	Ziel des Kommandoblocks: Es enthält die gewünschte Modulnummer plus 2.
0x03	T36_S36_SHORTCARD	source	Sender des Kommandos: Hier wird eine 1 für den PC oder die SPS eingetragen.
0x04	T36_S36_SHORTCARD	cmd	Kommandonummer des eigentlichen DNC-Kommandos
0x05	T36_S36_SHORTCARD Array mit maximal 48 Bytes Größe	data	Nutzdaten des Kommandos: Je nach Kommando können hier bis zu 48 Bytes, 24 Worte oder 16 3-Byte-Worte gesendet werden.
0x05 + length-3	T36_S36_SHORTCARD	check	Prüfsumme: Sie besteht aus dem Einer-Komplement der Summe des gesamten Datenblocks, außer der Prüfsumme selbst.

4.1.2 Antwortblock

Offset	Typ	Name	Beschreibung
0x00	T36_S36_SHORTCARD	zero	Startsignal für die Übertragung einer Antwort: Wird immer auf Null gesetzt.
0x01	T36_S36_SHORTCARD	length	Länge des Protokolls in Bytes: Sie berechnet sich aus der Anzahl der übertragenden Daten, aber ausschließlich der führenden Null, der Prüfsumme und der Längenangabe. Die kleinste Länge ist 3.

Offset	Typ	Name	Beschreibung
0x02	T36_S36_SHORTCARD	dest	Ziel des Antwortblocks: Hier wird eine 1 für den PC oder die SPS eingetragen.
0x03	T36_S36_SHORTCARD	source	Sender des Antwortblocks: Es enthält die Modulnummer des Senders plus 2.
0x04	T36_S36_SHORTCARD	cmd	Nummer des abgearbeiteten Kommandos: Es ist zu beachten, dass hier zusätzlich das höchstwertige Bit auf 1 gesetzt ist.
0x05	T36_S36_SHORTCARD Array mit maximal 48 Bytes Größe	data	Nutzdaten des Antwortblocks: Je nach Kommando können hier bis zu 48 Bytes, 24 Worte oder 16 3-Byte-Worte gesendet werden.
0x05 + length-3	T36_S36_SHORTCARD	check	Prüfsumme: Sie besteht aus dem Einer-Komplement der Summe des gesamten Datenblocks, außer der Prüfsumme selbst.

5 Adressierung der Geräte

Um die Antriebe von einer SPS oder einem eigenen PC-Programm aus ansteuern zu können, müssen die im DNC-Protokoll verwendeten Adressen bekannt sein. Die DNC-Antriebsadressen leiten sich aus der Moduladresse ab, die über den Adresswahlschalter am Gerät eingestellt wird. Über die Moduladresse kommuniziert die Software *drivemaster2* mit den Geräten.

Weitere Informationen zur korrekten Adressierung der Module über den Adresswahlschalter finden Sie in der Dokumentation „drivemaster2 - Bedienen“ im Kapitel „Kommuniaktion“.

Im DNC-Protokoll belegt jedes Gerät (Doppelantriebe sowie Einzelantriebe) immer zwei DNC-Adressen, d. h. Antrieb A wird unter der kleineren DNC-Adresse angesprochen, während Antrieb B unter der größeren Adresse angesprochen werden kann. Handelt es sich um einen Einzelantrieb, gibt es keinen Antrieb B und die höhere DNC-Adresse bleibt ungenutzt. Die DNC-Adressen 0 und 1 sind für den Master reserviert, somit beginnt die Nummerierung der DNC-Adressen bei 2.

Die DNC-Adresse eines Antriebs lässt sich mit folgender Formel aus der Stellung des Adresswahlschalters am Modul errechnen:

DNC-Adresse Antrieb A = (Adresswahlschalter × 2) + 2

DNC-Adresse Antrieb B = (Adresswahlschalter × 2) + 3

Die folgende Abbildung verdeutlicht den Zusammenhang zwischen der in *drivemaster2* verwendeten Antriebsadresse und der im DNC-Protokoll verwendeten Adresse:

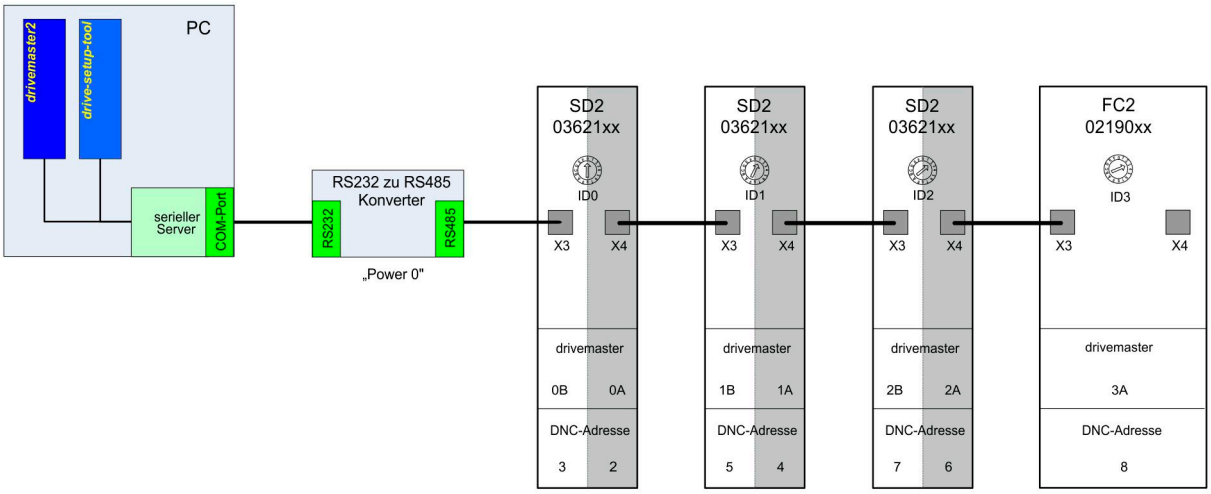


Abb. 1: Antriebsadressierung in drivemaster2 und im DNC-Protokoll

6 Objektzugriff

Über die Telegramme „Read Wide Object“ und „Write Wide Object“ kann auf Objekte zugegriffen werden. Jedes Objekt wird durch einen Index (Objektnummer) und einen Subindex (Objektsubnummer) identifiziert.

Die Objektnummer und weitere Daten zu den einzelnen Antriebsobjekten können Sie sich in der *drivemaster2*-Software anzeigen lassen (siehe Anhang, S. 23).

6.1 Read Wide Object

Mit dem Telegramm „Read Wide Object“ kann ein Objekt aus dem Antrieb gelesen werden.

Der Objektindex ist ein 16-Bit-Wert, der in den Bytes „index lo“ und „index hi“ gespeichert ist. Für Array- und String-Objekte steht ein Subindex zur Verfügung. Der Subindex wird als 32-Bit-Wert gespeichert.

Der Kommandoblock setzt sich wie folgt zusammen:

Name	Wert
zero	0
length	9
dest	Modulnummer plus 2
source	1 (für PC)
cmd	0x0d
index lo	niederwertiges Byte der Objektnummer
index hi	höherwertiges Byte der Objektnummer
subindex 0	Subindex-Bit [7…0]
subindex 1	Subindex-Bit [15…8]
subindex 2	Subindex-Bit [23…16]
subindex 3	Subindex-Bit [31…24]
check	checksum

check = 0xFF − (length + dest + source + cmd + index lo + index hi + subindex 0 +subindex 1 + subindex 2 + subindex 3)

Der folgende Antwortblock wird zurückgesendet:

Name	Wert
zero	0
length	5 + count
dest	1 (für PC)
source	Modulnummer plus 2
cmd	0x8d
count	Anzahl der gelesenen Bytes
error code	siehe Fehlercodes, S. 21
data	data 0
⋮	
data	data (count − 1)
check	Checksum

check = 0xFF – (length + dest + source + cmd + count + errorcode + $\sum_{i=count} data_{(i)}$)

count = 0…48

6.2 Write Wide Object

Mit dem Telegramm „Write Wide Object“ kann ein Objekt in den Antrieb geschrieben werden.

Der Objektindex ist ein 16-Bit-Wert, der in den Bytes „index lo“ und „index hi“ gespeichert ist. Für Array- und String-Objekte steht ein Subindex zur Verfügung. Der Subindex wird als 32-Bit-Wert gespeichert.

Der Kommandoblock setzt sich wie folgt zusammen:

Name	Wert
zero	0
length	10 + count
dest	Modulnummer plus 2
source	1 (für PC)
cmd	0x0e
index lo	niederwertiges Byte der Objektnummer
index hi	höherwertiges Byte der Objektnummer
subindex 0	Subindex-Bit [7…0]
subindex 1	Subindex-Bit [15…8]
subindex 2	Subindex-Bit [23…16]
subindex 3	Subindex-Bit [31…24]
count	Anzahl der zu schreibenden Bytes
data	data 0
⋮	
data	data (count − 1)
check	Checksum

check = 0xFF – (length + dest + source + cmd + index lo + index hi + subindex 0 +subindex 1 + subindex 2 + subindex 3 + count + $\sum_{i=count} data_{(i)}$)

count = 1…48

Der folgende Antwortblock wird zurückgesendet:

Name	Wert
zero	0
length	4
dest	1 (für PC)
source	Modulnummer plus 2
cmd	0x8e
error code	siehe Fehlercodes, S. 21
check	Checksum

check = 0xFF – (length + dest + source + cmd + errorcode)

6.3 Fehlercodes des Servicedatenkanals

Fehlercode	Beschreibung
0x00	Kein Fehler
0x81 oder 0x01	Toggle-Bit nicht nachgezogen
0x86 oder 0x06	CRC Fehler
0x87 oder 0x07	Kein freier Speicher
0x88 oder 0x08	Nicht erlaubter Zugriff auf ein Objekt
0x89 oder 0x09	Versucht ein Write-Only-Objekt zu lesen
0x8A oder 0x0A	Versucht ein Read-Only-Objekt zu beschreiben
0x8B oder 0x0B	Objekt nicht in der Objekt-Bibliothek vorhanden!
0x8C oder 0x0C	Reserviert
0x8D oder 0x0D	Reserviert
0x8E oder 0x0E	Allgemeiner Parameter-Inkompatibilitätsgrund
0x8F oder 0x0F	Allgemeine interne Inkompatibilität im Gerät
0x90 oder 0x10	Zugriff verweigert aufgrund eines Hardwarefehlers
0x91 oder 0x11	Datentyp nicht korrekt, Länge des Service-Parameters nicht korrekt
0x92 oder 0x12	Datentyp nicht korrekt, Länge des Service-Parameters zu hoch
0x93 oder 0x13	Datentyp nicht korrekt, Länge des Service-Parameters zu niedrig
0x94 oder 0x14	Subindex nicht vorhanden
0x95 oder 0x15	Wertebereich des Parameters überschritten (nur bei Schreibzugriff)
0x96 oder 0x16	Wert des geschriebenen Parameters zu hoch
0x97 oder 0x17	Wert des geschriebenen Parameters zu niedrig
0x98 oder 0x18	Maximalwert niedriger als Minimalwert
0x99 oder 0x19	Allgemeiner Fehler
0x9A oder 0x1A	Daten können nicht übertragen oder in der Applikation gespeichert werden.
0x9B oder 0x1B	Daten können aufgrund des Zustands der Steuerung nicht übertragen oder in der Applikation gespeichert werden.
0x9C oder 0x1C	Daten können aufgrund des zurückgesetzten Gerätes nicht übertragen oder in der Applikation gespeichert werden.
0x9D oder 0x1D	Dynamische Erzeugung des Objektverzeichnisses nicht möglich oder kein Objektverzeichnis vorhanden.
0x9E oder 0x1E	Lesezugriff verweigert
0x9F oder 0x1F	Schreibzugriff verweigert

7 Anhang

7.A Objektdaten anzeigen

Informationen zu den einzelnen Objekten eines Antriebs finden Sie über den Objektbrowser in der *drivemaster2*-Software (Registerkarte „Diagnose“).

Für die Objekte, die im Objektbrowser geladen sind, wird der Objektname, der Index, der aktuelle Wert und die Einheit angezeigt:

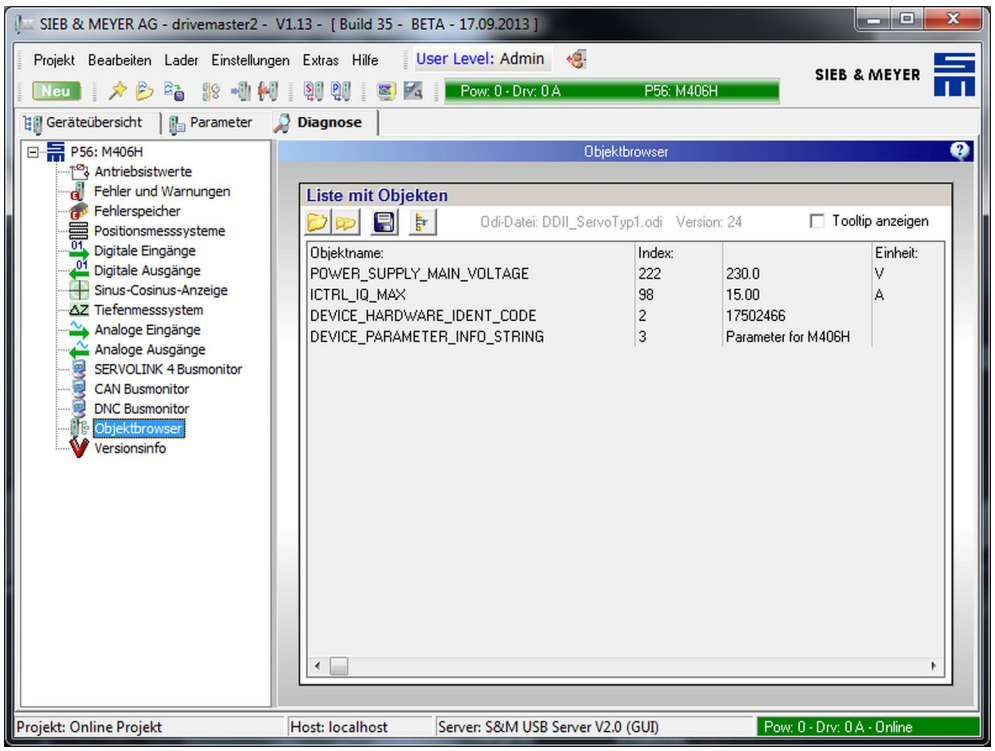


Abb. 2: Objektbrowser in drivemaster2

Der Objektauswahldialog zeigt im unteren Feld „Objektinformationen“ zusätzlich den Datentyp und einen Kommentar für das Objekt an, das auf der rechten Seite ausgewählt ist:

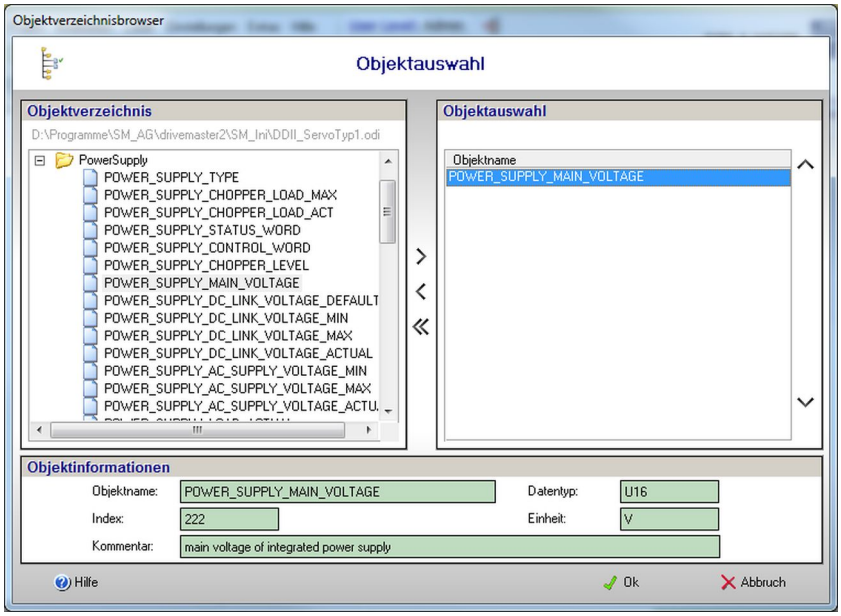


Abb. 3: Objektauswahl

7.A

7.B Beispiele

7.B.1 Drehzahl setzen und einschalten

Dieses Beispiel zeigt eine einfache Ansteuerung des Antriebs.

Voraussetzungen

Die folgenden Einstellungen/Status in der *drivemaster2*-Software sind Voraussetzung für dieses Beispiel:

- Als Steuerkanal ist entweder „Serielle Schnittstelle/RS485/USB“ oder „DNC 8 Byte Telegramm“ gewählt (siehe „Parameter → Antriebssteuerung → Steuerkanal“).
- Als Sollwertkanal ist entweder „Serielle Schnittstelle/RS485/USB“ oder „DNC 8 Byte Telegramm“ gewählt (siehe „Parameter → Antriebssteuerung → Sollwertkanal“).
- Der Antrieb meldet den Zustand „Status: Einschaltsperre, Softwarefunktion ‚Schnellhalt‘“ (siehe „Diagnose → Antriebsistwerte“).

7.B.1.1 Shutdown-Kommando an den Antrieb senden

Mit dem Shutdown-Kommando wird die Einschaltsperre freigeschaltet, die nach dem Booten des Antriebs aktiv ist. Das Objekt DEV_CTRL_CONTROL_WORD (Steuerwort) wird geschrieben.

Um das Shutdown-Kommando auszulösen, wird das Objekt mit dem **Wert 6** beschrieben. Die einzelnen Steuerkommandos sind in der Dokumentation „Antriebssystem SD2 - Gerätesteuerung“ beschrieben.

Das Objekt hat folgende Eigenschaften:

DEV_CTRL_CONTROL_WORD	
Index	68
Type	u16 (unsigned 16 Bit)
Wert	6

Send:

zero	length	dest	source	cmd	index lo	index hi	sub ID 0	sub ID 1	sub ID 2	sub ID 3	count	•••
0x00	0x0b	0x02	0x01	0x0e	0x44	0x00	0x00	0x00	0x00	0x00	0x02	

•••	data	data	check
	0x06	0x00	0x97

Der Wert für „count“ wird auf 2 gesetzt, da das Objekt ein 16-Bit-Datentyp ist und somit 2 Byte übertragen werden müssen. In „data“ werden dann die 2 Byte übertragen.

Der Antrieb antwortet mit folgender Sequenz:

Reply:

zero	length	dest	source	cmd	error code	check
0x00	0x04	0x01	0x02	0x8e	0x00	0x6a

Nun meldet der Antrieb in der Diagnose der *drivemaster2*-Software den Zustand „Status: Einschaltbereit“.

7.B.1.2 Drehzahlsollwert setzen

Der Drehzahlsollwert soll auf 1000 1/min gesetzt werden. Das Objekt SPG_TARGET_VELOCITY_VL_UUNIT wird geschrieben.

Das Objekt hat folgende Eigenschaften:

SPG_TARGET_VELOCITY_VL_UUNIT	
Index	395
Type	i32 (integer 32 Bit)
Wert	1000000 (Angabe in 0,001 1/min)

Send:

zero	length	dest	source	cmd	index lo	index hi	sub ID 0	sub ID 1	sub ID 2	sub ID 3	count	•••
0x00	0x0b	0x02	0x01	0x0e	0x8b	0x01	0x00	0x00	0x00	0x00	0x04	

•••	data	data	data	data	check
	0x40	0x42	0x0f	0x00	0xc2

Der Antrieb antwortet mit folgender Sequenz:

Reply:

zero	length	dest	source	cmd	error code	check
0x00	0x04	0x01	0x02	0x8e	0x00	0x6a

7.B.1.3 Regler Einschalten

Das Objekt DEV_CTRL_CONTROL_WORD wird mit dem **Wert 7** beschrieben.

Send:

zero	length	dest	source	cmd	index lo	index hi	sub ID 0	sub ID 1	sub ID 2	sub ID 3	count	•••
0x00	0x0b	0x02	0x01	0x0e	0x44	0x00	0x00	0x00	0x00	0x00	0x02	

•••	data	data	check
	0x07	0x00	0x96

Der Antrieb antwortet mit folgender Sequenz:

Reply:

zero	length	dest	source	cmd	error code	check
0x00	0x04	0x01	0x02	0x8e	0x00	0x6a

Nun meldet der Antrieb in der Diagnose der *drivemaster2*-Software den Zustand „Status: Eingeschaltet“.

7.B.1.4 Betrieb freigeben

Das Objekt DEV_CTRL_CONTROL_WORD wird mit dem **Wert 15** beschrieben.

Send:

zero	length	dest	source	cmd	index lo	index hi	sub ID 0	sub ID 1	sub ID 2	sub ID 3	count	•••
0x00	0x0b	0x02	0x01	0x0e	0x44	0x00	0x00	0x00	0x00	0x00	0x02	

•••	data	data	check
	0x0f	0x00	0x8e

Der Antrieb antwortet mit folgender Sequenz:

Reply:

zero	length	dest	source	cmd	error code	check
0x00	0x04	0x01	0x02	0x8e	0x00	0x6a

Nun meldet der Antrieb in der Diagnose der *drivemaster2*-Software den Zustand „Status: Betrieb freigegeben“ und der Motor dreht.

7.B.1.5 Betrieb sperren

Zum Anhalten wird der **Wert 7** in das Steuerwort geschrieben (siehe Abschnitt 7.B.1.3 „Regler Einschalten“, S. 26).

7.B.1.6 Regler ausschalten

Zum Ausschalten wird der **Wert 6** in das Steuerwort geschrieben (siehe Abschnitt 7.B.1.1 „Shutdown-Kommando an den Antrieb senden“, S. 25).

7.B.2 Drehzahlistwert auslesen

Das Objekt VCTRL_VELOCITY_ACTUAL_VALUE_UUNIT wird gelesen.

Das Objekt hat folgende Eigenschaften:

VCTRL_VELOCITY_ACTUAL_VALUE_UUNIT	
Index	398
Type	i32 (integer 32 Bit)

Send:

zero	length	dest	source	cmd	index lo	index hi	sub ID 0	sub ID 1	sub ID 2	sub ID 3	check
0x00	0x09	0x02	0x01	0x0d	0x8e	0x01	0x00	0x00	0x00	0x00	0x57

Der Antrieb antwortet mit folgender Sequenz:

Reply:

zero	length	dest	source	cmd	count	error code	data 0	data 1	data 2	data 3	check
0x00	0x09	0x01	0x02	0x8d	0x04	0x00	0x7f	0x50	0x0f	0x00	0x84

Die aktuelle Istdrehzahl liegt also bei 0x000f507f = 1003647 ≈ 1004 1/min.

7.B

7.B.3 Statuswort auslesen

Das Objekt DEV_CTRL_STATUS_WORD wird gelesen.

Das Objekt hat folgende Eigenschaften:

DEV_CTRL_STATUS_WORD	
Index	67
Type	u16 (unsigned 16 Bit)

Send:

zero	length	dest	source	cmd	index lo	index hi	sub ID 0	sub ID 1	sub ID 2	sub ID 3	check
0x00	0x09	0x02	0x01	0x0d	0x43	0x00	0x00	0x00	0x00	0x00	0xa3

Der Antrieb antwortet mit folgender Sequenz:

Reply:

zero	length	dest	source	cmd	count	error code	data 0	data 1	check
0x00	0x09	0x01	0x02	0x8d	0x02	0x00	0x37	0x66	0x84

Die aktuelle Statuswort ist 0x6637.

7.B.4 Parametersatzidentifikation auslesen

Das Objekt DEVICE_PARAMETER_IDENT_STRING wird gelesen.

Das Objekt hat folgende Eigenschaften:

DEVICE_PARAMETER_IDENT_STRING	
Index	22
Type	String

Send:

zero	length	dest	source	cmd	index lo	index hi	sub ID 0	sub ID 1	sub ID 2	sub ID 3	check
0x00	0x09	0x02	0x01	0x0d	0x16	0x00	0x00	0x00	0x00	0x00	0xd0

Der Antrieb antwortet mit folgender Sequenz:

Reply:

zero	length	dest	source	cmd	count	error code	data 0	data 1	data 2	data 3	data 4	•••
0x00	0x26	0x01	0x02	0x8d	0x21	0x00	0x20	0x54	0x65	0x73	0x74	
•••	data 5	data 6	data 7	data 8	data 9	data 10	data 11	data 12	data 13	data 14	data 15	•••
	0x20	0x4d	0x6f	0x74	0x6f	0x72	0x00	0x00	0x00	0x00	0x00	
•••	data 16	data 17	data 18	data 19	data 20	data 21	data 22	data 23	data 24	data 25	data 26	•••
	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	
					•••	data 27	data 28	data 29	data 30	data 31	data 32	check
						0x00	0x00	0x00	0x00	0x00	0x00	0x37

In „data“ wird jetzt der String in folgendem Format ausgegeben:

data 0 — Länge des Strings
Die Angabe 0x20 bedeutet, dass der String 32 Zeichen lang ist. Wurden beim Parametrieren weniger als die maximale Anzahl von 32 Zeichen eingegeben, so ist der String Null-terminiert.

data 1 – 32 — Ausgabe der einzelnen Zeichen im ASCII Format

Der ausgelesen String für die Parametersatzidentifikation ist somit „Test Motor“.