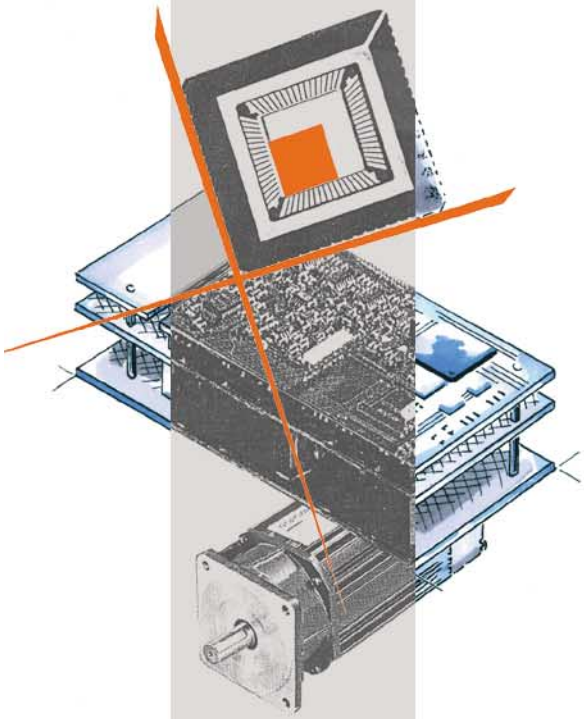
Busfunktionen ND31 und ND32

Version 4/2004



NOVOTRON

für Dynamik und Bewegung

In diesen Unterlagen gelten folgende Vereinbarungen:

Gefahr, Warnung vor lebensgefährlichen Betriebsspannungen
Mit diesem allgemeinen Gefahrensymbol sind Textstellen gekennzeichnet, die Sie unbedingt lesen und beachten müssen.

Nichtbeachtung kann zur Gefährdung von Leben und Gesundheit von Personen führen!

VORSICHT ! ☛ **Vorsicht, Warnung vor Zerstörung und Fehlfunktionen**
Mit diesem Vorsichtssymbol sind Textstellen gekennzeichnet, die Sie unbedingt lesen und beachten müssen.

Nichtbeachtung kann zu Zerstörung oder Fehlfunktionen des NOVODRIVE ND31 oder der Anlage führen!

☞ **Hinweis oder Empfehlung**
Hinweis auf andere Textstellen oder Empfehlungen für die Praxis.

1 2 **Reihenfolge einer Anweisung**

Menü *Grenzwerte* **Bezeichnung eines Menüs oder Untermenüs**

Befehl *Channel1* **Bezeichnung eines Befehls oder einer Funktion**

[ ], [enter] **Bezeichnung einer Taste oder Tastenfolge**

Graphische Darstellung

Graphische Darstellung von Bytes

SwVersion	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0xFF3D	7	6	5	4	3	2	1	0

SwVersion	Bezeichnung
0xFF3D	Adresse
R/W	Read/Write
R	ReadOnly

NOVOTRON

1 Allgemeines

1.1 Angaben zu diesem Handbuch

Die vollständige Dokumentation Ihres NOVODRIVE gliedert sich in 7 Teile:

1 Handbuch **Grundgerät** ND31 und ND32
Grundausstattung

2 Handbuch **Busfunktionen** ND31 und ND32 *)
Bei Bedarf

3 Handbuch **Grundfunktionen**
Bei Bedarf

4 Handbuch **Zusatzfunktionen**
Bei Bedarf

5 Reserviert

6 Handbuch **Inbetriebnahme**
Grundausstattung

7 **Anleitung für Einbau/Austausch von ND31 bzw. ND32**
Grundausstattung (Faltblatt)

Die in den Handbüchern verwendeten Symbole sind auf der Innenseite des Deckblatts aufgeführt. Diese Symbole sollen Ihnen das schnelle Auffinden wichtiger Informationen erleichtern.

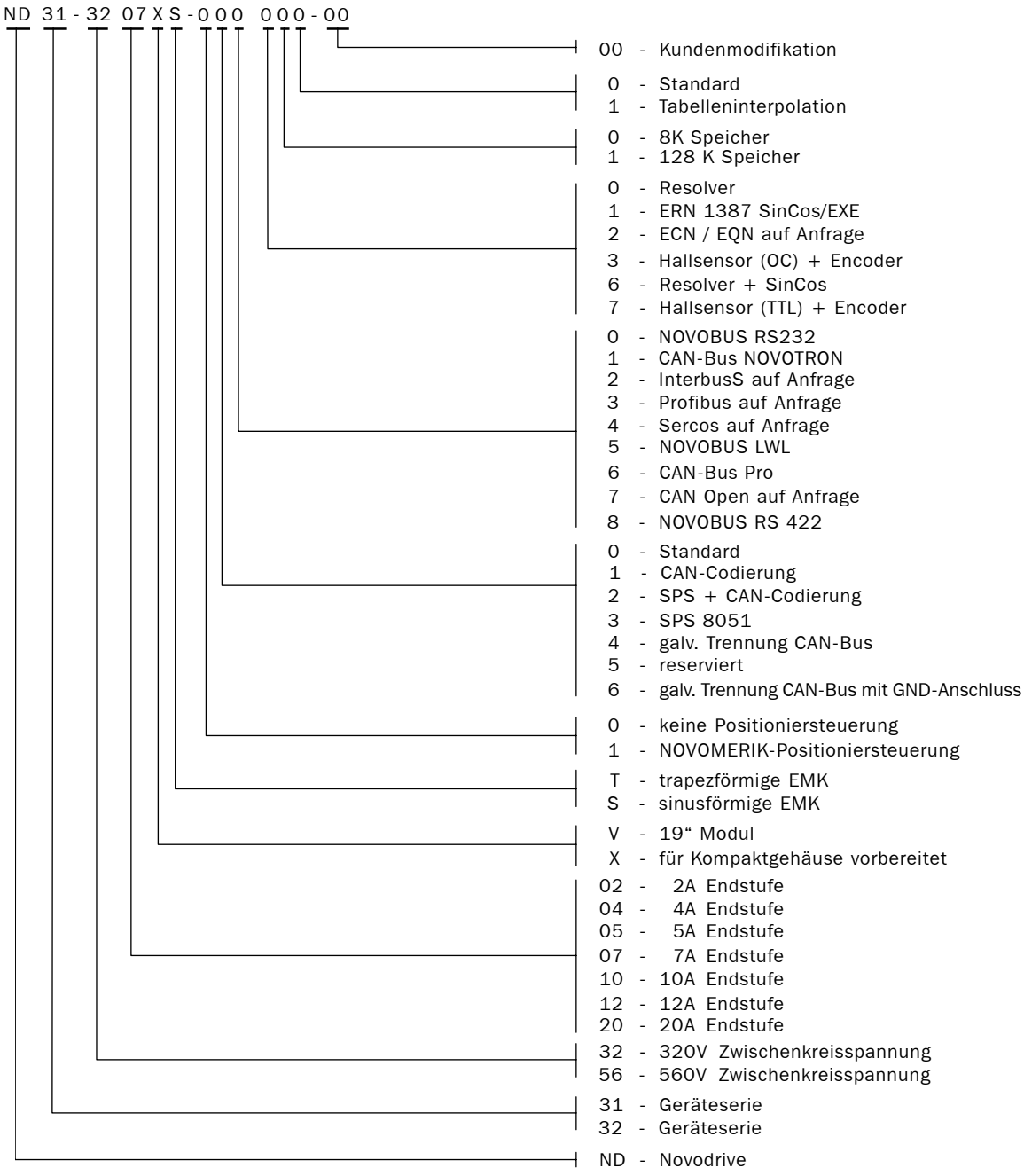
*) Vorliegendes Handbuch

1.2 Service / Kundendienst

NOVOTRON GmbH
Mauserstraße 31
D - 71640 Ludwigsburg

Telefon: +49-71 41-29 69-0
Fax +49-71 41-29 69-22

1.3 Bezeichnung



1.4 Schutzrechte

IBM ist eingetragenes Warenzeichen der IBM-Corporation

NOVOTRON

2 Allgemeine Sicherheitshinweise

Im NOVODRIVE gibt es lebensgefährliche Betriebsspannungen!

Verdrahtung Deshalb ist vor dem Einschalten des NOVODRIVE die Verdrahtung zu kontrollieren. Überprüfen Sie, ob alle Stecker richtig gesteckt sind, und ob die Erdung richtig ausgeführt wurde.

Absicherung Stellen Sie sicher, dass keine spannungsführenden Teile versehentlich berührt werden können und die Absicherung des NOVODRIVE vorhanden und richtig angeschlossen ist.

Not-Aus Sehen Sie eine „Not-Aus“-Schaltung vor mit der Motor jederzeit stillgesetzt werden kann.

Entladezeit Nach dem Ausschalten beträgt die Entladezeit der Elkos ca. 5 Minuten. Das bedeutet: Nach dem Ausschalten steht noch fünf Minuten lang eine gefährliche Berührspannung am Gerät an. Solange darf nichts berührt werden und kein Stecker gezogen werden.

Berührspannung Falls sich beim Ausschalten der Versorgungsspannung der Motor noch dreht, kann dieser die gefährliche Berührspannung noch bis zu seinem Stillstand aufrecht erhalten. Erst dann beginnt die Entladezeit der Elkos.

Ein- und Ausschalten Häufiges Ein- und Ausschalten der Versorgungsspannung in schneller Folge ist zu vermeiden, da dadurch die Einschalt-Strombegrenzung des NOVODRIVE überlastet werden kann. Diese Überlastung kann zur Zerstörung des Einschaltstrom-Begrenzungswiderstands führen. Es ist eine Wartezeit von 1 Minute zwischen Aus- und Einschalten einzuhalten.

Einschalt-Reihenfolge Beim Einschalten muss zuerst die 24VDC Versorgungsspannung für den Regelungsteil angelegt werden, bevor die Leistung zugeschaltet wird. Beim Abschalten ist umgekehrt zu verfahren.

☞ Lesen und beachten Sie:
Kapitel 2 Allgemeine Sicherheitshinweise im Handbuch
„Grundgerät ND31 und ND32“

3 NOVOBUS

In diesem Kapitel erfahren Sie, wie Sie den NOVOBUS zur Steuerung Ihres Antriebs nutzen können.

3.1 Allgemeines

Das NOVOBUS-Protokoll und der NOVOBUS-Treiber für IBM-kompatible PCs sind geistiges Eigentum des Herstellers.

NOVOBUS ist eine kostengünstige Lösung zur Vernetzung digitaler Antriebe und ermöglicht die schnelle Kommunikation zwischen einem Leitrechner (z.B. PC oder SPS) und den Antrieben:

- Soll-Istwert Austausch (z.B. Drehzahlsollwert- und -istwert)
- Übertragung neue Position-Sollwerte für Positionierachsen
- Parametrierung des Antriebreglers (z.B. die Einstellung und Online Änderung von Reglerstrukturen, Reglerparametern und den erlaubten Maximalwerten, usw.)
- Übertragung von Steuerbefehlen (Start, Stop, Reglersperre, etc.)
- Abfrage wichtiger Informationen (z.B. Kühlkörper- und Motortemperatur, Endschalter-, Betriebsbereit-, In-Position-Signal, zusätzliche externe Signale als Prozessinformation, integrierter Betriebsstundenzähler, Status des Antriebs, eventuelle Fehlermeldungen)
- Steuerung der programmierbaren analogen und digitalen Ausgänge der Antriebe (z.B. für Betätigung von Schützen oder Bremsen, Ausgeben von Warnsignalen usw.)

RS232/RS422 Das physikalische Übertragungsmedium für NOVOBUS ist die genormte serielle Schnittstelle RS232 oder RS422, dem Standard bei allen PCs und modernen Steuerungen. Für die Kommunikation per NOVOBUS sind keine Hardware-Erweiterungen (wie z.B. Buscontroller, Kommunikationskarte, Protokollchip, intelligente Busklemmen) nötig.

Treiber Alle digitalen Antriebe enthalten standardmäßig die notwendige serielle Schnittstelle, sowie den Software-Treiber für das NOVOBUS-Protokoll. Für PC-Programme steht ein NOVOBUS-Treiber als Software-Bibliothek (NB31.LIB) kostenlos zur Verfügung.

Ringstruktur Die einzige Voraussetzung für die Kommunikation mit NOVOBUS ist die ringförmige Verkabelung des Rechners und der Antriebe mit einem RS232- bzw. RS422-Kabel oder mit Lichtwellenleitern (Option).

In einem Ring kann man bis zu 250 Achsen steuern. Die Antriebe sind durch ihre Positionen im Ring automatisch adressiert. Für höhere Übertragungsgeschwindigkeit oder um autarke Maschinenteile zusammenzufassen, können die Antriebe auf mehrere Ringe verteilt werden.

3.2 Leistungsmerkmale

Die Übertragung eines Byte dauert 286,46 μs bei 38400 Bit/s Übertragungsgeschwindigkeit.

Geschwindigkeit

Die erforderliche Zeit zum kompletten Soll-Istwert Austausch bei drehzahlgeregelten Antrieben beträgt:

Antriebe pro Ring	**Zeit**
1	0,86 ms
2	2,0 ms
3	2,9 ms
4	3,7 ms
5	4,6 ms
6	5,4 ms

Die erforderliche Zeit zur Übertragung neuer Position-Sollwerte für Positionierachsen mit Starten der PS-Rechnung beträgt:

Antriebe pro Ring	**Zeit**
1	4,01 ms
10	40,39 ms
100	401,33 ms
250	1002,79 ms

3.3 Physikalisches Übertragungsmedium

Wahlweise RS232 oder RS422.

Übertragungsgeschwindigkeit: Standard 38400 Bit/s

Die Übertragung erfolgt mit 8 Datenbits, 1 Paritybit (ungerade Parity) und 1 Stopbit.

3.4 Busstruktur

NOVOBUS hat eine ringförmige Struktur, die Antriebe können auf einen oder mehrere Ringe verteilt sein.

Master/Slave

Pro Ring: 1 Bus Master (Leitrechner)
max. 250 Slaves (Antriebe)

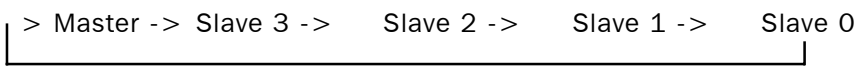
Im Normalbetrieb dürfen die Slaves nur auf die Master-Telegramme antworten. Beim Time-Out Fehler dürfen auch die Slaves selbständig Fehler-Telegramme schicken.

3.5 Adressierung der Teilnehmer

250 Achsen im Ring

In einem Ring können bis zu 250 Achsen gesteuert werden. Die Antriebe sind durch ihre Positionen im Ring automatisch adressiert. Die Numerierung der Teilnehmer beginnt mit dem letzten im Ring und der Adresse ‘0’. Die Adresse des ersten Teilnehmers im Ring ist N-1, wobei N die Zahl der Slaves bedeutet.

z.B. bei 4 Achsen:



NOVOTRON

4 NOVOBUS Protokoll

Der Master (Leitrechner) schickt kontinuierlich Telegramme. Die meisten Telegramme enthalten eine Adresse (Ausnahme: 'SYNC0' und 'PAUSE'). Die nicht angesprochenen Teilnehmer schicken die Telegramme weiter. Der angesprochene Teilnehmer antwortet auf das Telegramm. Die Länge der Antwort ist gleich der des Master-Telegramms.

4.1 Telegramm Aufbau

Synchronbyte (obligatorisch)

Adressbyte (optional)

Prozess-Datenkanal (2 Byte, optional)

Parameterkanal (1-7 Byte, optional)

Alle Bytes werden mit ungerader Parity gesendet. Die Netto-Telegrammlänge (Prozess-Datenkanal + Parameterkanal) kann maximal 7 Byte betragen. Bei einem Telegramm mit Prozess-Datenkanal ist also der Parameterkanal max. 5 Byte lang.

4.2 Synchronbyte

Das Synchronbyte ist immer das erste Byte eines Telegramms. Darin ist kodiert die Telegrammlänge enthalten.

Bit	7	6	5	4	3	2	1	0
Bedeutung	1	N	S	0	T2	T1	T0	D

N = 1 Next bei Short Address (sonst = 0)

S = 1 Short Address

T2-T0 Netto Telegrammlänge
(ohne Synchronbyte und Adressbyte, 0...7 Byte)

D = 1 Das Telegramm enthält Prozessdaten-Informationen
(2 Byte Datenkanal)

Kurzaddressierung

Bei S=1 und N=0 wird der gleiche Antrieb angesprochen wie im vorhergehenden Telegramm.

Bei S=1 und N=1 wird der nächste Antrieb angesprochen (Address = Address0+1).

Bei S=0 und N=0 folgt ein Adressbyte.

Falls kein Prozessdatenkanal oder Parameterkanal aktiv ist, wird das Synchronbyte 'SYNC0' gesendet, um den kontinuierlichen Datenstrom aufrechtzuhalten. 'SYNC0' wird von dem Empfänger unverändert weitergeschickt.

'SYNC0': 0x80

Um den Datenstrom aufrecht zu erhalten, kann ein Synchronbyte 'PAUSE' gesendet werden. 'PAUSE' als Synchronbyte wird vom Empfänger ignoriert (keine Antwort).

'PAUSE': 0x81

4.3 Addressbyte

Bit	7	6	5	4	3	2	1	0
Bedeutung	A7	A6	A5	A4	A3	A2	A1	A0

Beim Master-Telegramm enthält A7-A0:

k-N

k = Teilnehmer-Adresse im Ring

N = Zahl der passiven Teilnehmer (Slaves).

Jeder Teilnehmer erhöht die Adresse um eins und schickt das Telegramm mit der neuen Adresse weiter. Ein Antrieb ist adressiert, wenn die Adresse nach der Inkrementierung gleich null ist. Die nicht adressierten Antriebe schicken das ganze Telegramm bis auf das Adressbyte unverändert weiter, ohne den Inhalt zu prüfen.

Beispiel N=5 , der Leitrechner möchte den Teilnehmer Achse2 ansprechen: (k-N = 2-5 = -3 = 0xFD).

Bit	7	6	5	4	3	2	1	0	
Bedeutung	1	N	S	0	T2	T1	T0	D	
Master:	1	1	1	1	1	1	0	1	(0x FD=k-N)
Achse4:	1	1	1	1	1	1	1	0	(0x FE)
Achse3:	1	1	1	1	1	1	1	1	(0x FF)
Achse2:	0	0	0	0	0	0	0	0	(0x 00) <- !!!
Achse1:	0	0	0	0	0	0	0	1	(0x 01)
Achse0:	0	0	0	0	0	0	1	0	(0x 02 = k)

Die inkrementierte Adresse ist bei der adressierten Achse (Achse2) gleich Null. Der Master bekommt die Antwort auf sein Telegramm mit der Adresse ‘k’ zurück (In diesem Beispiel, in dem Achse2 angesprochen wurde: 0x02).

4.4 Prozessdatenkanal

Mit diesem Kanal kann schneller der Soll- und Istwert-Austausch realisiert werden. Dazu muss das niedrigste Bit im Synchronbyte gesetzt werden.

☞ siehe Abschnitt 4.2

Die Daten sind immer 2 Byte lang. Im Telegramm wird zuerst das Byte mit dem höheren Wert (MSB) und dann das mit dem niedrigen Wert (LSB) gesendet.

4.5 Parameterkanal

Mit Hilfe des Parameterkanals kann man die Antriebe mit Parametern versehen und Informationen von den Antrieben holen. Ein Befehl im Parameterkanal besteht aus einem Commandbyte, Datenbytes (0...4) und einem Kontrollbyte (Checksumme).

Mit einem NOVOBUS-Telegramm können mehrere Befehle durch den Parameterkanal geschickt werden, bzw. ein Befehl im Parameterkanal kann in mehrere Telegramme verteilt werden.

Die Antwort auf einen Befehl ist genauso lang wie der Befehl selbst (2 bis 6 Byte).

☞ Das Commandbyte und das Kontrollbyte dürfen nie den Wert 0x00 haben.

4.6 Befehle für ND31 und ND32 im Parameterkanal

Verwendete Abkürzungen

A0	Low Byte der Adresse
A1	High Byte der Adresse
D0	LSB der Daten (8, 16, und 32 Bit Werte)
D1	...
D2	...
D3	MSB der Daten (32 Bit Werte)
CS	Die Prüfsumme, die durch byteweises Addieren aller Nutzdaten im Telegramm berechnet wird
NCS	Das Zweierkomplement der Prüfsumme aus den Nutzdaten
-	wird ignoriert

☞ Die Adresse bezieht sich bei 16 und 32 Bit Daten auf das MSB

4.6.1 Speicherzugriffe auf den internen Speicher

Befehl **Read Byte**

Funktion Auslesen eines 8 Bit Wertes aus dem internen Speicher

Byte	0	1	2	3	4	5	6	7
Befehlstelegramm	0xC0	A0	A1	CS	-	-	-	-
Antworttelegramm	0xC0	A0	D0	NCS	-	-	-	-
Adressbereich	0x0000 - 0xFFFF							

Befehl **Read Word**

Funktion Auslesen eines 16 Bit Wertes aus dem internen Speicher

Byte	0	1	2	3	4	5	6	7
Befehlstelegramm	0xC1	A0	A1	0x3F	CS	-	-	-
Antworttelegramm	0xC1	A0	D0	D1	NCS	-	-	-
Adressbereich	0x0000 - 0xFFFF							

Befehl **Read Long**

Funktion Auslesen eines 32 Bit Wertes aus dem internen Speicher

Byte	0	1	2	3	4	5	6	7
Befehlstelegramm	0xC7	A0	A1	0x31	0x32	0x3F	CS	-
Antworttelegramm	0xC7	A0	D0	D1	D2	D3	NCS	-
Adressbereich	0x0000 - 0xFFFF							

Befehl **Write Byte**

Funktion Schreiben eines 8 Bit Wertes in den internen Speicher

Byte	0	1	2	3	4	5	6	7
Befehlstelegramm	0x82	D0	A0	A1	CS	-	-	-
Antworttelegramm	0x82	D0	A0	A1	NCS	-	-	-
Adressbereich	0xFE00 - 0xFE7F, 0xFEA0 - 0xFF7F							

Befehl **Write Word**

Funktion Schreiben eines 16 Bit Wertes in den internen Speicher

Byte	0	1	2	3	4	5	6	7
Befehlstelegramm	0x63	D0	D1	A0	A1	CS	-	-
Antworttelegramm	0x63	D0	D1	A0	A1	NCS	-	-
Adressbereich	0xFE00 - 0xFE7F, 0xFEA0 - 0xFF7F							

Befehl **Write Long**

Funktion Schreiben eines 32 Bit Wertes in den internen Speicher

Byte	0	1	2	3	4	5	6	7
Befehlstelegramm	0xC8	D0	D1	D2	D3	A0	A1	CS
Antworttelegramm	0xC8	D0	D1	D2	D3	A0	A1	NCS
Adressbereich	0xFE00 - 0xFE7F, 0xFEA0 - 0xFF7F							

☞ Da Write Long Telegramme 8 Byte lang sind, müssen sie auf zwei NOVOBUS Telegramme aufgeteilt werden.

4.6.2 Bitmanipulation

Befehl **AND**

Funktion Bitweises logisches UND mit einem Register im internen Speicher. Die oberen 8 Bit der Adresse sind immer 0xFF

Byte	0	1	2	3	4	5	6	7
Befehlstelegramm	0xA4	D0	A0	CS	-	-	-	-
Antworttelegramm	0xA4	D0	A0	NCS	-	-	-	-
Adressbereich	0xFF00 - 0xFFFF							

Befehl **OR**

Funktion Bitweises logisches ODER mit einem Register im internen Speicher. Die oberen 8 Bit der Adresse sind immer 0xFF

Byte	0	1	2	3	4	5	6	7
Befehlstelegramm	0xA5	D0	A0	CS	-	-	-	-
Antworttelegramm	0xA5	D0	A0	NCS	-	-	-	-
Adressbereich	0xFF00 - 0xFFFF							

4.6.3 Speicherzugriffe auf den externen Speicher

Befehl **Read Word X**

Funktion Auslesen eines 16 Bit Wertes aus dem externen Speicher

Byte	0	1	2	3	4	5	6	7
Befehlstelegramm	0xC9	A0	A1	0x3F	CS	-	-	-
Antworttelegramm	0xC9	A0	D0	D1	NCS	-	-	-
Adressbereich	0x0000 - 0xFFFF							

Befehl **Write Word X**

Funktion Schreiben eines 16 Bit Wertes in den externen Speicher

Byte	0	1	2	3	4	5	6	7
Befehlstelegramm	0x6A	D0	D1	A0	A1	CS	-	-
Antworttelegramm	0x6A	D0	D1	A0	A1	NCS	-	-
Adressbereich	0x4000 - 0xFFFF							

4.6.4 Steuerbefehle

Befehl **Reset**

Funktion Löst einen Reset des NOVODRIVE aus

Byte	0	1	2	3	4	5	6	7
Befehlstelegramm	0xDD	0x21	CS	-	-	-	-	-
Antworttelegramm	-	-	-	-	-	-	-	-

4.7 Prüfsumme im Parameterkanal

Der Master (Leitrechner) sendet alle Befehle im Parameterkanal mit einer Prüfsumme (CS). Diese Prüfsumme wird vom angesprochenen Antrieb kontrolliert. Der Befehl wird nur bei richtiger Prüfsumme ausgeführt, sonst meldet der Antrieb Fehler.

☞ siehe Abschnitt 4.8 Fehlerbehandlung

Prüfsumme Die Prüfsumme ist die Summe der Bytes eines Befehls. Falls die Summe 0 ergibt, muss die Prüfsumme durch den Wert 1 ersetzt werden (0x00 ist reserviert für Fehlermeldungen). Der angesprochene Antrieb bildet aus seiner Antwort eine neue Prüfsumme und sendet deren Zweierkomplement (NCS) weiter.

Die Prüfsumme betrifft nur den Parameterkanal und wird nur aus den Bytes des Parameterkanals gebildet. Synchronbyte, Adressbyte und Prozessdaten werden nicht in der Prüfsumme berücksichtigt.

4.8 Fehlerbehandlung

Wenn der Antrieb einen Fehler in der Kommunikation bemerkt (Parityfehler, Framingfehler, falsches Synchronbyte, falscher Befehl im Parameterkanal, falscher Befehls-Parameter oder falsche Prüfsumme), geht er in Fehlerzustand.

Nach einem Fehler antwortet der Antrieb, der ihn zuerst erkennt, mit 0x00 auf alle empfangenen Bytes. Die folgende Achse kann den Fehler schnell erkennen, wenn sie adressiert wurde: falsches Synchron-, Command- oder Kontrollbyte (dürfen nie 0x00 sein), eventuell falsche Befehls-Parameter (falls an der Stelle keine 0x00 akzeptabel ist).

Ein nicht adressierter Antrieb prüft den Inhalt des Telegrammes nicht. Er kann den Fehler erst am Ausbleiben des nächsten Synchronbytes erkennen. Ein Telegramm kann maximal 9 Byte lang sein. Falls der Fehler im Adressbyte vorkommt, bemerken ihn die anderen Achsen, die seit diesem Fehler lauter 0x00 empfangen, erst mit dem 9. Byte, nach dem letzten Synchronbyte

Ein fehlerfreies Telegramm darf 0x00 nur in dem Adressbyte (1 Byte), im Datenprozesskanal (2 Byte) und im Datenbereich des Parameterkanals (max. 4 Byte) enthalten. Nacheinander darf also höchstens 7 mal 0x00 erscheinen. Ein Antrieb im Fehlerzustand zählt deshalb, wieviele Nullbytes der vorherige Antrieb sendet. Falls er ohne Unterbrechung 8 mal 0x00 empfängt, bedeutet es für ihn, dass sich auch der vorherige Antrieb im Fehlerzustand befindet.

In diesem Fall prüft er die empfangenen Bytes, ob eine Checksequenz gesendet wird. Wenn er eine Checksequenz erkennt, geht er wieder in den Betriebszustand.

☞ siehe Abschnitt 4.9. Checksequenz

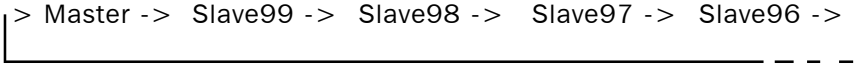
Damit auch der nächste Antrieb den Fehler mit Sicherheit erkennt, muss der mittlere mindestens 9mal 0x00 senden. Damit dieser nachfolgende Antrieb erkennt, dass der vorherige auch im Fehlerzustand ist, muss er weitere 8 Nullbytes empfangen.

Deshalb sendet jeder Antrieb im Fehlerzustand zuerst 17mal 0x00. Falls er erkannt hat, dass der vorherige Antrieb im Ring im Fehlerzustand ist, sendet er alle empfangenen Bytes inkrementiert weiter.

Der Master empfängt bei einem Fehler zuerst maximal 25mal 0x00, dann die Adresse des Antriebes der den Fehler bemerkt hat. Damit ist die einfache Lokalisierung des Fehlers gewährleistet.

Beispiel: 100 Achsen

Der Master möchte je ein Byte aus dem RAM der Antriebe No.95 bis No.99 lesen. Zwischen den Antrieben No.98 und No.97 wird ein Bit fehlerhaft übertragen. Der Antrieb No.97 bemerkt es als Parityfehler.



Das 1. Telegramm

1. Byte	0x88 =	Synchronbyte (es folgen Adressbyte und 4 Bytes im Parameterkanal, keine Prozessdaten)
2. Byte	0xFB =	-5 = 95-100 (Slave 95 ist angesprochen bei 100 Slaves im Ring)
3. Byte	0xC0 =	Readbyte-Befehl für ND31/ND32
4. Byte	0x13 =	AddressL, LSB der Adresse 0xFE13
5. Byte	0xFE =	AddressM, MSB der Adresse 0xFE13
6. Byte	0xD1 =	Kontrollbyte (C0+13+FE = 1D1)

NOVOTRON

Die folgenden Telegramme unterscheiden sich nur im 2. Byte (96-100 = -4 = 0xFC, 97-100 = -3 = 0xFD, ...):

Master	Slave 98	Slave 97	Fehler	Slave 97	Slave 96	Slave 95	...	Slave 00
88	88	88		88	88	88	...	88
FB	FC	FD		FE	FF	00		5F
C0	C0	C0		C0	C0	C0		C0
13	13	13		13	13	13		13
FE	FE	FE		FE	FE	>88		88
D1	D1	D1		D1	D1	+A5		A5
88	88	88		88	88	88		88
FC	FD	FE		FF	00	01		60
C0	C0	C0		C0	C0	C0		C0
13	13	13	12	!00	00	00		00
FE	FE	FE		00	!00	00		00
D1	D1	D1		00	00	00		00
88	88	88		00	00	!00		!00
FD	FE	FF		00	00	00		00
C0	C0	C0		00	00	00		00
13	13	13		00	00	00		00
FE	FE	FE		00	00	00		00
D1	D1	D1		00	*00	00		00
88	88	88		00	00	00		00
FE	FF	00		00	00	*00		*00
C0	C0	C0		00	00	00		00
13	13	13		00	00	00		00
FE	FE	88		00	00	00		00
D1	D1	A5		00	00	00		00
88	88	88		00	00	00		00
FF	00	01		#00	00	00		00
C0	C0	C0		00	#00	00		00
13	13	13		00	01	00		00
FE	88	88		00	01	#00		#00
D1	A5	A5		00	01	02	...	61
88	88	88		00	01	02		61
9C	9D	9e		00	01	02		61
.	.	.	.	.	.	.	.	

>: ausgelesener Registerwert im Antwort-Telegramm

+: Neue Prüfsumme

!: Der Antrieb erkennt einen Fehler

*: Der Antrieb erkennt, dass der vorherige Antrieb im Fehlerzustand ist

#: Das 17. gesendete Nullbyte nach der Fehlererkennung

☞ Die Meldungen 0x00 zeigen dem Master den Übertragungsfehler, die Bytes 0x61= 97 bedeuten, dass der Fehler vom Slave 97 bemerkt wurde.

4.9 Checksequenz

Falls der Master einen Fehler im Datenverkehr bemerkt, schickt er eine Checksequenz, um die Antriebe zurückzusetzen. Die ersten 17 Bytes vor der eigentlichen Checksequenz sind 0x00, damit auch die Antriebe in den Fehlerzustand gehen, die den Fehler noch nicht bemerkt haben. Nach der Checksequenz kann der Master die nicht ausgeführten Befehle wiederholen.

Die Checksequenz lautet:

(17mal 0x00) 0xFF 0x44 0x72 0x4C 0x41

Beispiel:

100 Achsen, die Achsen No.97 - No.0 sind im Fehlerzustand, die Achsen No.99 - No.98 haben keinen Fehler in der Kommunikation feststellen können. Der Master sendet gerade ein Telegramm, als er den Fehler durch ein unerwartetes Nullbyte erkennt. Nach der Erkennung des Fehlers sendet er 17mal 0x00, dann setzt er den Bus mit einer Checksequenz zurück.

```
|> Master ->  Slave99 ->  Slave98 ->  Slave97 ->  Slave96 ->
```

	Master	Slave99	Slave98	Slave97	Slave96	Slave95	...	Slave00
Angefangenes Telegramm	88	88	88	00	01	02	...	61
17 Nullbytes	00	01	02	00	01	02	...	61
	00	00	00	00	01	02		61
	00	00	00	00	01	02		61
	00	00	00	00	01	02		61
	00	!00	!00	00	01	02		61
	00	00	00	00	01	02		61
	00	00	00	00	01	02		61
	00	00	00	00	01	02		61
	00	00	00	*01	02	03		62
	00	00	00	01	02	03		62
	00	00	00	01	02	03		62
	00	*00	*00	01	02	03		62
	00	00	00	01	02	03		62
	00	00	00	01	02	03		62
	00	00	00	01	02	03		62
	00	00	00	01	02	03		62
	00	00	00	01	02	03		62
	00	00	00	01	02	03		62

Checkcode	FF	@FF	@FF	@FF	@FF	@FF	...	@FF
	44	44	44	44	44	44		44
	72	72	72	72	72	72		72
	4C	4C	4C	4C	4C	4C		4C
	41	41	41	41	41	41		41
Wiederholung des Telegramms	88	88	88	88	88	88	...	88
	FC	FD	FE	FF	00	01		60
	C0	C0	C0	C0	C0	C0		C0
	13	13	13	13	13	13		13
	FE	FE	FE	FE	>88	88		88
	D1	D1	D1	D1	+A5	A5		A5

Anmerkungen:

!: Der Antrieb erkennt einen Fehler

*: Der Antrieb erkennt, dass der vorherige Antrieb im Fehlerzustand ist

@: Der Antrieb erkennt das erste Byte der Checksequenz

>: Antwortbyte in einem Telegramm (Inhalt der Speicherzelle)

+: Neue Prüfsumme

☞ Nach der Checksequenz ist der Bus wieder funktionsfähig.

NOVOTRON

5 CAN Bus

5.1 Allgemeines

Der NOVODRIVE bietet als Option ein CAN Interface CAN 2.0 A und B nach ISO/DIS 11898. Von der Software werden nur Identifier mit 11 Bit unterstützt.

Als weitere Option ist auch eine galvanische Trennung der Signalleitungen erhältlich.

5.2 Baudrate

Dieser Abschnitt ist nur für spezielle Anwendungen gedacht. Standardmäßig ist eine Baudrate von 1MBit/s eingestellt.

Sie können dazu die Einstellung in den folgenden Registern kontrollieren.

Register	Adresse	Wert
CANInitBTR0	0xFEE8	0x40
CANInitBTR1	0xFEE9	0x25

Konfiguration

Der NOVODRIVE ist mit einem Intel CAN Controller 82527 ausgerüstet. Die Bitrate wird durch zwei 2 Parameter im Parametersatz festgelegt, mit denen zwei Register im CAN Controller des ND31 initialisiert werden. Über diese beiden Parameter kann die Bitrate verstellt werden. Sie werden nach einem Reset aktiv:

CANinitBTR0	7	6	5	4	3	2	1	0
0xFEE8	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Funktion	SJW		BRP					
Standardwert für 1 MBit	0	1	0	0	0	0	0	0

Erläuterung **BRP** Baud Rate Prescaler

CANinitBTR1	7	6	5	4	3	2	1	0
0xFEE9	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Funktion	SPL	TSEG2			TSEG1			
Standardwert für 1 MBit	0	0	1	0	0	1	0	1

Erläuterung

TSEG1 Time Segment 1

TSEG2 Time Segment 2

Baudrate = 10MHz / ((BRP + 1) * (3 + TSEG1 + TSEG2))

Beispiel 1 Für 1 MBit gilt:

Baudrate = 10 MHz / ((0 + 1) * (3 + 5 + 2)) = 1MHz

Ein einzelnes Bit auf dem CAN-Bus setzt sich aus 3 Segmenten zusammen:

$t_{Sync}+t_{TSEG1}+t_{TSEG2}$	Länge eines Bits
$t_{Sync}+t_{TSEG1}$	bilden die Zeit vor dem Samplingpoint
t_{TSEG2}	ist die Zeit nach dem Samplingpoint

Die Länge der drei Segmente ist programmierbar. Die drei Segmente setzen sich aus einer Anzahl von sogenannten Timequanta t_q zusammen. Die Länge eines t_q wird mit dem Parameter **CANInitBTR0** festgelegt. Für BRP sind die Werte 0 bis 63 zulässig. Für ein t_q gilt:

$t_q = (BRP+1)*100ns$

t_{Sync} entspricht immer 1 t_q.

Die Bits 7 und 6 in **CANInitBTR0** sind 0 und 1 und sollten nicht verändert werden.

Mit **CANInitBTR1** wird die Länge von t_{TSEG1} und t_{TSEG2} festgelegt. Das Bit SPL bestimmt den Samplingmode:

1 = jedes Bit wird dreifach abgetastet

0 = jedes Bit wird einmal abgetastet.

Bei Bitraten oberhalb 100kBit darf nur einmal abgetastet werden.

Die Länge von t_{TSEG2} beträgt:

$t_{TSEG2} = (TSEG2+1)*t_q$ gültige Werte für TSEG2 sind 1 – 7

Die Länge von t_{TSEG1} beträgt:

$t_{TSEG1} = (TSEG1+1)*t_q$ gültige Werte für TSEG1 sind 2 – 15

Beispiel 2 **1 MBit/s:**

CANInitBTR0 = 0x40
CANInitBTR1 = 0x16

BPR = 0 -> t_q = (0+1)*100ns = 100ns
t_{Sync} = 1 tq = 100ns

TSEG2 = 1
t_{TSEG2} = (1+1)*100ns = 200ns

TSEG1 = 6
t_{TSEG1} =(6+1)*100ns = 700ns
$t_{Sync}+t_{TSEG1}+t_{TSEG2}$ = 100ns+200ns+700ns =1μs

Das Bit wird nach 700ns abgetastet.

Beispiel 3 **500 kBit/s**

CANInitBTR0 = 0x40
CANInitBTR1 = 0x2F

BPR = 0 -> t_q = (0+1)*100ns = 100ns
t_{Sync} = 1 tq = 100ns

TSEG2 = 2
t_{TSEG2} = (2+1)*100ns = 300ns

TSEG1 = 6
t_{TSEG1} =(15+1)*100ns = 1600ns
$t_{Sync}+t_{TSEG1}+t_{TSEG2}$ = 100ns+300ns+1600ns =2μs

Das Bit wird nach 1700ns abgetastet.

Hinweise

- t_q möglichst klein wählen
- Den Abtastpunkt möglichst weit nach hinten im Bit legen (ermöglicht größere Leitungslängen.

NOVOTRON

6 CAN Protokoll Novotron

6.1 Service Kanal

Für nicht zeitkritische Operationen steht ein Servicekanal mit verschiedenen Befehlen zur Verfügung.

6.1.1 Message Objekte auf dem NOVODRIVE

Message Objekt **CAN Service Receive (Empfang von Befehlen)**

Identifier Bit	10	9	8	7	6	5	4	3	2	1	0
Register	7	6	5	4	3	2	1	0	4	3	2
	CANService								**CANIDLSB**		
Adresse	0xFF38								0xFED2		

Message Objekt **CAN Service Transmit (Senden der Antwort)**

Identifier Bit	10	9	8	7	6	5	4	3	2	1	0
	(Identifier CAN Service Receive) + 1										

☞ Wenn **CANService** auf Null gesetzt wird, wird der CAN Bus beim Reset deaktiviert.

6.1.2 Bestimmung des CANService Identifiers über Digitaleingänge

Für die Generierung der Identifier für den Service Kanal können auch Digitaleingänge benutzt werden. Die Auswahl erfolgt mit dem Parameter **CANServiceSel**.

Register	**CANServiceSel**
Adresse	0xFEEC
Größe	16 Bit

Funktion

Ein ‘0’ Bit bedeutet: Der Wert des Digitaleingangs wird ignoriert.
Ein ‘1’ Bit bedeutet: Der Digitaleingang wird zur Generierung von **CANService** verwendet.

Die Bits in **CANServiceSel** bedeuten im Einzelnen:

Bit0	DipSwitch 1	(D1)	On = 0
Bit1	Dipswitch 2	(D2)	On = 0
Bit2	Dipswitch 3	(D3)	On = 0
Bit3	Dipswitch 4	(D4)	On = 0
Bit4	Dipswitch 5	(D5)	On = 0
Bit5	Dipswitch 6	(D6)	On = 0
Bit6	Dipswitch 7	(D7)	On = 0
Bit7	Dipswitch 8	(D8)	On = 0
Bit8	Tast8	(T8)	24V oder offen = 0
Bit9	Tast7	(T7)	24V oder offen = 0
Bit10	Tast6	(T6)	24V oder offen = 0
Bit11	Tast5	(T5)	24V oder offen = 0
Bit12	GPIn2	(G2)	0V oder offen = 1
Bit13	GPIn6	(G6)	0V oder offen = 1
Bit14	GPIn7	(G7)	0V oder offen = 1
Bit15	GPIn10	(G10)	0V oder offen = 1

Von den Bits 0 - 10 des CANService Identifiers werden soviele Bits, angefangen von Bit 0, verändert, wie Digitaleingänge angewählt sind. Die übrigen Bits bleiben unverändert.

Falls mehr als 11 Bits selektiert si nd werden nur die ersten 11 Bits berücksichtigt.

Die Funktion wird durch **CANServiceSel** = 0 abgeschaltet.

Die Eingänge Dipswitch 1 - 8 stehen nur als Option bei bei 19“-Geräten zur Verfügung. Alle anderen Eingänge können standardmäßig verwendet werden.

☞ Bei Dipswitch 7 = On ist der Encoderemmulator sowie die RS442 Schnittstelle abgeschaltet!

Beispiel 1

CANServiceSelect:	B’0000 0000 0000 1111
CANServiceIDEmpfang:	B’uuuu uuuu uuu u D4 D3 D2 D1
CANServiceIDSenden:	B’uuuu uuuu uuu u D4 D3 D2 D1 + 1

Beispiel 2

CANServiceSelect:	B’0100 1111 0000 0001
CANServiceIDEmpfang:	B’uuuu uuuu uu G7 T5 T6 T7 T8 D0
CANServiceIDSenden:	B’uuuu uuuu uu G7 T5 T6 T7 T8 D0+1

6.2 Befehle für ND31 und ND32 auf dem Service Kanal

Von den Telegrammen sind nur die Nutzdaten angegeben.

Verwendete Abkürzungen:

A0	Low Byte der Adresse
A1	High Byte der Adresse
D0	LSB der Daten (8, 16 und 32 Bit Werte)
D1	...
D2	...
D3	MSB der Daten (32 Bit Werte)
CS	Die Prüfsumme, die durch byteweises Addieren aller Nutzdaten berechnet wird.
NCS	Das Zweierkomplement der Prüfsumme aus den Nutzdaten
-	wird ignoriert

☞ Die Adresse bezieht sich bei 16 und 32 Bit Daten auf das MSB.

6.2.1. Speicherzugriffe auf den internen Speicher

Befehl **Read Byte**

Funktion Auslesen eines 8 Bit Wertes aus dem internen Speicher

Byte	0	1	2	3	4	5	6	7
Befehlstelegramm	0xC0	A0	A1	CS	-	-	-	-
Antworttelegramm	0xC0	A0	D0	NCS	-	-	-	-
Adressbereich	0x0000 - 0xFFFF							

Befehl **Read Word**

Funktion Auslesen eines 16 Bit Wertes aus dem internen Speicher

Byte	0	1	2	3	4	5	6	7
Befehlstelegramm	0xC1	A0	A1	0x3F	CS	-	-	-
Antworttelegramm	0xC1	A0	D0	D1	NCS	-	-	-
Adressbereich	0x0000 - 0xFFFF							

Befehl **Read Long**

Funktion Auslesen eines 32 Bit Wertes aus dem internen Speicher

Byte	0	1	2	3	4	5	6	7
Befehlstelegramm	0xC7	A0	A1	0x3F	0x3F	0x3F	CS	-
Antworttelegramm	0xC7	A0	D0	D1	D2	D3	NCS	-
Adressbereich	0x0000 - 0xFFFF							

Write Byte

Befehl Schreiben eines 8 Bit Wertes in den internen Speicher

Funktion

Byte	0	1	2	3	4	5	6	7
Befehlstelegramm	0x82	D0	A0	A1	CS	-	-	-
Antworttelegramm	0x82	D0	A0	A1	NCS	-	-	-
Adressbereich	0xFE00 - 0xFE7F, 0xFEA0 - 0xFF7F							

Write Word

Befehl Schreiben eines 16 Bit Wertes in den internen Speicher

Funktion

Byte	0	1	2	3	4	5	6	7
Befehlstelegramm	0x63	D0	D1	A0	A1	CS	-	-
Antworttelegramm	0x63	D0	D1	A0	A1	NCS	-	-
Adressbereich	0xFE00 - 0xFE7F, 0xFEA0 - 0xFF7F							

Befehl **Write Long**

Funktion Schreiben eines 32 Bit Wertes in den internen Speicher

Byte	0	1	2	3	4	5	6	7
Befehlstelegramm	0xC8	D0	D1	D2	D3	A0	A1	CS
Antworttelegramm	0xC8	D0	D1	D2	D3	A0	A1	NCS
Adressbereich	0xFE00 - 0xFE7F, 0xFEA0 - 0xFF7F							

6.2.2. Bitmanipulation

Befehl **AND**

Funktion Bitweises logisches UND mit einem Register im internen Speicher. Die oberen 8 Bit der Adresse sind immer 0xFF

Byte	0	1	2	3	4	5	6	7
Befehlstelegramm	0xA4	D0	A0	CS	-	-	-	-
Antworttelegramm	0xA4	D0	A0	NCS	-	-	-	-
Adressbereich	0xFF00 - 0xFFFF							

Befehl **OR**

Funktion Bitweises logisches ODER mit einem Register im internen Speicher. Die oberen 8 Bit der Adresse sind immer 0xFF

Byte	0	1	2	3	4	5	6	7
Befehlstelegramm	0xA5	D0	A0	CS	-	-	-	-
Antworttelegramm	0xA5	D0	A0	NCS	-	-	-	-
Adressbereich	0xFF00 - 0xFFFF							

6.2.3. Speicherzugriffe auf den externen Speicher

Befehl **Read Word X**

Funktion Auslesen eines 16 Bit Wertes aus dem externen Speicher

Byte	0	1	2	3	4	5	6	7
Befehlstelegramm	0xC9	A0	A1	0x3F	CS	-	-	-
Antworttelegramm	0xC9	A0	D0	D1	NCS	-	-	-
Adressbereich	0x0000 - 0xFFFF							

Befehl **Read Long X**

Funktion Auslesen eines 32 Bit Wertes aus dem externen Speicher

Byte	0	1	2	3	4	5	6	7
Befehlstelegramm	0xCB	A0	A1	0x3F	0x3F	0x3F	CS	-
Antworttelegramm	0xCB	A0	D0	D1	D2	D3	NCS	-
Adressbereich	0x0000 - 0xFFFF							

Befehl **Write Word X**

Funktion Schreiben eines 16 Bit Wertes in den externen Seicher

Byte	0	1	2	3	4	5	6	7
Befehlstelegramm	0x6A	D0	D1	A0	A1	CS	-	-
Antworttelegramm	0x6A	D0	D1	A0	A1	NCS	-	-
Adressbereich	0x4000 - 0xFFFF							

Befehl **Write Long X**

Funktion Schreiben eines 32 Bit Wertes in den externen Speicher

Byte	0	1	2	3	4	5	6	7
Befehlstelegramm	0x6C	D0	D1	D2	D3	A0	A1	CS
Antworttelegramm	0x6C	D0	D1	D2	D3	A0	A1	NCS
Adressbereich	0x4000 - 0xFFFF							

6.2.4. Steuerbefehle

Befehl **Reset**

Funktion Löst einen Reset des NOVODRIVE aus

Byte	0	1	2	3	4	5	6	7
Befehlstelegramm	0xDD	0x21	CS	-	-	-	-	-
Antworttelegramm	-	-	-	-	-	-	-	-

6.3 Prozessdatenkanal

Der Prozessdatenkanal realisiert einen zyklischen Datenaustausch zwischen Leitrechner und NOVODRIVE.

Der Ablauf sieht folgendermaßen aus:

1. Der Leitrechner schickt ein Sollwerttelegramm an den NOVODRIVE mit bis zu 8 Byte Sollwertdaten.

2. Der Leitrechner schickt ein Zeittakttelegramm.

3. Nach Empfang des Zeittakttelegramms durch den NOVODRIVE werden die Daten des Sollwerttelegramms in den Speicherbereich **CANinput1** - **CANinput4** kopiert. Anschließend wird das Istwerttelegramm zusammengestellt und abgesendet.

Auf die Daten des Sollwerttelegramms können die Softwaremodule des NOVODRIVE über Zeiger zugreifen. Die Istwerttelegramme können über Adressfelder beliebig konfiguriert werden.

6.3.1 Message Objekte auf dem NOVODRIVE

Message Objekt **CAN Time (Empfang von Zeittakttelegrammen)**

Identifier Bit	10	9	8	7	6	5	4	3	2	1	0
Register	7	6	5	4	3	2	1	0	15	14	13
	CANTime								**CANIDLSB**		
Adresse	0xFF3B								0xFED2		

Message Objekt **CAN Master (Empfang von Sollwerttelegrammen)**

Identifier Bit	10	9	8	7	6	5	4	3	2	1	0
Register	7	6	5	4	3	2	1	0	12	11	10
	CANMaster								**CANIDLSB**		
Adresse	0xFF39								0xFED2		

Message Objekt **CAN Slave (Senden von Istwerttelegrammen)**

Identifier Bit	10	9	8	7	6	5	4	3	2	1	0
Register	7	6	5	4	3	2	1	0	7	6	5
	CANSlave								**CANIDLSB**		
Adresse	0xFF3A								0xFED2		

6.3.2 Konfiguration

CANCFG	7	6	5	4	3	2	1	0	Bit
0xFF11	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Bit 0 Reserviert = 0
Bit 1 1 = Istwert Telegramme senden
Bit 2 - 5 Telegrammlänge von Istwerttelegrammen
0 - 8 Byte
Bit 6 Bytereihenfolge bei Sollwerttelegrammen
0 = Little Endian Mode
1 = Big Endian Mode
Bit 7 1 = Spezial Modus (siehe unten)

6.3.3 Belegung von Istwerttelegrammen

Der Inhalt von Istwerttelegrammen wird über ein Adressfeld ausgewählt.
Ausgewählt werden können alle Register im Adressbereich 0xFE00 - 0xFFFF des internen Speichers.

Register	Adresse	Inhalt
CANSlaveBit8	0xFF08	Auswahlbits für den Adressbereich. Ist das Bit i gesetzt, wird für das Telegrammbyte i der Speicherbereich 0xFFyy ausgewählt. Wenn das Bit i gelöscht ist, gilt der Speicherbereich 0xFEyy
CANSlaveB0	0xFF09	Die unteren 8 Bit der Adresse für das Telegrammbyte 0
CANSlaveB1	0xFF0A	Die unteren 8 Bit der Adresse für das Telegrammbyte 1
CANSlaveB2	0xFF0B	Die unteren 8 Bit der Adresse für das Telegrammbyte 2
CANSlaveB3	0xFF0C	Die unteren 8 Bit der Adresse für das Telegrammbyte 3
CANSlaveB4	0xFF0D	Die unteren 8 Bit der Adresse für das Telegrammbyte 4
CANSlaveB5	0xFF0E	Die unteren 8 Bit der Adresse für das Telegrammbyte 5
CANSlaveB6	0xFF0F	Die unteren 8 Bit der Adresse für das Telegrammbyte 6
CANSlaveB7	0xFF10	Die unteren 8 Bit der Adresse für das Telegrammbyte 7

Beispiel	Register	Inhalt	Telegrammaufbau Nutzdaten 0 - 8
	CANSlaveBit8	1000 0001	
	CANSlaveB0	0xF2	0xFFF2 **Status**
	CANSlaveB1	0x68	0xFE68 **nIst** (MSB)
	CANSlaveB2	0x69	0xFE69 **nIst** (LSB)
	CANSlaveB3	0x7C	0xFE7C **LageIst** (MSB)
	CANSlaveB4	0x7D	0xFE7D **Lageist** (LSB)
	CANSlaveB5	0x00	0xFE00 **errorcode** (MSB)
	CANSlaveB6	0x01	0xFE01 **errorcode** (LSB)
	CANSlaveB7	0xFF	0xFFFF

6.3.4 Bytereihenfolge bei Sollwerttelegrammen

Die Bytereihenfolge (Big Endian oder Little Endian Modus) ist durch das Register **CANCfg** einstellbar.

Sollwerttelegramm (8 Bytes)							
b0	b1	b2	b3	b4	b5	b6	b7

CANCfg Bit 6		1		0	
		Big Endian Mode		Little Endian Mode	
Register	Adresse	MSB	LSB	MSB	LSB
CANinput1	0xFE3C	b0	b1	b7	b6
CANinput2	0xFE3E	b2	b3	b5	b4
CANinput3	0xFE40	b4	b5	b3	b2
CANinput4	0xFE42	b6	b7	b1	b0

Auf die Telegrammdaten in **CANinput1** ... **CANinput4** kann durch Zeigerparameter zugegriffen werden.

Zeigername	Beschreibung	Adresse	Zugriff	Bit	Standardwert
?LageSollExt	Bei Lageregelung Solllage	0xFEA6	R/W	16	0
?Rampe+	Beschleunigungsrampe	0xFEA8	R/W	16	**Rampe+**
?Rampe-	Bremsrampe	0xFEAA	R/W	16	**Rampe-**
?SchRampe	Schnellstoprampe	0xFEC4	R/W	16	**SchRampe**
?msoll	Bei Drehmomentregelung: Sollmoment	0xFEBC	R/W	16	**msoll**
?FIUmdr	Eingangswert Feininterpolator bzw. PS0	0xFECC	R/W	16	-
?FILage	Eingangswert Feininterpolator bzw. PS0	0xFEB0	R/W	16	-
?CANControl	siehe **CANcontrol**	0xFEB2	R/W	16	**CANControl**
?mmax	Begrenzung Spitzenmoment	0xFF36	R/W	16	**mmax16**
?Sollwert	Drehzahlsollwert	0xFEBE	R/W	16	**Sollwert**
?GPO	Ausgabewerte für die Digitalausgänge	0xFEC2	R/W	16	**GPIO**
?AnOut1	Ausgabewert Analogausgang 1	0xFF24	R/W	16	-
?AnOut2	Ausgabewert Analogausgang 2	0xFF12	R/W	16	-
	bzw. PS0 Positioniergeschwindigkeit				
?FForward	Vorsteuerung Drehzahl	0xFEB6	R/W	16	0

Beispiel **Little Endian Modus eingestellt**

?Sollwert = Adresse von **CANinput2**
?CANControl = Adresse von **CANinput1**

Die Geschwindigkeitssollwerte für den Regler müssen nun in den Bytes b5 und b4 des Sollwerttelegramms stehen.

Ausserdem kann der Regler über das Byte b7 freigegeben und gestartet bzw. gestoppt werden.

6.3.5 Spezial Modus

Das Sollwerttelegramm ist vorbelegt mit:

b0	b1	b2	b3	b4	b5	b6	b7
CANControl	frei	frei	frei	frei	frei	frei	frei
CANinput1		**CANInput2**		**CANInput3**		**CANInput4**	

Das Istwertelegramm ist vorbelegt mit:

b0	b1	b2	b3	b4	b5	b6	b7
Status	**nist** MSB	**nist** LSB	**Lageist** MSB	**Lageist** LSB	**?CANOut** MSB	**?CANOut** LSB	-

6.3.6 Timeout

Bleiben die Zeittakttelegramme für eine längere Zeit als in **CANTimeout** angegeben aus, dann wird der Antrieb gesperrt und es wird der Fehler 518 ausgegeben. Die Überwachung beginnt mit dem ersten Zeittakttelegramm.

Register	**CANTimeout**
Adresse	0xFF16
Größe	8 Bit
Zugriff	R/W
Funktion	Timeoutwert für Prozessdatenkanal
Skalierung	10ms / Bit
Wertebereich	0x00 keine Überwachung 0xFF keine Überwachung 0x01 ... 0xFE Überwachung ein

6.3.7 Bus Reaktivierung

Nach dem Auftreten von 255 Übertragungsfehlern geht der CAN Controller in den BOFF (= Bus off) Zustand. Nach einem BOFF kann eine automatische Reaktivierung durch den NOVODRIVE erfolgen.

SWVersion	7	6	5	4	3	2	1	0
0xFF3D	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

0 — keine Reaktivierung nach BOFF

1 — Reaktivierung des CAN-Bus nach BOFF

6.4 Feininterpolator

Der Feininterpolator ist ab der H8 Version vom 5.6.1997 verfügbar.

6.4.1 Funktion

Vom Leitrechner werden Positionssollwerte auf dem CAN-Bus zyklisch an den NOVODRIVE gesendet. Die Zykluszeit wird dem NOVODRIVE in dem Register Tcycle mitgeteilt. Der Feininterpolator berechnet aus diesen Angaben eine Sollgeschwindigkeit für den NOVODRIVE, sodass der Positionssollwert mit dem nächsten Zeittakttelegramm erreicht wird.

Die Positionsvorgabe kann absolut mit 32 Bit oder relativ mit 16 Bit erfolgen. Die Positionsvorgabe kann eine Auflösung von 16 oder 12 Bit pro Umdrehung aufweisen. Mit dem Parameter **FIVerrundung** lassen sich die Ecken der Linearinterpolation verrunden. Die Verrundung ist mit dem Wert 0 abgeschaltet. Je größer der Wert, umso stärker ist die Verrundung. Mit der Vergrößerung der Verrundung nimmt auch der Schleppfehler zu.

Ein relativer Sollwert oder die Differenz zum letzten absolutem Sollwert darf nie mehr als ±28670 Inkremente (Auflösung 16 Bit) betragen, sonst findet ein Überlauf statt.

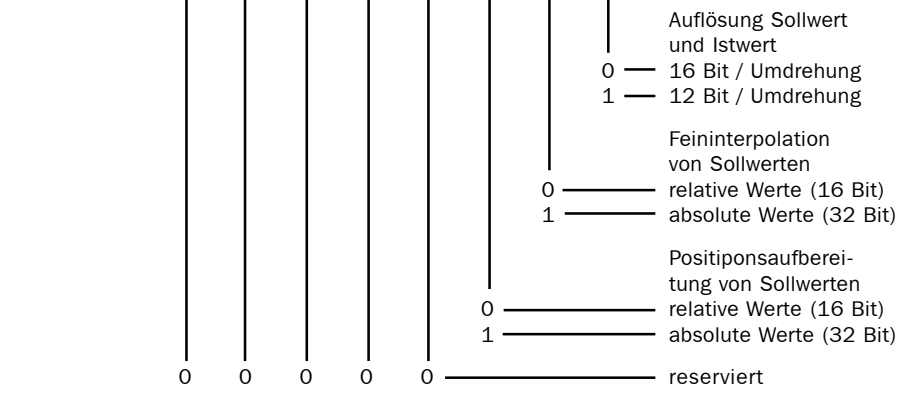
Mit dem Positionsaufbereiter werden die Istwerte zum Zeitpunkt des nächsten Empfangs des Zeittakttelegramms vorausberechnet. Dadurch können zeitlich exaktere Istwerte an den Leitrechner zurückgesendet werden. Die Istwerte können relativ 16 Bit und absolut 32 Bit groß sein. Die Auflösung kann 16 oder 12 Bit pro Umdrehung betragen.

Vor der Reglerfreigabe müssen die Positionssollwerte und die aktuelle Position des Antriebs abgeglichen werden, damit es nicht zu einem unkontrollierten Motorlauf mit Höchstdrehzahl kommt.

Die CAN Timeout Überwachung sollte aktiviert sein. Sonst fährt der Antrieb bei einer Unterbrechnung der Verbindung mit konstanter Geschwindigkeit weiter.

6.4.2 Register

FICSR	R	R	R	R	R	R/W	R/W	R/W
0xFF06	7	6	5	4	3	2	1	0



Register	**?512usA**
Adresse	0xFEC8
Größe	16 Bit vorzeichenlos
Zugriff	R/W
Funktion	Funktionsaufruf für den Feininterpolator
Wertebereich	Name Adresse Hinweise @**Dummy** 0x01EA Feininterpolator deaktiviert @**Feininterpolator** 0x01E8 Feininterpolator aktiviert

Register	**FIVerundung**
Adresse	0xFEEE
Größe	8 Bit vorzeichenlos
Zugriff	R/W
Funktion	Verrundung der Linearinterpolation
Wertebereich	0 = keine Verrundung 1 ... 10 Verrundung

Register	Tcycle
Adresse	0xFEEA
Größe	16 Bit vorzeichenlos
Zugriff	R/W
Funktion	Zykluszeit von 1ms ... 6,5ms
Skalierung	200ns / Bit

Register	?FILage		
Adresse	0xFEB0		
Größe	16 Bit vorzeichenlos		
Zugriff	R/W		
Funktion	Zeiger auf die unteren 16 Bit des Lage-Sollwerts für die Feininterpolation		
Wertebereich	Name	Adresse	Hinweise
	CANInput1	0xFE3C	siehe Abschnitt 6.3.4
	CANInput2	0xFE3E	
	CANInput3	0xFE40	
	CANInput4	0xFE42	

Register	?FIUmdr		
Adresse	0xFECC		
Größe	16 Bit vorzeichenlos		
Zugriff	R/W		
Funktion	Zeiger auf die oberen 16 Bit des Lage-Sollwerts für die Feininterpolation (nur bei absoluter Lagevorgabe notwendig)		
Wertebereich	Name	Adresse	Hinweise
	CANInput1	0xFE3C	siehe Abschnitt 6.3.4
	CANInput2	0xFE3E	
	CANInput3	0xFE40	
	CANInput4	0xFE42	

Register	FISoll
Adresse	0xFE1A
Größe	16 Bit mit Vorzeichen
Zugriff	R
Funktion	Der vom Feininterpolator berechnete Drehzahlsollwert. Die Sollwertvorgabe wird durch Setzen von ?Sollwert auf den Feininterpolator umgestellt.
Skalierung	siehe Grundfunktionen Abschnitt 3.6.5

Register	Pa0
Adresse	0xFE6A
Größe	16 Bit mit Vorzeichen
Zugriff	R
Funktion	Der vom Positionsaufbereiter berechnete Lageistwert zum nächsten Zeittakt. Im absoluten Modus enthält das Register die oberen 16 Bit der Lage, im relativen Modus ist es undefiniert.
Skalierung	siehe Grundfunktionen Abschnitt 3.6.3.5

Register	Pa1
Adresse	0xFE6C
Größe	16 Bit mit Vorzeichen
Zugriff	R
Funktion	Der vom Positionsaufbereiter berechnete Lageistwert zum nächsten Zeittakt. Im absoluten Modus enthält das Register die unteren 16 Bit der Lage, im relativen Modus die Lageänderung.
Skalierung	siehe Grundfunktionen Abschnitt 3.6.3.5

6.4.3 Initialisierung

1. Der NOVODRIVE muss gestoppt sein.
2. Einschalten des Lagereglers durch Setzen des Registers **?nsoll** auf die Adresse von **nsoll2**.
3. Die Sollwertvorgabe erfolgt über das Register **FISoll** (**?Sollwert** = **FISoll**).
4. Die Zeiger **?FIUmdr** und **?FILage** auf das Sollwerttelegramm in **CANinput1** ... **CANinput4** setzen.
5. Die Zykluszeit im Register **Tcycle** setzen.
6. Die Timeout-Überwachung muss eingeschaltet werden (siehe Abschnitt 6.3.6).
7. Der Feininterpolator muss über das Register **FICSR** konfiguriert werden.

☞ Es wird empfohlen, die Beschleunigungsrampe und die Bremsrampe abzuschalten (siehe Handbuch Grundfunktionen Abschnitt 3.6.3.4). Ansonsten kann es zur Verfälschung der Zielposition kommen, wenn der Sollwertverlauf die Rampenwerte überschreitet.

8. Die Funktion wird aktiviert durch Setzen des Registers **?512usA** auf **@Feininterpolator**.
9. Der Lagewert in der Steuerung muss auf den Lagewert im Antrieb angepasst werden. Dazu kann der Lage-Istwert zyklisch über die Istwerttelegramme ausgelesen werden. Alternativ kann durch Istwert-Setzen auch die Lage im NOVODRIVE an die Steuerung angepasst werden (siehe dazu Zusatzfunktionen Abschnitt 6).

10 Vor dem Start der Feininterpolation muss der zyklische Datenaustausch über den Prozessdatenkanal laufen.
11 Erst dann darf der NOVODRIVE in den Startzustand versetzt werden.

6.4.4 Beispiele

Vorgabe 32 Bit absolut mit 16 Bit Auflösung:
- FICSR mit 0xxx xx10 initialisieren

Vorgabe 32 Bit absolut mit 12 Bit Auflösung:
- FICSR mit 0xxx xx11 initialisieren

Vorgabe 16 Bit relativ mit 16 Bit Auflösung:
- FICSR mit 0xxx xx00 initialisieren

Vorgabe 16 Bit relativ mit 12 Bit Auflösung:
- FICSR mit 0xxx xx01 initialisieren

Vorgabe 32 Bit absolut, Bytes b0 – b3 des CAN-Sollwert-Telegramms, Little Endian Mode (Bit 6 in **CANCFG** = 0):
- in **?FILage** 0xFE42 eintragen
- in **?FIUmdr** 0xFE40 eintragen

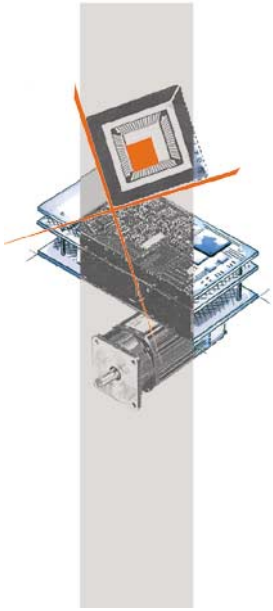
Vorgabe 32 Bit absolut, Bytes b4 – b7 des CAN-Sollwert-Telegramms, Little Endian Mode (Bit 6 in **CANCFG** = 0):
- in **?FILage** 0xFE3E eintragen
- in **?FIUmdr** 0xFE3C eintragen

Vorgabe 32 Bit absolut, Bytes b0 – b3 des CAN-Sollwert-Telegramms, Big Endian Mode (Bit 6 in **CANCFG** = 1):
- in **?FILage** 0xFE3C eintragen
- in **?FIUmdr** 0xFE3E eintragen

Vorgabe 32 Bit absolut, Bytes b4 – b7 des CAN-Sollwert-Telegramms, Big Endian Mode (Bit 6 in **CANCFG** = 1):
- in **?FILage** 0xFE40 eintragen
- in **?FIUmdr** 0xFE42 eintragen

Vorgabe 16 Bit relativ, Bytes b0 – b1 des CAN-Sollwert-Telegramms, Big Endian Mode (Bit 6 in **CANCFG** = 1):
- in **?FILage** 0xFE3C eintragen

Vorgabe 32 Bit relativ, Bytes b0 – b1 des CAN-Sollwert-Telegramms, Little Endian Mode (Bit 6 in **CANCFG** = 0):
- in **?FILage** 0xFE42 eintragen



NOVOTRON

für Dynamik und Bewegung

N O V O T R O N

Industrie - Automation GmbH
Mauserstrasse 31
D - 71640 Ludwigsburg
Telefon 07141/2969 - 0
Telefax 07141/2969 - 22
e-mail: info@novotron-online.com
http: //www.novotron-online.com