Betriebsanleitung

ZK-USB-CAN-1

Betriebsanleitung Version: 1.0.0

Inhalt

1 Einleitung

ZK-USB-CAN-1 ist ein USB-zu-CAN-Adapter mit galvanischer Trennung zwischen USB- und CAN-Schnittstelle. Er ermöglicht eine Anbindung an CAN-Netzwerke. Durch ein kompaktes Kunststoffgehäuse eignet er sich auch für den mobilen Einsatzbereich.

Die Stromversorgung erfolgt über den USB 2.0 Typ-B-Stecker. Die CAN-Schnittstelle kann mit einer geschirmten RJ45-Buchse oder einem 4-poligen Leiterplattenanschluss verbunden werden. Der Adapter unterstützt die USB 2.0 Full-Speed-Schnittstelle und CAN 2.0.

ZK-USB-CAN-1 verfügt auch über eine schaltbare Bus-Terminierung sowie über zwei Status-LEDs (USB und CAN).

1.1 Versionshinweise

Version Dokument	Datum	Änderungen	Version Firmware
1.0.0	03/2021	Veröffentlichung	v2110

1.2 Urheberrecht, Kennzeichnung und Kontakt

CE

Nanotec Electronic GmbH & Co. KG

Kapellenstraße 6

85622 Feldkirchen

Deutschland

Tel.+49 89 900 686-0

Fax +49 89 900 686-50

www.nanotec.de

1.3 Bestimmungsgemäße Verwendung

ZK-USB-CAN-1 findet Verwendung als Komponente von Antriebssystemen in vielfältigen Industrieanwendungen, wo eine Anbindung eines Computers an den CAN-Bus über USB benötigt wird.

Verwenden Sie das Produkt bestimmungsgemäß innerhalb der durch die technischen Daten definierten Grenzen (siehe Elektrische Eigenschaften und technische Daten) und unter den freigegebenen Umgebungsbedingungen.

Unter keinen Umständen darf dieses Nanotec-Produkt als Sicherheitsbauteil in ein Produkt oder eine Anlage integriert werden. Alle Produkte, in denen eine von Nanotec hergestellte Komponente enthalten ist, müssen bei der Übergabe an den Endnutzer entsprechende Warnhinweise und Anweisungen für eine sichere Verwendung und einen sicheren Betrieb aufweisen. Alle von Nanotec bereitgestellten Warnhinweise müssen unmittelbar an den Endnutzer weitergegeben werden.

1.4 Gewährleistung, Haftungsausschluss

Nanotec haftet nicht für Schäden und Betriebsstörungen, die durch Montagefehler, Nichtbeachtung dieses Handbuchs oder unsachgemäße Reparaturen entstehen. Verantwortlich für Auswahl, Betrieb und Verwendung von Nanotec-Produkten sind Anlagenkonstrukteur, Betreiber und Endnutzer. Nanotec verantwortet keine Integration des Produkts ins Endsystem.

Es gelten unsere allgemeinen Geschäftsbedingungen auf www.nanotec.de.

HINWEIS

Änderungen oder Umbauten des Produkts sind nicht zulässig.

1.5 Zielgruppe und Qualifikation

Das Produkt und diese Dokumentation richten sich an technisch geschulte Fachkräfte wie:

- Software-Entwickler
- Entwicklungsingenieure
- Monteure/Servicekräfte
- Applikationsingenieure

Nur Fachkräfte dürfen das Produkt installieren und in Betrieb nehmen. Fachkräfte sind Personen, die

- eine entsprechende Ausbildung und Erfahrung im Umgang mit Feldbussystemen und elektrostatisch gefährdeten Bauteilen haben,
- den Inhalt dieses technischen Handbuchs kennen und verstehen,
- die geltenden Vorschriften kennen.

1.6 EU-Richtlinien zur Produktsicherheit

Folgende EU-Richtlinien wurden beachtet:

- RoHS-Richtlinie (2011/65/EU, 2015/863/EU)
- EMV-Richtlinie (2014/30/EU)

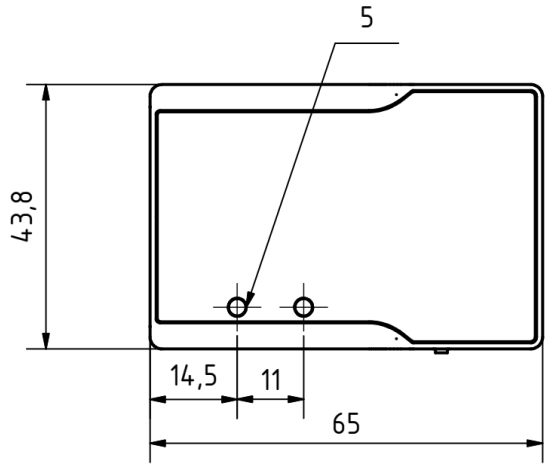
1.7 Zahlenwerte

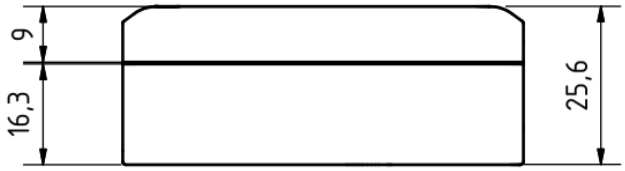
Zahlenwerte werden grundsätzlich in dezimaler (*dec*) Schreibweise angegeben. Sollte eine hexadezimale (*hex*) Notation verwendet werden, wird das mit *0x* markiert.

2 Technische Daten und Anschlussbelegung

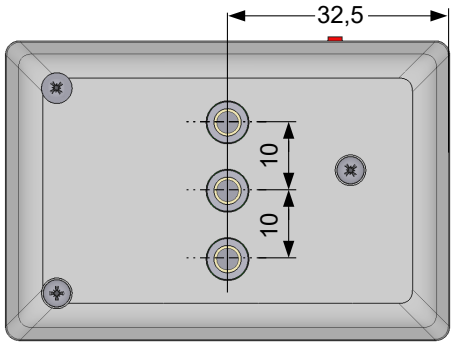
2.1 Maßzeichnungen und Montagemöglichkeiten

Alle Maße sind in Millimetern.





Sie können den Adapter mit 1 bis 3 M4-Schrauben befestigen. Es sind dafür 3 M4x6-Gewindebohrungen vorgesehen:



Alternativ können Sie auch eine Hutschienenklammer anschrauben und den Adapter in einen Schaltschrank einbauen.

2.2 Umgebungsbedingungen

Umgebungsbedingung	Wert	Einheit
Umgebungstemperatur (Betrieb)	-10... 70	°C
Umgebungstemperatur (Lagerung)	-25... 85	°C
relative Luftfeuchtigkeit (nicht kondensierend)	0...95	%
Schutzklasse nach EN/IEC 60529	IP 40	

2.3 Elektrische Eigenschaften und technische Daten

Eigenschaft	Wert	Einheit	Bemerkung
Stromversorgung	5	V DC	via USB
Isolationsspannung	1000 / 500	V DC / V AC	für 1 Minute
CAN-Baudrate	bis 1000	KBaud	

2.4 LED-Signalisierung

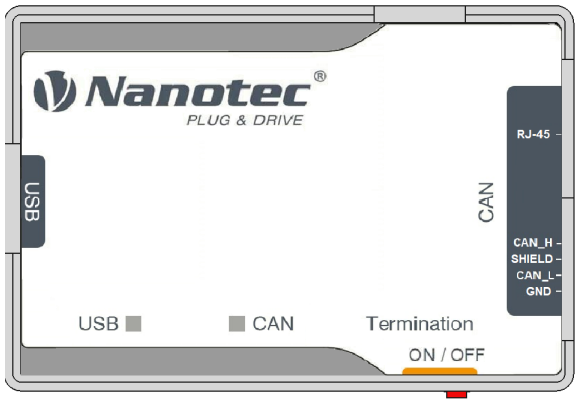
Wenn USB initialisiert ist, leuchtet die USB-Status-LED grün und das Gerät ist betriebsbereit. Wenn die Kommunikation läuft, blinkt die LED grün. Wenn ein Fehler auftritt, leuchtet die LED rot.

Die CAN-Status-LED hat ähnliche Funktionen. Wenn der CAN-Bus initialisiert ist, leuchtet sie grün. Wenn die Kommunikation läuft, blinkt sie grün. Wenn ein Fehler auftritt, leuchtet die LED rot.

LED-Verhalten	USB-/CAN-Status
dauerhaft rot	Fehler
dauerhaft grün	initialisiert
blinkend grün	laufende Kommunikation

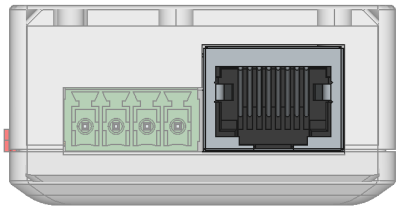
2.5 Anschlussbelegung

Übersicht



CAN-Anschluss 1 — RJ-45

Typ: RJ45-Buchse

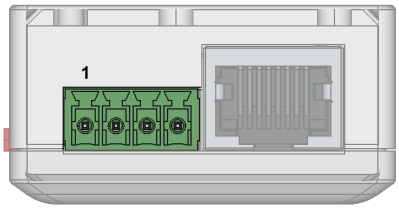


Pin	Funktion	Bemerkung
1	CAN_H	CAN-High
2	CAN_L	CAN-Low
3	CAN_GND	CAN-Ground
4	-	-
5	-	-
6	CAN_SHLD	verbunden mit Gehäuse
7	-	-
8	-	-
Gehäuse	Schirm	verbunden mit Pin 6

CAN-Anschluss 2 — Leiterplattenklemme

- Typ: Phoenix Contact MCV 1,5/ 4-G-3,5
- Gegenstecker (im Lieferumfang enthalten): Phoenix Contact FMC 1,5/ 4-ST-3,5 (oder äquivalent)

Pin 1 ist mit einer 1 markiert.

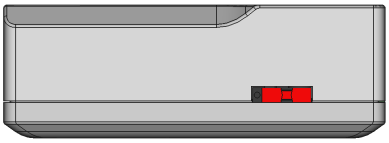


Pin	Funktion	Bemerkung
1	CAN_GND	CAN-Ground
2	CAN_L	CAN-Low
3	CAN_SHLD	Anschluss für die Schirmung, verbunden mit dem Gehäuse von RJ45
4	CAN_H	CAN-High

Terminierungswiderstand

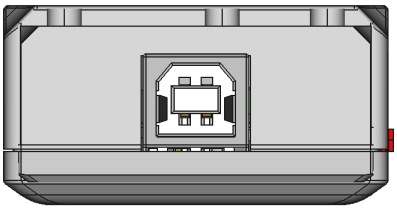
Typ: Schiebschalter

Dieser Schiebschalter schaltet die Terminierung von 120 Ω zwischen CAN_H und CAN_L des CAN-Busses zu oder ab. Die Schaltereinstellung "links" schaltet die Terminierung zu.



USB-Anschluss

Typ: USB 2.0, Typ-B-Stecker. Ein passendes USB-Kabel ist im Lieferumfang enthalten



3 Treiber und Adapter installieren

In diesem Kapitel lesen Sie, wie Sie den Adapter an einen PC mit Windows 10 anschließen.

1. Schließen Sie den Adapter über USB an den PC an. Der Treiber wird automatisch installiert und der Gerätebus wird als COM-Port erkannt.
2. Um die Nummer des zugewiesenen COM-Port zu finden:
   a. Klicken Sie in der Taskleiste auf das Suchsymbol.
   b. Suchen Sie nach "Device Manager" und wählen Sie ihn aus.
   c. Gehen Sie zu "Ports (COM & LPT)".
   Es erscheint ein USB Serial Device mit der zugewiesenen COM-Nummer (COM "X").

4 Protokoll-Beschreibung

In diesem Kapitel werden alle Kommandos aufgelistet, die das Protokoll des Adapters unterstützt. Die Kommandos können Sie (beispielsweise mithilfe eines Terminal-Programms) über den COM-Port schicken.

Das Protokoll basiert auf die Spezifikation *CiA 309: Access from other networks - Part 3: ASCII mapping* .

Alle Kommandos werden bestätigt. Sie können jedes Kommando mit einer Sequenz-Nummer (UNSIGNED32) beginnen, die dann in der Response enthalten ist.

Beispiel

```
// Request: Get name info, sequence=[1234]
[1234] info name
// Response: product name
[1234] name: ZK-USB-CAN-1
```

Der [sequence] darf aus Kompatibilitätsgründen das Token [net] folgen, dieses wird aber vom Adapter ignoriert.

4.1 Konfiguration und Initialisierung

4.1.1 Get Device Information: Version, State, Ticks, Name

Mit diesem Kommando können Sie die Produktinformationen auslesen und beispielsweise die Verbindung initial testen.

1. Info Version
   Syntax: **”info version”**
   Mit diesem Kommando fordern Sie die Version des CANopen-Gateway-Servers an.

   Beispiel

   ```
   // Request: Get version info
   info version
   // Response:
   620 690 2110.0 352977019 0 00.00 00.00
   ```

   - 620: *Vendor ID*
   - 690: *Product Code*
   - 2110.0: Firmware-Version, Jahr 2021, Woche 10
   - 352977019: Seriennummer
2. Info State
   Anforderungszustand des CAN-Bus-Zustandes.
   Syntax: **"info state"**
   Mit diesem Kommando fragen Sie den Status des CAN-Busses ab.

   Beispiel

   ```
   // Request: Get baudrate state info
   info state
   // Response: -1 if not initialized, >=0 -> baudrate; 0 = 1000kB
   state:0
   ```
3. Info Ticks

Syntax: **”info tick”**
liefert die Anzahl der Ticks (Millisekunden seit dem Einschalten) zurück

Beispiel

```
// Response: Get tick info
info tick
// Response: tick = 6940278
tick:6940278
```

4. Info Name
   Syntax: **”info name”**
   liefert den Produktnamen zurück

Beispiel

```
// Request: Get name info
info name
// Response: product name
name: ZK-USB-CAN-1
```

4.1.2 Read Hardware Version

Syntax:**0 r 0x1009 0 vs**

Beispiel

```
// Read adapter hardware version: NodeId=0; Index=0x1009;
 Subindex=0:
// Data Type= visual string
0 r 0x1009 0 vs
// SDO Read Response: hardware version= W003
"W003"
```

4.1.3 Read Firmware Version

Syntax:**0 r 0x100A 0 vs**

Beispiel

```
// Read adapter firmware version: NodeId=0; Index=0x100A;
 Subindex=0:
// Data Type= visual string
0 r 0x100A 0 vs
// SDO Read Response: firmware version= ZKUSBCAN1-FIB-v2110-
B123456
"ZKUSBCAN1-FIB-v2110-B123456"
```

4.1.4 Read Serial Number

Syntax:**0 r 0x4040 0 vs**

Beispiel

```
// Read adapter serial number: NodeId=0; Index=0x4040; Subindex=0:
// Data Type= visual string
0 r 0x4040 0 vs
// SDO Read Response: serial number= B123456 21/10-0123
"B123456 21/10-0123"
```

4.1.5 Gateway initialisieren

Syntax: "**[sequence] [net] init <baudrate>**"

Legt die Baudrate fest, mit welcher der CAN-Gateway-Server arbeitet. Die Baudrate kann als Wert oder Tabellenindex der Standard-CANopen-Tabelle angegeben werden.

Index	Baudrate
-1	Stop Bus
0	1000
1	800
2	500
3	250
4	125
5	100
6	50
7	20
8	10

HINWEIS

Autobaud wird nicht unterstützt.

Beispiel

```
// Request: Initialize CAN-Baudrate 1000 Kbits
init 0
// Response: "OK"
OK
```

4.1.6 Command set: Node-ID, Netzwerk, Timeout

Mit diesen Befehlen können Sie Node-ID, Netzwerk und SDO-Timeout einstellen:

1. Syntax: "**[sequence] [net] set node <value>**"
   Stellt die Default-Node-ID ein, die bei jedem Kommando mitzusenden ist, wenn sonst keine Node-ID im Kommando enthalten ist.

Beispiel

```
// Request:  Set CAN-NodeId = 127
set node 127
```

```
// Response: "OK"
OK
```

2. Syntax: "**[sequence] [net] set network <value>**"
   Hier stellen Sie das Default-Netzwerk ein (nur aus Kompatibilitätsgründen)..

Beispiel

```
// Request: Set CAN-Network = 2
set network 2
// Response: "OK"
OK
```

3. SDO Timeout
   Syntax: "**[sequence] [net] node] set sdo_timeout <ms>**"
   Mit diesem Kommando stellen Sie den SDO-Timeout ein.

Beispiel

```
// Request: Set timeout = 1ms
set sdo_timeout 1
// Response: "OK"
OK
```

4.1.7 Start Node (NMT)

Syntax: "**[sequence] [net] node] start**"

Mit diesem Kommando schalten Sie das CAN-Netzwerk in den Zustand "Operational", in welchem die PDO-Übertragung stattfindet.

Beispiel

```
// Request: Send Start operational command to NodeId = 1
1 start
// Response: "OK"
OK
```

```
// Incoming CAN-Message with CobId = 0x701; Data Length = 1;
// Data = 0x5
:>701 1 5
```

```
// Incoming CAN-Message with CobId = 0x281; Data Length = 4;
// Data = 0 0 0 0
:>281 4 0 0 0 0
```

```
// Incoming CAN-Message with CobId = 0x381; Data Length =  1;
// Data = 0 0
:>381 2 0 0
```

```
// Incoming CAN-Message with CobId = 0x481; Data Length = 4;
// Data = 0 0 0 0
:>481 4 0 0 0 0
```

4.1.8 Stop Node (NMT)

Syntax: **"[[sequence] [net] node] stop"**

Mit diesem Kommando schalten Sie das CAN-Netzwerk in den Zustand "Stop".

Beispiel

```
// Request: Send stop command to NodeId = 1
1 stop
// Response: "OK"
OK
```

```
// Incoming CAN-Message with CobId = 0x701; Data Length = 1;
// Data = 4
:>701 1 4
```

4.1.9 Set node Pre-Operational (NMT)

Syntax: **"[[sequence] [net] node] preop[erational]"**

Mit diesem Kommando schalten Sie das CAN-Netzwerk in den Zustand "Pre-Operational", in welchem Sie das PDO-Mapping verändern können.

Beispiel

```
// Request: Set NodeId = 1 to pre-operational
1 preop
// Response: "OK"
OK
```

```
// Incoming CAN-Message with CobId=0x701; Data Length=0x1;
 Data=0x7F
:>701 1 7F
```

4.1.10 Reset node (NMT)

Syntax: **"[[sequence] [net] node] reset node"**

Mit diesem Kommando reseten Sie das CAN-Netzwerk.

Beispiel

```
// Request: Send Reset Node to NodeId = 1
1 reset node
// Response: "OK"
OK
```

```
// Incoming CAN-Message with CobId=0x701; Data Length=0x1;
 Data=0x0
:>701 1 0
```

4.1.11 Reset communication (NMT)

Syntax: **"[[sequence] [net] node] reset comm"**

Beispiel

```
// Request: Send Reset communication to NodeId = 1
1 reset comm
// Response: "OK"
OK
```

4.1.12 Command for message format

Syntax: **set msg_format [n]**

Dabei ist [n] 0 (Standardformat) oder 1:

```
Format (0)
// All return strings are hex values!
:>601 8 40 63 20 0 0 0 0 0
:>581 8 43 63 20 0 1 0 0 0
```

```
Format (1)
// All return strings are hex values!
:>0x00000601 8 0x40 0x63 0x20 0x00 0x00 0x00 0x00 0x00
:>0x00000581 8 0x43 0x63 0x20 0x00 0x01 0x00 0x00 0x00
```

Beispiel

```
// Request: Set the format of message to 0
set msg_format 0
// Response: "OK"
OK
```

4.1.13 Command for CAN mode

Syntax: **set can_mode [m]**

Dabei ist [m] 0…3:

- 0 - (Standard) normaler CAN-Modus
- 1 - Loopback-Modus
- 2 - Silent-Modus
- 3 - Loopback mit Silent-Modus

Neue Moduseinstellungen werden nach der Neuinitialisierung des CAN-Busses (Kommando init) wirksam.

Beispiel

```
// Request: Set CAN mode = 1 (Loopback mode)
set can_mode 1
// Response: "OK"
OK
```

4.1.14 Command for: Filters, Filter ID, Filter Mask

Sie können Filter verwenden, um die eingehenden CAN-Nachrichten zu filtern.

1. Command for setting Filter
   Syntax: **`set filterX [S]`**
   Dabei ist:
   - X - 0..7 - Filternummer
   - S - 0..1 - 0 Filter deaktivieren, 1 Filter aktivieren

   Beispiel

   ```
   // Request: Enable filter 0
   set filter0 1
   // Response: "OK"
   OK
   ```

2. Command for setting Filter ID
   Syntax: **`set filter idX [id]`**
   Dabei ist:
   - X - 0..7 - Filternummer
   - [id] - Wert für den ID-Abgleich

   Die Bits der Filter-ID werde direkt mit den Bits der ID der eingehenden Nachricht abgeglichen.

   Beispiel

   ```
   // Request:
   set filter_id0 0x581
   // Response: "OK"
   OK
   ```

3. Command for setting Filter Mask
   Syntax: **`filter_maskX [id_mask]`**
   Dabei ist:
   - X- 0..7 - Filternummer
   - [id_mask] - Bitmaske für den ID-Abgleich

   Die Bitmaske definiert, welche Bits der ID berücksichtigt werden.

   Beispiel

   ```
   // Request: Set filter MaskId = 0xFFF
   set filter_mask0 0xFFF
   // Response: "OK"
   OK
   ```

Beispiel

```
set filter_id0 0x07D
set filter_mask0 0x0FF
set filter0 1
```

Nur Nachrichten werden angezeigt, deren niedrigstwertigen Bits der ID 0x7D entsprechen.

Nach der Aktivierung des Filters, würden beispielsweise folgende Nachrichten angezeigt:

```
:>17D 8 0 0 0 0 0 0 0 0
:>47D 4 0 1 1 0
```

4.1.15 Command Notification

Syntax: **`set notification [n]`**

Dabei ist [n] 0…3:

- 0 - Es wird keine Anzeige für RAW-CAN-Nachrichten empfangen.
- 1 (Standard) - Es werden nur empfangene RAW-CAN-Nachrichten angezeigt, die sich nicht auf den aktuellen SDO-Transfer beziehen.
- 2 - Alle empfangenen RAW-CAN-Nachrichten werden angezeigt.
- 3 - Alle empfangenen und gesendeten RAW-CAN-Nachrichten werden angezeigt.

Beispiel

```
// Request: Set notification = 0
set notification 0
// Response: “OK”
OK
```

4.1.16 Command for CAN Error list

Syntax: **`info CAN_ERR`**

Liefert eine Nachricht im Format CAN_ERR: [REC] [TEC] [LEC] [BOFF] [EPVF] [EWGF]

Dabei ist:

- [REC] - Fehlerzähler (Receive Error)
- [TEC] - niedrigstwertiges Byte des Fehlerzählers (Transmit Error)
- [LEC] - Letzter Fehlercode:
  - 0: No error
  - 1: Stuff Error
  - 2: Form Error
  - 3: Acknowledgment Error
  - 4: Bit recessive Error
  - 5: Bit dominant Error
  - 6: CRC Error
  - 7: Reserved
- [BOFF] - CAN Bus OFF Flag (0 oder 1)
- [EPVF] - CAN Bus Error Passive Flag (0 oder 1)
- [EWGF] - CAN Bus Error Warning Flag (0 oder 1)

Beispiel

```
// Request: Get info for a CAN error
info CAN_ERR
//Response CAN_ERR: [REC] [TEC] [LEC] [BOFF] [EPVF] [EWGF]
CAN_ERR: 0 0 0 0 0 0
```

4.2 Kommandos zum Lesen und Schreiben

Die folgende Tabelle zeigt alle unterstützen Datentypen und das in den Kommandos verwendete Format.

Datentyp	Format
SIGNED8	i8
SIGNED16	i16
SIGNED32	i32
UNSIGNED8	u8
UNSIGNED16	u16
UNSIGNED32	u32
Octet String (base64 encoded data)	os
Domain (base64 encoded data)	d
Visual String (siehe nachfolgenden Abschnitt)	vs

Visual Strings

Sie können *Visual Strings* innerhalb Anführungszeichen verwenden:

`"vs"`

Escape-Sequenzen innerhalb von *Visual Strings* werden ebenfalls unterstützt:

Escape Sequence	Format
Backslash	\\ - character \
Single Quotation Mark	\' - character '
Double Quotation Mark	\" - character "
Question Mark	\? - character ?

Sie können auch einen der folgenden *Special Control Characters* verwenden:

Control Character	Format
Bell	\a
Backspace	\b
Form Feed	\f
Line Feed	\n
Carriage Return	\r
Horizontal Tab	\t
Vertical Tab	\v
Oktalzahl von ASCII-Zeichen	\xNNN,wo NNN die Oktalzahl des ASCII-Zeichens, das Sie in die *Escape-Sequenz* einfügen wollen

4.2.1 Command "r" (SDO Read Request)

Syntax: **`[[sequence] [net] node] r <multiplexer> <datatype>`**

Beispiel

```
// SDO Read Request: NodeId=1; Index=0x1000; Subindex=0:
// Data Type=UInt32
1 r 0x1000 0 u32
// SDO Read Response: Value = 393618 = 0x00060192
393618
```

4.2.2 Command "w" (SDO Write Request)

Syntax: **`[[sequence] [net] node] w <index> <subindex> <datatype> <value>`**

Beispiel

```
// SDO Write Request: NodeId=1; Index=0x1016; Subindex=0:
// Data Type=UInt32, Value = 100
1 w 0x60FE 1 u32 100
// Response: "OK"
OK
```

4.2.3 Command "wl" (Initiate download SDO command for large data)

Syntax: **"`[sequence] [net] [node] wl <multiplexer> <command-specifier> <datatype> <overall_size> <value>`"**

Dieses Kommando wird zum Schreiben von großen Datenmengen verwendet und ist direkt mit dem Befehl "wlseg" verbunden (für weitere Details siehe *CiA 309:Access from other networks - Part 3: ASCII mapping*).

4.2.4 Command "rl" (Initiate upload SDO command for large data)

Syntax: **"`[sequence] [net] [node] rl <multiplexer>`"**

Dieses Kommando wird zum Lesen von großen Datenmengen verwendet und ist direkt mit dem Befehl "NEXT" verbunden (für weitere Details siehe *CiA 309:Access from other networks - Part 3: ASCII mapping*).

4.2.5 Command "wlb" (Initiate download SDO command for large data)

Syntax: **"`[sequence] [net] [node] wlb <multiplexer> <command-specifier> <datatype> <overall_size> <value>`"**

Dieses Kommando funktioniert genauso wie "wl", aber mit SDO-Blocktransfer.

4.2.6 Command "rlb" (Initiate upload SDO command for large data)

Syntax: **"`[sequence] [net] [node] rlb <multiplexer>`"**

Dieses Kommando funktioniert genauso wie "rl", aber mit SDO-Blocktransfer.

4.2.7 Command for SDO-Abort

Syntax: **`[sequence] [net] sdo_abort [force]`**

Bricht die SDO-Übertragung mit den Kommandos "wl(b)" und "rl(b)" ab, mit dem SDO-Abort-Code 0x05000000. Mit `[force]` wird ein SDO-Abort-Request gesendet, auch wenn im Moment keine SDO-Übertragung stattfindet.

Beispiel

```
// Request: SDO abort
sdo_abort
// Response: “OK”
OK
```

4.2.8 Command "rm" (remote request)

Syntax: **`[sequence] rm <cobid>`**

Beispiel

```
// Request: Send remote request
rm 0x181
// Response: “OK”
OK
```

4.2.9 Command "wm" (write CAN message)

Syntax: **`[sequence] [net] wm <cobid> <dataLength> <value1>[..<value8>]`**

Beispiel

```
// Request: CobId=0x00;
wm 0 2 2 0
// Response: “OK”
OK
```

```
// Incoming CAN-Message with CobId=0x701; Data Length=1; Data=0x4
:>701 1 4
```

4.2.10 Command ":<" (send CAN message)

Syntax: **`[sequence] [net] :< <cobid> <dataLength> <value1>[..<value8>]`**

Alle Werte sind in hexadezimaler Schreibweise, ohne *0x* vor de Zahl.

Beispiel

```
// Request: CobId=0x1F4;
:< 1F4 5 21 5A 3 1 CA
// Response: “OK”
OK
```

5 Fehlermeldungen

Im Fehlerfall gibt der Adapter einen Fehlercode zurück. Dabei kann es sich um interne Fehler handeln (Syntax, falsche Parametereingabe) oder Fehler bei der SDO-Übertragung.

Das Format der Nachricht hängt vom Fehlertyp ab:

Synchronous Error

Fehler, der nach dem Senden eines Kommandos auftritt

Format: `ERROR: XXX`, wo XXX der Fehlercode

Asynchronous Error

Fehler, der jederzeit auftreten kann

Format: `ERROR XXX`, wo XXX der Fehlercode. Ausschließlich die Fehlercodes 300 und 301 werden in diesem Format registriert

Beispiel

```
// Request: Send wrong command!!!
wrong com mand
// Response: "ERROR" with error code = 101
ERROR: 101
```

Interne Fehler

Fehlercode (dec)	Fehlermeldung
100	equest not supported
101	Syntax error
102	Request not processed due to internal state
103	Timeout (where applicable)
300	Error passive
301	Bus off
303	CAN buffer overflow
304	CAN init
305	CAN active (at init or start-up)

SDO-Fehlermeldungen

HINWEIS

Abhängig von Ihrem CAN-Controller können weitere SDO-Fehlermeldungen auftreten.

Fehlercode (hex)	Beschreibung
0500 0000	General SDO protocol error detected
0503 0000	Toggle bit not alternated
0504 0000	SDO protocol timed out

Fehlercode (hex)	Beschreibung
0504 0001	Client/server command specifier not valid or unknown
0504 0002	Invalid block size (block mode only)
0504 0003	Invalid sequence number (block mode only)
0504 0004	CRC error (block mode only)
0504 0005	Out of memory.
0601 0000	Unsupported access to an object
0601 0001	Attempt to read a write only object
0601 0002	Attempt to write a read only object
0602 0000	Object does not exist in the object dictionary
0604 0041	Object cannot be mapped to the PDO
0604 0042	The number and length of the objects to be mapped would exceed PDO length
0604 0043	General parameter incompatibility reason
0604 0047	General internal incompatibility in the device