AC21

Baugruppe AC21 – Steuerung von 21 Servomotoren
Firmware-Beschreibung

Handbuch

Ab Version 0.30

Stand 15/05/2015

System-Software-Entwicklung
Dipl.Inform.
Thomas Schneider

Im Schieb 22 40668 Meerbusch

Tel. 02150 911 747
Fax 02150 911 748

Thomas.Schneider@sys-thos.de

pfortner

Ing. Büro Pfortner

Krautstückerweg 13
76706 Dettenheim

Tel. +49 7247 906060
Fax +49 7247 906063

wopfo@t-online.de
http://www.pfortner.de

Inhaltsverzeichnis

1 Übersicht

Die AC21 ist eine Elektronikplatine zur Ansteuerung von bis zu 21 sensor- und bürstenlosen, 3phasigen Synchronmotoren. Die Kommunikation erfolgt über einen CAN – Bus auf Basis von CAN-OPEN oder über eine RS232 – Schnittstelle. Jeder Motor wird dabei von einem eigenen Prozessor überwacht.

Die Kommunikation über den CAN-Bus vermittelt ein weiterer Prozessor, der über einen internen Bus die Motor-Prozessoren ansteuert. Auf diese Weise kann die CAN-Bus-Belastung auf ein Minimum reduziert werden. Der CAN-Bus ist durchgeschleift, so daß mehrere Module problemlos hintereinander angeschlossen werden können.
Zusätzlich ist zu Service-Zwecken eine serielle Schnittstelle vorhanden; das Service-Programm „AC21.EXE“ (s. Kap. 3) erlaubt die Bedienung eines einzelnen Moduls über die serielle Schnittstelle oder mehrerer Module über den CAN-Bus.

Das implementierte CAN-OPEN-Protokoll basiert auf dem Device-Service-Profil DSP-402 mit spezifischen Erweiterungen, die im folgenden beschrieben werden. Zum weiteren Verständnis werden grundlegende Kenntnisse zum Protokoll CAN-OPEN bzw. der entsprechenden Dokumente DS-301 – DS-303 und DSP-402 vorausgesetzt.

1.1 Technische Daten

Stromversorgung	nominell 24V DC +/- 5%, max. 20A 18V – 28V möglich, mit Ninitialisierung der Motore
Stromaufnahme	210mA – 230mA im Leerlauf ca. 16 - 20A bei voller Last mit 21 Motoren
Ausgangsleistung	je Motorkanal max. 800 – 900mA, ca. 20W pro Motor
Betriebsarten	getrennt programmierbar für jeden Motor: • Drehzahlregelung von 0 – 500 U/min • DC – Betrieb mit Vorgabe der Amplitude in % der max. PWM (entspricht einem DC-Motor mit Vorgabe der Klemmenspannung) • Haltemoment mit Vorgabe der Amplitude in % der PWM bei Nennstrom (entspricht einem Schrittmotor mit konstanter Bestromung der Phasen im Stillstand) • Positionierbetrieb (ab V.0.90)
CAN-Bus	Default: 1 MBd, ab V.0.90: 500kBd und 250kBd möglich Node – Adresse frei konfigurierbar (Auslieferungszustand = Node 126) Protokoll auf CAN-open aufbauend, wie in AC21.pdf beschrieben
RS232	Geschwindigkeit 115,2 kBd, Format 8N1 Bedienung über Programm AC21.exe wie in Kap. 3 beschrieben

1.2 Abbildungen

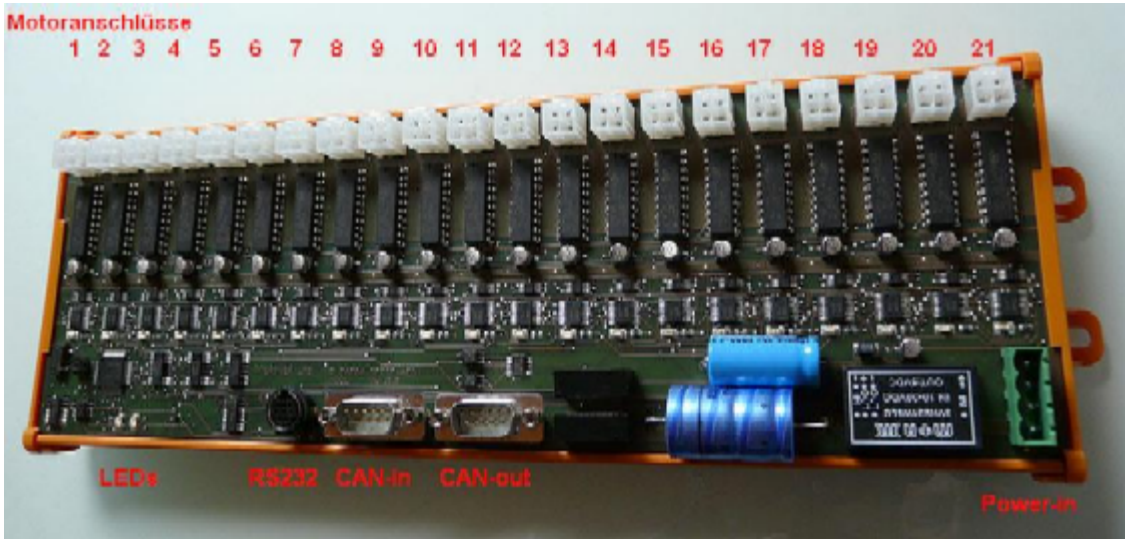


Abbildung 1: AC21

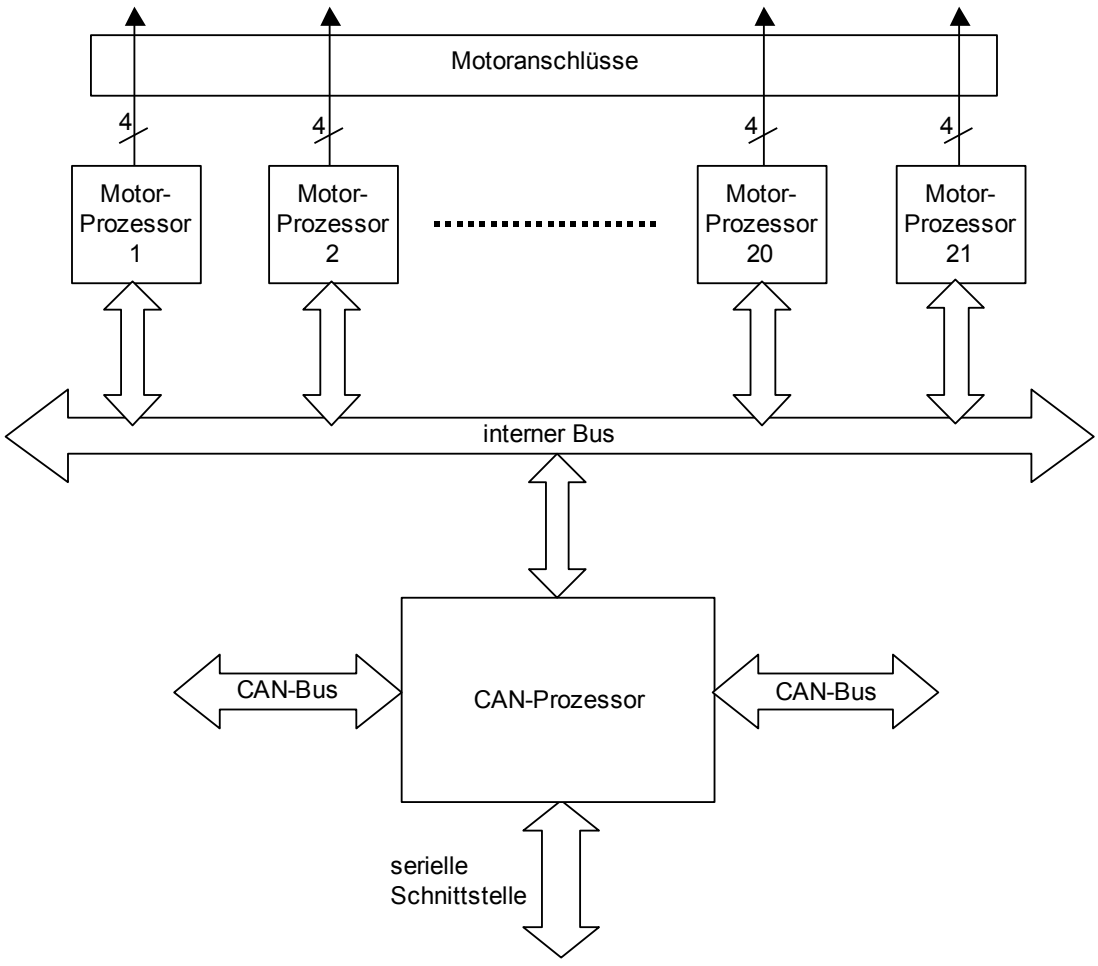
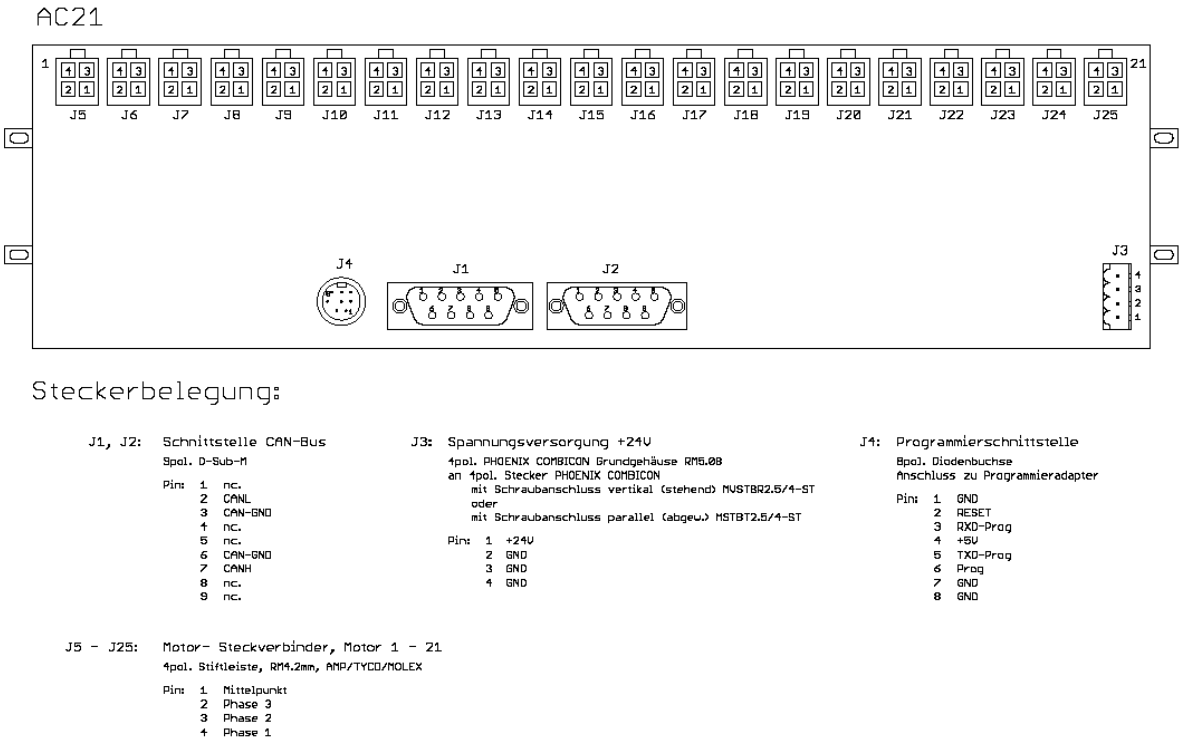


Abbildung 2: AC21 - Kommunikationsschema



Steckerbelegung:

J1, J2: Schnittstelle CAN-Bus
9pol. D-Sub-M

Pin:		
	1	nc.
	2	CANL
	3	CAN-GND
	4	nc.
	5	nc.
	6	CAN-GND
	7	CANH
	8	nc.
	9	nc.

J3: Spannungsversorgung +24V
4pol. PHOENIX COMBICON Grundgehäuse RM5.08
an 4pol. Stecker PHOENIX COMBICON
mit Schraubanschluss vertikal (stehend) MVSTBR2.5/4-ST
oder
mit Schraubanschluss parallel (abgew.) MSTBT2.5/4-ST

Pin:		
	1	+24V
	2	GND
	3	GND
	4	GND

J4: Programmierschnittstelle
8pol. Diodenbuchse
Anschluss zu Programmieradapter

Pin:		
	1	GND
	2	RESET
	3	RXD-Prog
	4	+5V
	5	TXD-Prog
	6	Prog
	7	GND
	8	GND

J5 - J25: Motor- Steckverbinder, Motor 1 - 21
4pol. Stiftleiste, RM4.2mm, AMP/TYCO/MOLEX

Pin:		
	1	Mittelpunkt
	2	Phase 3
	3	Phase 2
	4	Phase 1

Abbildung 3: Steckerbelegung

2 CAN-Kommunikation

2.1 Allgemeines

Die Kommunikation erfolgt über CAN-Bus nach dem Protokoll CAN-OPEN, das implementierte Profil orientiert sich an DSP-402. Da dieses Profil aber nur bis zu 8 Geräte unterstützt, wurden die profilspezifischen Objekte 0x6000-0x9FFF nicht implementiert. Statt dessen wird der herstellerspezifische Bereich 0x2000-0x5FFF benutzt; Objekte im Bereich 0x5000-0x57FF beeinflussen alle Motoren gleichzeitig. Objekte, die einzelne Motoren betreffen befinden sich im herstellerspezifischen Bereich 0x3000-0x4FFF. Insgesamt ergibt sich folgende Einteilung der Objekte:

Übersicht Object Dictionary:

0x0000..0x0FFF:	Datentypen Es werden nur die Standarddatentypen benutzt
0x1000..0x1FFF:	Communication Profile Area
0x2000..0x5FFF:	Manufacturer Specific Profile Area
0x2000..0x2FFF:	allgemeine Objekte
0x3000..0x3FFF	motorspezifische Objekte vom Objekttyp OT_ARRAY; die Daten für die einzelnen Motoren finden sich unter den Sub-Indizes 1..21
0x4000..0x4CFF:	motorspezifische Objekte; die Daten für die einzelnen Motoren finden sich jeweils im Abstand 0x80:
0x4000..0x407F:	Motor 1
....	
0x4A00..0x4A7F:	Motor 21
0x4D00..0x4CFF:	unbenutzt
0x5000..0x57FF:	Objekte, die global alle Motoren beeinflussen
0x5800..0x5FFF:	unbenutzt
0x6000..0x9FFF:	Standardized Device Profile Area unbenutzt!
0xA000..0xBFFF:	Standardized Interface Profile Area unbenutzt!
0xC000..0xFFFF:	reserved

2.2 Startup

Nach dem Einschalten läuft der nach CAN-OPEN definierte Startup-Prozess; danach befindet sich das Modul im Zustand „PREOPERATIONAL“. Der Master kann nun über das NMT-Kommando (COB-ID 0x000) das Modul in den gewünschten Zustand setzen (START, STOP) bzw. neu initialisieren (RESET COMMUNICATION, RESET NODE).

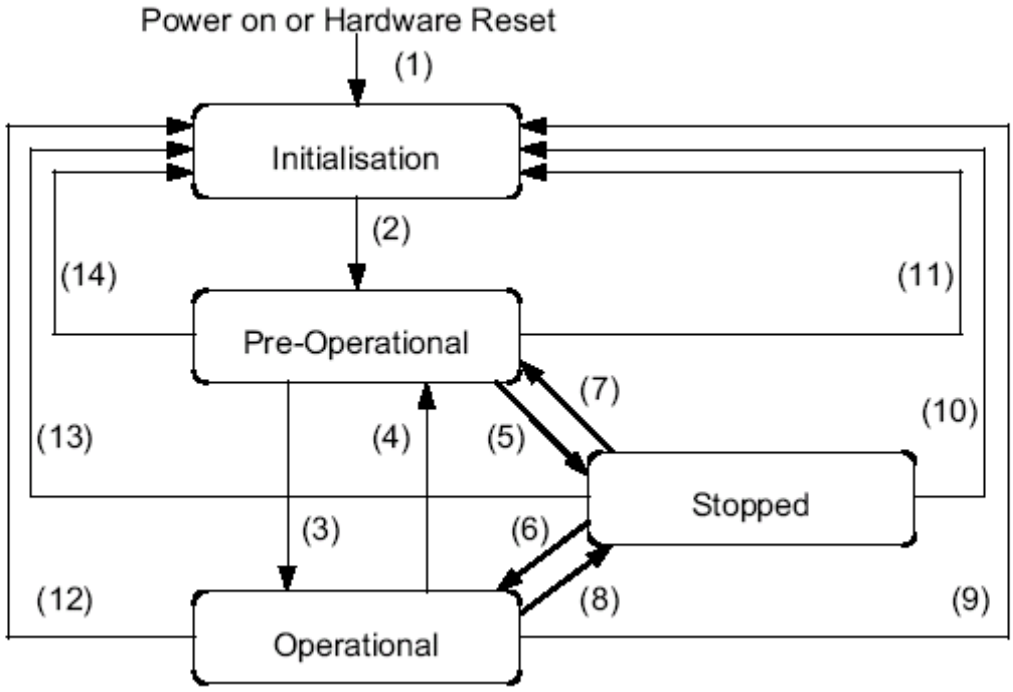


Abbildung 4: CAN-OPEN Startup-Diagramm

Die Kommunikation erfolgt mit 1MBaud.

Optional sendet das Modul über die COB-ID „0x700+NODE-ID“ eine Heartbeat-Message; die Heartbeat-Zeit ist einstellbar über das Object 1017 (Default = 0: keine Heartbeat-Message). Zusätzlich ist das „Node Guarding Protocol“ implementiert, d.h das Modul antwortet auf Anfragen mit der COB-ID „0x700+NODE-ID“; Details zur Modul-Überwachung s. Kap. 2.5.

Die NODE-ID ist per Default 126; über das Objekt 0x2000.01 kann sie geändert werden und ist dann nach dem nächsten NMT-Kommando „RESET COMMUNICATION“ gültig. Dauerhaft kann dieser Wert über das Objekt 0x1010 / STORE PARAMETERS im internen Flash-ROM gespeichert werden.

***Hinweis**:* *Es sind nicht alle NODE-IDs zulässig; weitere Details dazu s. Kap. 2.6.*

Die CAN-Bus-Baudrate beträgt per Default 1MBd; über das Objekt 0x2000.02 kann sie geändert werden und ist dann nach dem nächsten NMT-Kommando „RESET COMMUNICATION“ gültig. Dauerhaft kann dieser Wert über das Objekt 0x1010 / STORE PARAMETERS im internen Flash-ROM gespeichert werden.

***Hinweis**:* *Weitere Details zum Ändern von Node-ID und Baudrate s. Kap. 2.6.*

2.2.1 Sicherung der Betriebsparameter

Die Betriebsparameter werden im internen Flash-Speicher der einzelnen Prozessoren gesichert, so daß sie nach Programmstart automatisch wieder zur Verfügung stehen. Zuständig für die Sicherung sind die Objekte 0x1010 bzw. 0x3412; weitere Hinweise finden sich dort bzw. bei der Beschreibung der einzelnen Objekte.

Firmware-Version 0.61:

Ab dieser Firmware-Version wird im CAN-Prozessor ein Backup der Flash-Daten der einzelnen Motorprozessoren verwaltet. Bei Programmstart wird während der Bootup-Phase die Korrektheit und die Plausibilität der Flash-Daten in den Motor-Prozessoren und der Backup-Daten überprüft. Bei Bedarf werden automatisch die Flash-Daten in den Motor-Prozessoren mit den Backup-Daten im CAN-Prozessor synchronisiert.
Die Synchronisierung kann mit dem Objekt 0x2404 explizit durchgeführt werden. Dies sollte immer dann geschehen, **nachdem(!)** mittels Objekt 0x3412 Parameter-Daten im Motor-Daten-Flash gesichert wurden.

***Hinweis**:* *Das Objekt 0x2404 synchronisiert **immer** alle Motor-Prozessoren. Es empfiehlt sich deshalb, dieses Objekt erst zu schreiben, wenn alle erforderlichen Änderungen an den Motorprozessoren durchgeführt wurden.*

2.3 Object Dictionary

***Hinweis**:* *Eine nach CAN-OPEN standardisierte Beschreibung („electronic data sheet“) der einzelnen Objekt findet sich unter „AC21.eds“. Die Datei wurde jedoch nicht auf Konformität getestet und enthält daher möglicherweise syntaktische Fehler.*

***Hinweis**:* *Die nach CAN-OPEN standardisierten Objekte werden im folgenden nur aufgeführt und nicht weiter beschrieben.*

2.3.1 Übersicht

Index.sub (hex)	Objekt-Typ	Name	Datentyp		Zugriff
Communication Profile Area					
1000	VAR	device type	UNSIGEND32		CONST
1001	VAR	error register	UNSIGNED8		RO
1002	VAR	manufacturer status register	UNSIGNED32		RO
1003	ARRAY	emergency handle	UNSIGNED32		RO
1008	VAR	manufacturer device name	VISIBLE_STRING		CONST
1009	VAR	manufacturer hardware version	VISIBLE_STRING		CONST
100A	VAR	manufacturer software version	VISIBLE_STRING		CONST
100B	VAR	node id	UNSIGNED32		RO
100C	VAR	guard time	UNSIGNED16		RW
100D	VAR	lifetime factor	UNSIGNED8		RW
1010	ARRAY	store parameters	UNSIGNED32		RW
1011	ARRAY	restore default parameters	UNSIGNED32		RW
1014	VAR	COB-ID EMCY	UNSIGNED32		RW
1015	VAR	Inhibit Time EMCY	UNSIGNED16		RW
1017	VAR	Producer heartbeat time	UNSIGNED16		RW
1018.00-04	RECORD	Identity Object			RO
1018.00	VAR	number of entries	UNSIGNED8		RO
1018.01	VAR	Vendor ID	UNSIGNED32		CONST
1018.02	VAR	Product code	UNSIGNED32		RO
1018.03	VAR	Revision	UNSIGNED32		RO
1018.04	VAR	Serial no	UNSIGNED32		RO
14xx		RPDO Communication parameter			
1400.00-02	RECORD	1. Receive PDO Parameter	PDO CommPar		RO
1400.00	VAR	number of entries	UNSIGNED8		RO
1400.01	VAR	COB-ID	UNSIGNED32		RO
1400.02	VAR	transmission type	UNSIGNED8		CONST
1401.00-02	RECORD	2. Receive PDO Parameter	PDO CommPar		RO
1402.00-02	RECORD	3. Receive PDO Parameter	PDO CommPar		RO
1403.00-02	RECORD	4. Receive PDO Parameter	PDO CommPar		RO
1404.00-02	RECORD	5. Receive PDO Parameter	PDO CommPar		RO
1405.00-02	RECORD	6. Receive PDO Parameter	PDO CommPar		RO
1406.00-02	RECORD	7. Receive PDO Parameter	PDO CommPar		RO
1407.00-02	RECORD	8. Receive PDO Parameter	PDO CommPar		RO
1408.00-02	RECORD	9. Receive PDO Parameter	PDO CommPar		RO
16xx	NULL	RPDO-Mapping – nicht implementiert			

18xx		TPDO Communication parameter			
1800.00-05	RECORD	1. Transmit PDO Parameter	PDO CommPar		RO
1800.00	VAR	number of entries	UNSIGNED8		RO
1800.01	VAR	COB-ID	UNSIGNED32		RO
1800.02	VAR	transmission type	UNSIGNED8		CONST
1800.03	VAR	inhibit time	UNSIGNED16		RO
1800.04	VAR	reserved	UNSIGNED8		CONST
1800.05	VAR	event time	UNSIGNED16		RO
1801.00-05	RECORD	2. Transmit PDO Parameter	PDO CommPar		RO
1802.00-05	RECORD	3. Transmit PDO Parameter	PDO CommPar		RO
1803.00-05	RECORD	4. Transmit PDO Parameter	PDO CommPar		RO
1804.00-05	RECORD	5. Transmit PDO Parameter	PDO CommPar		RO
1805.00-05	RECORD	6. Transmit PDO Parameter	PDO CommPar		RO
1806.00-05	RECORD	7. Transmit PDO Parameter	PDO CommPar		RO
1807.00-05	RECORD	8. Transmit PDO Parameter	PDO CommPar		RO
1Axx	NULL	TPDO-Mapping – nicht implementiert			
Manufacturer Specific Profile Area - Allgemeine Objekte					
2000.00-03	RECORD	Parameter			
2000.00	VAR	number of entries	UNSIGNED8		RO
2000.01	VAR	NodeID	UNSIGNED8		RW
2000.02	VAR	BaudID	UNSIGNED8		RW
2000.03	VAR	Heartbeat time	UNSIGNED16		RW
2001	VAR	NodeState	UNSIGNED8		RO
200A.00-03	RECORD	Version information			
200A.00	VAR	number of entries	UNSIGNED8		RO
200A.01	VAR	software version no	UNSIGNED32		RO
200A.02	VAR	version date	UNSIGNED32		RO
200A.03	VAR	hardware version no	UNSIGNED32		RO
2404	VAR	BackupSync	UNSIGNED32		WO
2420	VAR	Axis Status Polltime	UNSIGNED16		RW
2431	VAR	Axis Communication timeout	UNSIGNED16		RW
Manufacturer Specific Profile Area – Achsspezifische Objekte • Alle Objekte sind vom Typ „ARRAY“ • Subindex 0: Anzahl der Achsen • Subindex = Achsnummer 1-21: Daten für die jeweilige Achse					
3000.00-15	ARRAY	Axis configuration	UNSIGNED32		RW
3001.00-15	ARRAY	Axis software version no	UNSIGNED16		RO
3011.00-15	ARRAY	Axis ramp	UNSIGNED16		RW
3014.00-15	ARRAY	UMax	UNSIGNED16		RW
3015.00-15	ARRAY	PFac	UNSIGNED16		RW
3018.00-15	ARRAY	VFac	UNSIGNED16		RW
3019.00-15	ARRAY	PDFac	UNSIGNED16		RW
301C.00-15	ARRAY	Axis Guard Time	UNSIGNED16		RW
301D.00-15	ARRAY	INenn	UNSIGNED16		RW
3027.00-15	ARRAY	AmpLimit	UNSIGNED16		RW

3101.00-15	ARRAY	AmpMin	INTEGER16		RW
3412.00-15	ARRAY	Init Parameters	UNSIGNED32		WO
Manufacturer Specific Profile Area – Achsspezifische Objekte • Für jede Achse sind 128 Objekt-Indizes reserviert • Die Objekte für die Achse n (n = 1-21) sind unter dem Index 0x4000+(n-1)*0x80 implementiert					
4000-407F	Objekte für die Achse 1:				
4000.00	VAR	number of entries	UNSIGNED8		RO
4000.01	VAR	Axis control word	UNSIGNED16		RW
4000.02	VAR	Axis status word	UNSIGNED16		RO
4000.03	VAR	Axis USoll	INTEGER16		RW
4000.04	VAR	Axis Uist	INTEGER16		RO
4000.05	VAR	Axis ISoll	UNSIGNED16		RW
4000.06	VAR	Axis ASoll	UNSIGNED16		RW
4000.07	VAR	Axis Distanz	SIGNED16		RW
4001-407F	Reserviert – weitere Objekte für die Achse 1				
	Analog für die Achsen 2-21:				
4080-40FF	Objekte für die Achse 2				
4100-417F	Objekte für die Achse 3				
4180-41FF	Objekte für die Achse 4				
4200-427F	Objekte für die Achse 5				
4280-42FF	Objekte für die Achse 6				
4300-437F	Objekte für die Achse 7				
4380-43FF	Objekte für die Achse 8				
4400-447F	Objekte für die Achse 9				
4480-44FF	Objekte für die Achse 10				
4500-457F	Objekte für die Achse 11				
4580-45FF	Objekte für die Achse 12				
4600-467F	Objekte für die Achse 13				
4680-46FF	Objekte für die Achse 14				
4700-477F	Objekte für die Achse 15				
4780-47FF	Objekte für die Achse 16				
4800-487F	Objekte für die Achse 17				
4880-48FF	Objekte für die Achse 18				
4900-497F	Objekte für die Achse 19				
4980-49FF	Objekte für die Achse 20				
4A00-4A7F	Objekte für die Achse 21				
Manufacturer Specific Profile Area – Achskontrolle für alle Achsen gemeinsam					
5000.00-01	RECORD	Axis status			
5000.00	VAR	number of entries	UNSIGNED8		RO
5000.01	VAR	control word	UNSIGNED16		WO
5060	VAR	modes of operation	UNSIGNED8		RO
5061	VAR	modes of operation display	UNSIGNED8		RO
5100	VAR	UMin0 – min. Drehzahl	UNSIGNED16		RW
5101	VAR	AMin0 – Startwert für AmpMin	UNSIGNED16		RW
5102	VAR	Rollendurchmesser	UNSIGNED16		RW

Tabelle 1: Objekt-Verzeichnis

2.3.2 Communication Profile Area

Die folgenden Objekte sind beschrieben im Dokument CiA Draft Standard 301 V.4.02. In dieser Übersicht werden nur einige spezielle Implementierungsdetails dokumentiert.

2.3.2.1 Objekt 0x1000 DEVICETYPE

2.3.2.2 Objekt 0x1001 ErrorRegister

2.3.2.3 Objekt 0x1002 Manufacturer Status Register

2.3.2.4 Objekt 0x1003 Error Handle SDO

2.3.2.5 Objekt 0x1008 Device Name

2.3.2.6 Objekt 0x1009 Hardware Version

2.3.2.7 Objekt 0x100A Version Information

2.3.2.8 Objekt 0x100B Node ID

Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED8
Default: 126
Zugriff: RO

Dieses Objekt liefert die aktuell gültige Node-ID. Eine Änderung ist über das Objekt 0x2000.01 möglich, s.dort. Außerdem sind nicht alle NodeIDs zugelassen, s. Kap. 2.3.3.1.

2.3.2.9 Objekt 0x100C guard time

Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED16
Default: 0
Zugriff: RW

Ein Wert <> 0 aktiviert das Node guarding Protokoll, Änderungen sind sofort wirksam. Details zu Modul-Überwachung s. Kap. 2.5.

2.3.2.10 Objekt 0x100D life time factor

Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED8
Default: 0
Zugriff: RW

Ein Wert <> 0 aktiviert das Node guarding Protokoll, Änderungen sind sofort wirksam. Details zu Modul-Überwachung s. Kap. 2.5.

2.3.2.11 Objekt 0x1010 Store Parameters

Schreibt die Parameter ins interne Datenflash des Hauptprozessors.

Folgende Objekte werden gesichert:

0x100B bzw.	
0x2000.01	Node ID
0x2000.02	Baud ID
0x100C	guard time
0x100D	life time factor
0x1015	EMCY inhibit time
0x1017	heartbeat time
0x1404.01 - 0x1407.01	COB-IDs der RPDOs 5-7
0x1804.01 - 0x1807.01	COB-IDs der TPDOs 5-7
0x2420	Axis Status Polltime
0x2431	Axis Communication timeout
0x3000.01-15:	Achskonfiguration
0x3101.01-15:	Minimal-Amplitude
0x5100	UMin0 – min. Drehzahl Stillstanderkenung
0x5101	AMin0 – Startamplitude für AmpMin-Bestimmung
0x5102	Motor-Rollen-Durchmesser

***Hinweise**:*

- *Der Hersteller spezifiziert ca. 10000 Lösch-/Schreib-Zyklen für das interne Datenflash*
- *Wenn die Objekte 0x100B und 0x2000.01 nicht übereinstimmen, erfolgt ein Abbruch mit SDO_ABORT_NOT_STORED = 0x08000020.*
- *Ein Abbruch mit SDO_ABORT_NOT_STORED = 0x08000020 erfolgt auch, wenn versucht wird, eine andere BaudID im Objekt 0x2000.02 als die z.Z. aktive BaudID abzuspeichern, s.a. Hinweise zum Objekt 0x2000 / CAN-OPEN Konfiguration*
- *Die Parametrierung der einzelnen Achsen wird im Datenflash des jeweiligen Achsprozessors gesichert; s. Objekt 0x3412.*

2.3.2.12 Objekt 0x1011 Restore Default Parameters

Liest die Default-Parameter aus dem internen Datenflash

2.3.2.13 Objekt 0x1014 EMCY_COBID

COB-ID des Emergency-Objekts

Objekttyp:	OT_VAR
Datentyp:	DT_UNSIGNED32
Default:	Default: 0x80 + Node-ID
Zugriff:	RO

2.3.2.14 Objekt 0x1015 EMCY_InhibitTime

Objekttyp:	OT_VAR
Datentyp:	DT_UNSIGNED16

Default: 0
Zugriff: RW

2.3.2.15 Objekt 0x1017 HeartBeat Time

Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED16
Default: 0
Zugriff: RW

Ein Wert <> 0 aktiviert das Heartbeat Protokoll, Änderungen sind sofort wirksam. Details zu Modul-Überwachung s. Kap. 2.5.

2.3.2.16 Objekt 1018h Identity Object

Implementiert sind die Subindizes 01-04

2.3.2.17 RPDO: Receive PDO COMMUNICATION PARAMETER

Die Objekte 0x1400-0x1408 definieren die Kommunikationsparameter der RPDOs 1-9; die COB-IDs für die RPDOs 1-4 entsprechen den in CAN-OPEN vorgesehenen Standardwerten, für die RPDOs 5-8 werden die Standardwerte der RPDOs 1-4 der NodeID+5 benutzt, das RPD9 ist ein globales RPDO für alle AC21-Knoten. Die COB-IDs sind nach diesem Schema vorgegeben und es ergeben sich daher Einschränkungen bei der Wahl der NodeIDs. Ab V. 0.92 können die COB-IDs der RPDO5-8 geändert und mit Hilfe des Objekts 0x1010 / STORE PARAMETERS im Datenflash dauerhaft gespeichert werden. Details s. Kap. 2.6.

Objekt 0x1400.00: Anzahl der Datenfelder
Objekttyp: OT_RECORD
Datentyp: DT_UNSIGNED8
Default: 2
Zugriff: RO

Objekt 0x1400.01: RPDO1_COB
Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED8
Default: 0x200 + Node-ID
Zugriff: RO

Objekt 0x1400.02: Transmission type
Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED8
Default: 255
Zugriff: CONST

Der Aufbau / die Werte der Objekte 0x1401..0x1408 sind analog und unterscheiden sich lediglich im Sub-Index 01 – COB-ID:

Objekt 0x1401.01: RPDO2_COB
Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED8
Default: 0x300 + Node-ID
Zugriff: RO

Objekt 0x1402.01: RPDO3_COB
Objekttyp: OT_VAR

Datentyp: DT_UNSIGNED8
Default: 0x400 + Node-ID
Zugriff: RO

Objekt 0x1403.01: RPDO4_COB
Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED8
Default: 0x500 + Node-ID
Zugriff: RO

Objekt 0x1404.01: RPDO5_COB
Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED8
Default: 0x205 + Node-ID
Zugriff: RO ab V.0.91 RW

Objekt 0x1405.01: RPDO6_COB
Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED8
Default: 0x305 + Node-ID
Zugriff: RO ab V.0.91 RW

Objekt 0x1406.01: RPDO7_COB
Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED8
Default: 0x405 + Node-ID
Zugriff: RO ab V.0.91 RW

Objekt 0x1407.01: RPDO8_COB
Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED8
Default: 0x505 + Node-ID
Zugriff: RO ab V.0.91 RW

Objekt 0x1408.01: RPDO9_COB
Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED8
Default: 0x300 (globales RPDO für alle AC21-Knoten, Details s. Kap.2.6)
Zugriff: RO

2.3.2.18 RPDO: Receive PDO MAPPING

Objekte 0x1600-0x1608
- nicht implementiert

2.3.2.19 TPDO: Transmit PDO COMMUNICATION PARAMETER

Die Objekte 0x1800-0x1807 definieren die Kommunikationsparameter der TPDOs 1-8; die COB-IDs für die TPDOs 1-4 entsprechen den in CAN-OPEN vorgesehenen Standardwerten, für die TPDOs 5-8 werden die Standardwerte der TPDOs 1-4 der NodeID+5 benutzt. Die COB-IDs sind nach diesem Schema vorgegeben und es ergeben sich daher Einschränkungen bei der Wahl der NodeIDs. Ab V. 0.92 können die COB-IDs der TPDO5-8 geändert und mit Hilfe des Objekts 0x1010 / STORE PARAMETERS im Datenflash dauerhaft gespeichert werden. Details s. Kap. 2.6.

Objekt 0x1800.00: Anzahl der Datenfelder
Objekttyp: OT_RECORD
Datentyp: DT_UNSIGNED8

Default: 2
Zugriff: RO

Objekt 0x1800.01: TPDO1_COB
Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED8
Default: 0x180 + Node-ID
Zugriff: RO

Objekt 0x1800.02: Transmission type
Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED8
Default: 254
Zugriff: CONST

Objekt 0x1800.03: inhibit time
Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED16
Default: 0
Zugriff: CONST

Objekt 0x1800.04: CMS priority group
Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED8
Default: 3
Zugriff: CONST

Objekt 0x1800.05: event time
Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED16
Default: 0
Zugriff: CONST

Der Aufbau / die Werte der Objekte 0x1801..0x1807 sind analog und unterscheiden sich lediglich im Sub-Index 01 – COB-ID:

Objekt 0x1801.01: TPDO2_COB
Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED8
Default: 0x280 + Node-ID
Zugriff: RO

Objekt 0x1802.01: TPDO3_COB
Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED8
Default: 0x380 + Node-ID
Zugriff: RO

Objekt 0x1803.01: TPDO4_COB
Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED8
Default: 0x480 + Node-ID
Zugriff: RO

Objekt 0x1804.01: TPDO5_COB
Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED8

Default: 0x185 + Node-ID
Zugriff: RO ab V.0.91 RW

Objekt 0x1805.01: TPDO6_COB
Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED8
Default: 0x285 + Node-ID
Zugriff: RO ab V.0.91 RW

Objekt 0x1806.01: TPDO7_COB
Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED8
Default: 0x385 + Node-ID
Zugriff: RO ab V.0.91 RW

Objekt 0x1807.01: TPDO8_COB
Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED8
Default: 0x485 + Node-ID
Zugriff: RO ab V.0.91 RW

2.3.2.20 TPDO: Transmit PDO MAPPING

Objekte 0x1A00-0x1A07
- nicht implementiert

2.3.3 Manufacturer Specific Profile Area - Allgemeine Objekte

2.3.3.1 Objekt 0x2000 CAN-OPEN Configuration

Die Objekte 0x2000.01 / Node-ID und (ab. V.0.93) 0x2000.02 / BaudID können mit Hilfe des Objekts 0x1010 / Store Parameters im internen Flash-ROM gespeichert werden und stehen damit nach einem Neustart direkt zur Verfügung.
Weitere Hinweis s.a. Objekt 0x1010 / STORE PARAMETERS

Objekt 0x2000.00: Anzahl der Datenfelder
Objekttyp: OT_RECORD
Datentyp: DT_UNSIGNED8
Default: 2
Zugriff: RO

Objekt 0x2000.01: NodeID
Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED8
Default: 126
Zugriff: RW

Im Gegensatz zum Objekt 0x100B definiert dieses Objekt die Node-ID, die mit dem Command service „RESET COMMUNICATION“ (neu) gesetzt wird, d.h. das Objekt 0x2000.01 wird ins Objekt 0x100B kopiert. Eine Änderung der NodeID (0x2000.01) ist deshalb erst nach einem NMT-Kommando „RESET COMMUNICATION“ wirksam.
Damit ist gleichzeitig ein einfacher Mechanismus zur Änderung der Node-ID implementiert.

***Hinweis**:* *Da für 21 Motore die 4 Standard-PDO-IDs zur Übertragung der Prozess-Information nicht ausreichen, werden weitere PDO-IDs benutzt, die eigentlich*

anderen NODE-IDs zugeordnet sind. Diese NODE-IDs dürfen daher nicht vergeben werden. Weitere Details dazu s. Kap. 2.6: PDO-Kommunikation.

Objekt 0x2000.02 BaudID
Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED8
Default: 0 = 1 MBaud
Zugriff: RO ab. V.0.93: RW
Wertebereich 0,2,3 = 1MBd, 500kBd, 250kBd

Ab Firmware-Version 0.93 kann die Baudrate geändert werden; der Mechanismus ist der gleich, wie beim ändern des Objekts 0x2000.01 / NodeID, d.h. eine Änderung ist erst nach einem NMT-Kommando "RESET COMMUNICATION" wirksam.

Dauerhaftes Ändern der NodeID und der BaudID:

- *Objekt 0x2000.01 / NodeID*
  und / oder
- *Objekt 0x2000.02 / BaudID*
  ändern

- *NMT-Kommando "RESET COMMUNICATION"*
  → Die Verbindung zum CAN-Knoten muß mit diesen (neuen) Parametern neu aufgebaut werden! Sollte dies nicht gelingen, so kann der CAN-Knoten problemlos durch einen Reset / Unterbrechung der Stromzufuhr wieder in den alten Zustand zurückgesetzt werden.

- *Schreiben des Objekts 0x1010 / STORE PARAMETERS: erst jetzt werden NodeID und BaudID dauerhaft gespeichert. Der Versuch, eine andere als die aktuell verwendete NodeID / BaudID zu speichern, wird mit SDO_ABORT_NOT_STORED = 0x08000020 abgebrochen.*

Dieser Mechanismus stellt sicher, daß eine Kommunikation mit den gewünschten Parametern möglich ist.

2.3.3.2 Objekt 0x2001 Node State

Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED8
Default: PREOPERATIONAL
Zugriff: RO

2.3.3.3 Objekt 0x200A Version Info

Das Objekt implementiert die Versionsinformationen:

Objekt 0x200A.00: Anzahl der Datenfelder
Objekttyp: OT_RECORD
Datentyp: DT_UNSIGNED8
Default: 3
Zugriff: RO

Objekt 0x200A.01: Software-Version
Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED32
Default: Versionsnummer der Firmware – BCD-codiert:
Bit31..24: Hauptversion
Bit23..16: Nebenversion
Bit15..0: Revision

Zugriff: RO

Beispiel: 0x00200000 → Versionsnummer 00.20.0000: Firmware-Version 0.20

Objekt 0x200A.02 Software-Datum
Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED32
Default: Erstellungsdatum – BCD-codiert:
Bit31..16: Jahr
Bit15..8: Monat
Bit7..0: Tag
Zugriff: RO

Objekt 0x200A.03 Hardware-Version
Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED32
Default: Bit31..8: Versionsnr.
Bit7..0: Anzahl der Achsen
Zugriff: RW

2.3.3.4 Objekt 0x2404 BackupSync

Dieses Objekt synchronisert das Backup-Flash mit dem Motor-Datenflash aller Motorprozessoren. Weitere Details s. Kap. 2.2.1.

Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED32
Default: 0
Zugriff: WO

***Hinweis**: Dies kann u.U. 1 min und länger dauern!*

2.3.3.5 Objekt 0x2420 Axis Status Polltime

Dieses Objekt definiert das interne Abfrage-Intervall für die Statusinformation aus den Motorprozessoren. Die Angabe erfolgt in ms.

Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED16
Default: 1000
Zugriff: RW

2.3.3.6 Objekt 0x2431 Axis Communication Timeout

Dieses Objekt definiert den Timeout für die interne Kommunikation mit den einzelnen Motorprozessoren. Die Angabe erfolgt in ms.

Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED16
Default: 200
Zugriff: RW

2.3.4 0x3000-0x3FFF: Motorspezifische Objekte – Arrays

Alle Objekte sind vom Typ OT_ARRAY; die Subindizes indizieren die einzelnen Motoren.

***Hinweis**:* *In V.0.20 arbeiten die Befehle 0x3013..0x301F fehlerhaft und dürfen(!) deshalb nicht verwendet werden!*

2.3.4.1 Objekt 0x3000 Konfiguration

Datentyp: DT_UNSIGNED32
Default: 0
Zugriff: RW

Bit 0..4: Motor / Antrieb - Mapping
Lowbyte (0x3000.mm) = aa:
Befehle für den Motor Nr. „mm“ werden am Antrieb „aa“ ausgeführt;
Sonderfälle:
„aa“ = 0: es erfolgt kein Mapping, d.h. es gilt „aa“ = „mm“.
„aa“ = 31: der Motor ist abgeschaltet / nicht vorhanden

Bit 15: Drehrichtung
Bit 15 = 1: Die Drehrichtung des Motors wird invertiert

Änderungen an diesem Objekt sollten nur im Zustand PREOPERATIONAL vorgenommen werden; die Konfiguration wird beim Zustandswechsel von PREOPERATIONAL → OPERATIONAL aktiviert.

Das Objekt kann mit Hilfe des Objekts 0x1010 / Store Parameters im Datenflash des Hauptprozessors gespeichert werden und steht damit nach einem Neustart direkt zur Verfügung.

2.3.4.2 Objekt 0x3001 Version

Datentyp: DT_UNSIGNED16
Default: 0
Zugriff: RW

Firmware-Versionsnummer der einzelnen Motorelektroniken.

2.3.4.3 Objekt 0x3011 Rampe

Datentyp: DT_UNSIGNED16
Default: 100
Zugriff: RW

Hochlaufzeit des Motors von 0→UMax in ms

Das Objekt kann mit Hilfe des Objekts 0x3412 / Config Parameters im Datenflash des jeweiligen Motorprozessors gespeichert werden und steht damit nach einem Neustart direkt zur Verfügung.

2.3.4.4 Objekt 0x3014 UMax

Datentyp: DT_UNSIGNED16
Default: 500
Zugriff: RW

Maximaldrehzahl des Motors in U/min

Das Objekt kann mit Hilfe des Objekts 0x3412 / Config Parameters im Datenflash des jeweiligen Motorprozessors gespeichert werden und steht damit nach einem Neustart direkt zur Verfügung.

2.3.4.5 Objekt 0x3015 PFac

Datentyp: DT_SIGNED16
Default: 20
Zugriff: RW

Proportional-Faktor für den Lageregelkreis.

Das Objekt kann mit Hilfe des Objekts 0x3412 / Config Parameters im Datenflash des jeweiligen Motorprozessors gespeichert werden und steht damit nach einem Neustart direkt zur Verfügung.

2.3.4.6 Objekt 0x3018 VFac

Datentyp: DT_SIGNED16
Default: 25
Zugriff: RW

Vorsteuer-Faktor für den Lageregelkreis.

Das Objekt kann mit Hilfe des Objekts 0x3412 / Config Parameters im Datenflash des jeweiligen Motorprozessors gespeichert werden und steht damit nach einem Neustart direkt zur Verfügung.

2.3.4.7 Objekt 0x3019 PDFac

Datentyp: DT_UNSIGNED16
Default: 40
Zugriff: RW

Proportional-Faktor für die Geschwindigkeitsregelung.

Das Objekt kann mit Hilfe des Objekts 0x3412 / Config Parameters im Datenflash des jeweiligen Motorprozessors gespeichert werden und steht damit nach einem Neustart direkt zur Verfügung.

2.3.4.8 Objekt 0x301C Axis Guard time

Datentyp: DT_UNSIGNED16
Default: 0
Zugriff: RW

Überwachungszeit in ms, Default = 0 = keine Überwachung.
Kommunikationsüberwachung CAN-Prozessor ←→ Motorprozessor; der Motorprozessor schaltet die Freigabe weg, wenn während der hier eingestellten Zeit keine Kommunikation auf dem internen Bus erkannt wird und setzt den Fehlerstatus 0x38. Im Objekt 0x4000 ist dieser Status natürlich erst dann sichtbar, wenn die Kommunikation CAN-Prozessor ←→ Motorprozessor wieder aktiv ist...!

Das Objekt kann mit Hilfe des Objekts 0x3412 / Config Parameters im Datenflash des jeweiligen Motorprozessors gespeichert werden und steht damit nach einem Neustart direkt zur Verfügung.

***Hinweis**:* *Da die Kommunikation in den Motorprozessoren überwacht wird, wird auch die Überwachungszeit im Motorprozessor verwaltet bzw. dort auch im jeweiligen Datenflash gesichert. Dies hat zur Folge, daß für jeden Motorprozessor die Überwachungszeit getrennt programmiert werden muß; insb. ist es möglich, für jeden Motorprozessor eine eigene Zeit zu definieren, was i.a. aber nicht sinnvoll sein dürfte.*

2.3.4.9 Objekt 0x301D INenn

Datentyp: DT_UNSIGNED16
Default: 500
Zugriff: RO

Nennstrom des Motors in mA.

2.3.4.10 Objekt 0x3027 AmpLimit

Datentyp: DT_UNSIGNED16
Default: 100
Zugriff: RW

Amplitudenbegrenzung in % der Maximalamplitude. Die Begrenzung wirkt nur bei der Betriebsart OperationStartAmp.

Das Objekt kann mit Hilfe des Objekts 0x3412 / Config Parameters im Datenflash des jeweiligen Motorprozessors gespeichert werden und steht damit nach einem Neustart direkt zur Verfügung.

***Hinweis**: Dieses Objekt ist ab Version 0.63 verfügbar*

2.3.4.11 Objekt 0x3101 AmpMin

Datentyp: DT_INTEGER16
Default: 10
Zugriff: RW

Das Objekt verwaltet die jeweilige Amplitude (in % der Maximal-Amplitude) eines jeden Motors, bei der der Motor grade zu drehen beginnt. Das Objekt wird bei der Funktion „Beladungs-Test“ verwendet und kann auch automatisch bestimmt werden.
Das Objekt wird im CAN-Prozessor verwaltet und kann (deshalb) mit Hilfe des Objekts 0x1010 / Store Parameters im Datenflash des CAN-Prozessors gespeichert werden und steht damit nach einem Neustart direkt zur Verfügung.

***Hinweis**: Dieses Objekt ist ab Version 0.66 verfügbar*

2.3.4.12 Objekt 0x3412 Config Parameter

Datentyp: DT_UNSIGNED32
Default:
Zugriff: WO

Das Objekt verwaltet die Konfiguration der Betriebsparameter der einzelnen Motorprozessoren; das übergebene Langwort bestimmt die auszuführende Aktion:

0x65766173 = "save": alle Konfigurationsparameter werden ins Datenflash des Motorprozessors geschrieben.
Gesichert werden folgende Objekte:

0x3011	Rampe
0x3014	UMax
0x3015	Pfax
0x3018	VFac
0x3019	PDFac
0x301C	Guard time
0x3027	AmpLimit

0x64616F6C = "load": alle Konfigurationsparameter werden auf ihre Default-Werte gesetzt

0x74696E69 = "init": Motor-Initialisierung:
- alle Konfigurationsparameter werden auf ihre Default-Werte gesetzt
- der Motor-Initialisierungslauf wird ausgeführt
- Konfigurationsparameter und Initialisierungsdaten werden ins Datenflash des Motorprozessors geschrieben

0x646C6572 = "reld": alle Konfigurationsparameter werden aus dem Datenflash des Motorprozessors gelesen

ab V.0.30:
0x32696E69 = "ini2": es wird nur der Motor-Initialisierungslauf ausgeführt

Hinweise
- *Der Hersteller spezifiziert ca. 1000 Lösch-/Schreib-Zyklen für das interne Daten-Flash.*
- *Allgemeine, nicht motorspezifische Parameter werden im Datenflash des Hauptprozessors gesichert; s. Objekt 0x1010*
- ***Ab V.0.62** wird ein Backup der Motor-Daten-Flashs im CAN-Prozessor verwaltet. Die Synchronisierung erfolgt automatisch bei Programmstart in der Bootup-Phase. Es empfiehlt sich jedoch nachdem die Konfigurationsparameter **aller** Motor-Prozessoren in das Motor-Datenflash geschrieben wurden, explizit eine Synchronisation mit dem Backup-Flash zu veranlassen (s. Objekt 0x2404 – BackupSync)*

2.3.5 0x4000-0x4DFF: Motorspezifische Objekte

Im Bereich 0x4000-0x407F befinden sich Objekte für den Motor 1. Analog befinden sich im Bereich 0x4080-0x4A7F jeweils mit Offset 0x80 die Objekte für die weiteren Motoren.

2.3.5.1 Objekt 0x4000

Objekt 0x4000.00 Anzahl der Datenfelder

Objekttyp: OT_RECORD
Datentyp: DT_UNSIGNED8
Default: 7
Zugriff: RO

Objekt 0x4000.01 Controlwort

Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED16
Default: 0
Zugriff: RO

Objekt 0x4000.02 Statuswort

Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED16
Default: 0
Zugriff: RO

Objekt 0x4000.03 Soll-Drehzahl

Objekttyp: OT_VAR
Datentyp: DT_SIGNED16
Default: 500
Zugriff: RW

Solldrehzahl in U/min für die Betriebsart „Geschwindigkeitsregelung“

Objekt 0x4000.04 Ist-Drehzahl

Objekttyp: OT_VAR
Datentyp: DT_SIGNED16
Default: 0
Zugriff: RO

Aktuelle Istdrehzahl in U/min

Objekt 0x4000.05 Soll-Strom

Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED16
Default: 0
Zugriff: RW

Soll-Strom in % des Nennstroms für die Betriebsart „Stop mit Haltemoment“; der Nennstrom ist fest mit 500 mA vorgegeben.

Objekt 0x4000.06 Soll-Amplitude

Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED16
Default: 0
Zugriff: RW

Soll-Amplitude in % der Maximal-Amplitude für die Betriebsart „DC-Betrieb“; die Maximal-Amplitude der PWM ist die Amplitude, bei der im Stillstand des Motors der Nennstrom erreicht wird.

Objekt 0x4000.07 Distanz

Objekttyp: OT_VAR
Datentyp: DT_SIGNED16
Default: 0
Zugriff: RW

Ab Firmware-Version V. 0.90 ist eine Positionierfunktion verfügbar; mit diesem Objekt wird die Distanz zum Ziel festgelegt bzw. der noch zu fahrende Restweg gelesen. Die Angabe erfolgt in mm; intern wird zur Umrechnung der Rollen-Durchmesser der Achse benötigt - s. Objekt 0x5102.00 / Rollen-Durchmesser. Die maximal zulässige Distanz ist durch die interne Verarbeitung eingeschränkt auf

max. Distanz[mm] = 114,89 * Rollendurchmesser[mm]

Ab V.0.93: Ist kein Rollendurchmesser gesetzt (Objekt 0x5102.00 = 0), so wird die Distanzangabe nicht als mm sondern als grd [°] interpretiert.
Die maximale Distanz beträgt dann ca. 13200° oder ca. 36,5 Motor-Umdrehungen.

Als Distanz sind ebenso wie für die Soll-Geschwindigkeit positive und negative Werte zugelassen; die effektive Fahr-Richtung ergibt sich aus der Kombination beider Vorzeichen!

NB: *...wird für sowohl für die Solldrehzahl als auch für die Distanz ein negativer Wert programmiert, so ergibt sich eine positive Fahrbewegung!*

2.3.6 Gemeinsame Objekte

In diesem Indexbereich liegen Objekte die für alle Motoren gleichermaßen gelten bzw. alle Motoren gleichermaßen beeinflussen.

2.3.6.1 Objekt 0x5000

Objekt 0x5000.00 Anzahl der Datenfelder
Objekttyp: OT_RECORD
Datentyp: DT_UNSIGNED8
Default: 1
Zugriff: RO

Objekt 0x5000.01 Controlword
Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED16
Zugriff: WO

Über dieses Objekt können alle Control-Worte (Index = 0x4000 + Achsindex*0x80, Subindex = 01) gleichzeitig gesetzt werden.

2.3.6.2 Objekt 0x5060 modes of operation

Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED8
Default: 2 (velocity mode)
Zugriff: RO

2.3.6.3 Objekt 0x5061 modes of operation display

Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED8
Default: 2 (velocity mode)
Zugriff: RO

2.3.6.4 Objekt 0x5100 UMin0 – Stillstandsdrehzahl

Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED16
Default: 30
Zugriff: RW

Dieses Objekt definiert die min. Drehzahl für den Beladungstest; dreht der Motor unterhalb dieser Drehzahl, so wird er als „stillstehend“ bewertet.
Das Objekt kann mit Hilfe des Objekts 0x1010 / Store Parameters im internen Flash-ROM gespeichert werden und steht damit nach einem Neustart direkt zur Verfügung.

***Hinweis**: Dieses Objekt ist ab Version 0.76 verfügbar*

2.3.6.5 Objekt 0x5101 AMin0 – Startamplitude

Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED16
Default: 5
Zugriff: RW

Dieses Objekt definiert die Start-Amplitude (in % der Maximal-Amplitude) für die automatische Bestimmung der Minimal-Amplitude (s. Beladungstest).
Das Objekt kann mit Hilfe des Objekts 0x1010 / Store Parameters im internen Flash-ROM gespeichert werden und steht damit nach einem Neustart direkt zur Verfügung.

***Hinweis**:* *Dieses Objekt ist ab Version 0.76 verfügbar*

2.3.6.6 Objekt 0x5102 Rollen-Durchmesser

Objekttyp: OT_VAR
Datentyp: DT_UNSIGNED16
Default: 50
Zugriff: RW

Dieses Objekt definiert den Durchmesser der Motor-Rollen und wird für die Positionierfunktion benötigt s.a. Objekt 0x4000.07 / Distanz. Z.Zt. ist nur ein identischer Durchmesser für alle Motor-Rollen implementiert.
Das Objekt kann mit Hilfe des Objekts 0x1010 / Store Parameters im internen Flash-ROM gespeichert werden und steht damit nach einem Neustart direkt zur Verfügung.

***Hinweis**:* *Dieses Objekt ist ab Version 0.90 verfügbar*

2.4 Control- und Status-Wort

Control- und Statuswort implementieren die in DSP-402 vorgesehen Statusmaschine. Der Aufbau weicht jedoch von der DSP-402 ab: Anstatt die einzelnen Zustände bzw. Befehle in einzelnen Bits zu codieren, wurden entsprechende Konstanten definiert. Zusätzlich kann im Zustand „SwitchedOn“ die Betriebsart des Motors gesetzt werden; je nach gewählter Betriebsart befindet sich der Motor dann im Zustand „OperationEnabled_<Betriebsart>“.

Im übrigen folgt die Implementierung der in DSP-402 beschriebenen Status-Maschine:

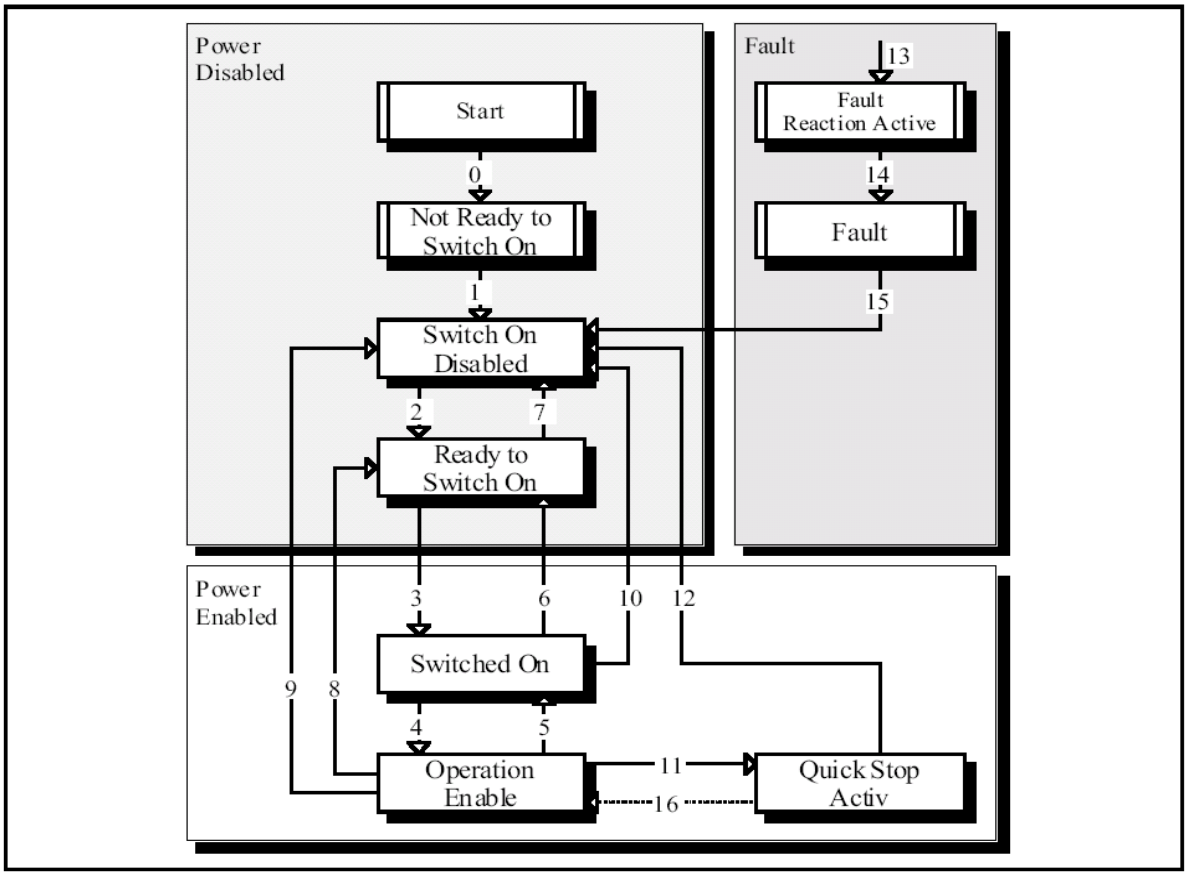


Abbildung 5: DSP-402-Status-Maschine

2.4.1 Statuswort

Der Zustand wird in den 5 Lowbits des Status codiert; die Bits 8..15 beinhalten ggf. einen Fehlercode, die übrigen Bits sind unbelegt.
Der Zustand „OperationEnabled“ wird durch mehrere Statuswerte (0x10..0x14) repräsentiert: er beinhaltet zugleich Informationen über die angewählte Betriebsart (s.u.).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Fehlercode											Status				

2.4.1.1 Status – Bits 0..4

NotReadyToSwitchOn	0x18
SwitchOnDisabled	0x19
ReadyToSwitchOn	0x1A
SwitchedOn	0x1B
(QuickStopActive	0x1D
	nicht implementiert)
FaultReactionActive	0x1E
Fault	0x1F

Zustand „OperationEnabled“, unterteilt nach gewählter Betriebsart:

OperationEnabled_Stopped	0x10	Geschwindigkeitsregelung aus, Achse bestromt!
OperationEnabled_Running	0x11	Geschwindigkeitsregelung
OperationEnabled_Halt	0x12	Stop mit Haltemoment
OperationEnabled_DC	0x13	DC-Betrieb
OperationEnabled_Amp	0x14	Geschwindigkeitsregelung mit Amplitudenbegrenzung entspr. Objekt 0x3027 – AmpLimit
OperationEnabled_Move	0x15	Positionierbetrieb

2.4.1.2 Fehlercodes – Bits 8..13

Überstrom	0x30	
Phasenfehler	0x31-0x37:	Bit0..2 = 1 → Phase1/2/3 fehlt / Phasen-Messimpuls nicht messbar, verursacht z.B. durch Kabelbruch
***Hinweis**:*	*0x37 (alle Phasen fehlen) deutet auf einen nicht angeschlossenen Motor.*	
Kommunikation-Timeout	0x38	Kommunikationsausfall CAN-Prozessor ←→ Motor-Prozessor: Axis Guard Time (Objekt 0x301C) überschritten
Hinweis:	*dieser Fehler ist im Status natürlich erst dann sichtbar, wenn die Kommunikation wieder aktiv ist...!*	
Kommutierungsfehler	0x39	Der Motor-Contoller konnte den nächsten Umschaltpunkt für die Phasen-Kommutierung nicht finden.
Flashfehler	0x3D	Parameter im Datenflash sind ungültig
Flashfehler	0x3E	Parameter konnten nicht ins Datenflash geschrieben werden
Initialisierungsfehler	0x3F	Der Motor-Initialisierungslauf konnte nicht fehlerfrei durchgeführt werden

2.4.2 Controlwort

Steuerdaten werden in den 5 Lowbits codiert; von den übrigen Bits werden die 3 Highbits (Bit 13-15) für interne Zwecke benutzt, die übrigen Bits (Bit 5-12) sind unbenutzt. Wird das Controlwort geschrieben, müssen Bit 5-15 = 0 sein.
Eine Betriebsart kann im Controlwort gesetzt werden, wenn sich die Achse im Status „SwitchedOn“ befindet.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Intern											Control				

2.4.2.1 Steuerworte

ShutDown	0x18	
SwitchOn	0x19	
DisableVoltage	0x1A	
(QuickStop	0x1B	nicht implementiert)
DisableOperation	0x1C	
EnableOperation	0x1D	
FaultReset	0x1F	

2.4.2.2 Betriebsart

OperationStop	0x10	Geschwindigkeitsregelung aus
OperationStart	0x11	Geschwindigkeitsregelung ein
OperationHalt	0x12	Stop mit Haltemoment
OperationDC	0x13	DC-Betrieb
OperationStartAmp	0x14	Geschwindigkeitsregelung ein, Amplitudenbegrenzung entspr. Objekt 0x3027 – AmpLimit
OperationMove	0x15	Positionierbetrieb: rel. Positionierung entspr. Distanz im Objekt 0x4000+(n-1)*0x80.07, n = Achsnr.

2.5 Modul-Überwachung

Das Modul unterstützt sowohl das Node Guarding- als auch das Heartbeat-Protokoll.

***Hinweis**:* *Nach CAN-Open-Spezifikation DS-301 soll nur eines der beiden Protokolle verwendet werden.*

Außerdem besitzt das Modul 2 Anzeige-LEDs. Die Funktion der beiden LEDs entspricht der CiA-Spezifikation DR-303-3.

2.5.1 Heartbeat-Protokoll

Diese Protokoll ist automatisch aktiv, wenn das Objekt 0x1017 (Heartbeat Time) einen Wert <> 0 enthält. Im angegebenen Zeitintervall wird eine Heartbeat-Message versandt, die das Vorhandensein des Knotens signalisiert.

Weitere Details s. CiA-Dokument DS-301.

2.5.2 Node Guarding Protokoll

Das Protokoll ist aktiv, wenn die beiden Objekte 0x100C (Guard Time) und 0x100D (Lifetime Factor) einen Wert <> 0 enthalten. In diesem Fall überwacht das Modul den regelmäßigen Empfang der ERRCTRL-Message vom Master. Ist die durch die beiden Objekte definierte Lifetime abgelaufen, ohne daß eine ERRCTRL-Message empfangen wurde, so schaltet das Modul in den Zustand „PREOPERATIONAL“, d.h. alle Motorbewegungen werden gestoppt und die Motoren stromlos geschaltet. Dieser Zustand wird auch durch die rote LED signalisiert (s.u.).

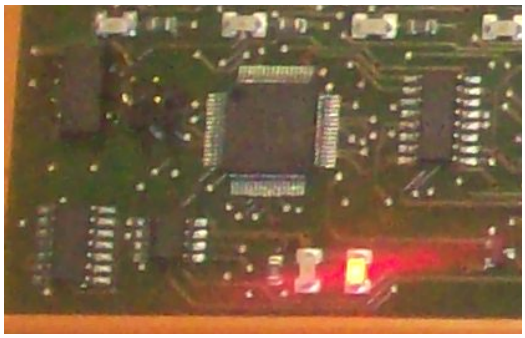
Unabhängig davon, ob das Node guarding Protokoll aktiv ist (Guard Time, Lifetime Factor <> 0), antwortet das Modul auf eine empfangene ERRCTRL-Message vom Master mit einer Heartbeat-Message.

Weitere Details s. CiA-Dokument DS-301.

2.5.2.1 Erweitertes Node Guarding

Zusätzlich zu dem in CAN-Open definierten Verhalten wertet das Modul nicht nur die ERRCTRL-Message aus, sondern auch den Empfang von Receive- und Transmit-PDOs. Der Empfang dieser Messages hat fast dieselbe Wirkung wie der Empfang einer ERRCTRL-Message; es wird jedoch keine Heartbeat-Message an den Master versandt.

2.5.3 LEDs



Die Funktion der beiden LEDs entspricht der CiA-Spezifikation DR-303-3; folgende Zustände werden signalisiert:

2.5.3.1 Grüne LED: Node state

(rechts)

Node state = STOPPED:	Single Flash
Node state = PREOPERATIONAL:	Blinken
Node state = OPERATIONAL:	LED Ein

2.5.3.2 Rote LED: CAN-Bus-Status

(links)

CAN Working:	LED aus
CAN-Bus Warning level:	Single Flash
CAN-Bus off:	LED ein
ERRCTRL-Event:	Double Flash

2.5.3.3 LEDs der einzelnen Motor-Controller

Jeder der Motor-Controller steuert eine eigene LED an:

Freigabe aus:	LED aus
Freigabe an / Motor bestromt:	LED an
Fehler:	LED blinkt

2.6 PDO-Kommunikation

2.6.1 Allgemeines

Unter CAN-OPEN stehen standardmäßig für jedes CAN-Gerät je 4 Identifier für Transmit- und Receive-PDO-Objekte zur Verfügung. Um Prozessinformation á 16 Bit für mehr als 16 Motoren zu übertragen, reichen diese jedoch nicht aus.
Deshalb werden nach einem festgelegten Schema TPDO- und RPDO-Identifier reserviert, die eigentlich CAN-Geräten mit anderen CAN-IDs zugeordnet sind. Diese CAN-IDs dürfen daher NICHT vergeben werden bzw. Knoten mit diesen IDs dürfen am CAN-Bus nicht vorhanden sein!

2.6.2 Firmware-Version bis 0.20

Es werden je ein TPDO- und RPDO-Identifier reserviert, die eigentlich CAN-Geräten mit CAN-ID > 100 zugeordnet sind. Damit stehen jedem Gerät je 5 TPDO- und RPDO-Identifier zur Verfügung; zu beachten ist, daß CAN-IDs > 100 aus diesem Grund nicht vergeben werden dürfen!

Die zusätzlichen PDO-IDs (PDO 5) werden nach folgendem Schema bestimmt:
Für die Geräte mit den CAN-IDs 1 bis 4 werden die PDOs 1-4 des Geräts mit CAN-ID 101 reserviert; entsprechendes gilt für jede weitere 4er-Gruppe.

***Hinweis**:* *Die Datei PDOs.XLS enthält ein Utility zur einfachen Bestimmung der jeweiligen PDO-IDs.*

2.6.3 Firmware-Version ab 0.30

Für jedes Modul mit Node-ID N werden zusätzlich die 4 Standard-TPDO/RPDO-Identifier der Node-ID N+5 reserviert, also für NodeID 1 zusätzlich die PDO-Identifier der Node-ID 6 usw. Dies bedeutet, daß jeweils die Node-IDs mit den Endziffern 6, 7, 8, 9 und 0 nicht vergeben werden dürfen!

***Hinweis**:* *Von den zusätzlichen PDOs (Standard-PDOs der NodeID+5) sind nur die beiden ersten PDOs tatsächlich implementiert.*

Die Node-IDs 121 bis 127 sind reserviert und dienen speziellen Zwecken:

Node-ID 121-125	vordefinierte IDs; PDO5-8 stehen nicht zur Verfügung
Node-ID 126:	Default-ID (s.o.)
Node-ID 127:	Die PDOs dieser NodeID werden als PDO5-8 für die Node-ID 126 definiert, d.h. diese ID darf nicht vergeben werden

2.6.4 Firmware-Version ab 0.70

Ab Version 0.70 sind alle 4 zusätzlichen PDOs (PDO5-PDO8 = PDO1-PDO4 der NodeID+5) implementiert. Tatsächlich ausgewertet bzw. versandt werden bisher aber nur RPDO7 / TPDO7.
Außerdem wird als 9. RPDO das RPDO1 der Node-ID 0 als Global-RPDO erkannt, d.h. über dieses RPDO können alle angeschlossenen Module gleichzeitig angesprochen werden.

***Hinweis**:* *RPDO9 ist in allen Modulen identisch (1. RPDO-ID / NodeID 0)!*

2.6.5 Aufbau der PDOs 1-6

Jedes Prozessdaten-Objekt (PDO) besteht aus 4x16Bit für 4 Motoren; im 6. PDO sind jedoch nur 1x16 Bit belegt.

Belegung der 8 Datenbyte des PDO – jeweils 16 Bit bilden ein Datenwort für einen Motor:

	Bit 63-48		Bit 47-32		Bit 31-16		Bit 15-0	
PDO1:	Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
	Motor 4		Motor 3		Motor 2		Motor 1	

	Bit 63-48		Bit 47-32		Bit 31-16		Bit 15-0	
PDO1:	Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
	Motor 8		Motor 7		Motor 6		Motor 5	

	Bit 63-48		Bit 47-32		Bit 31-16		Bit 15-0	
PDO3:	Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
	Motor 12		Motor 11		Motor 10		Motor 9	

	Bit 63-48		Bit 47-32		Bit 31-16		Bit 15-0	
PDO4:	Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
	Motor 16		Motor 15		Motor 14		Motor 13	

	Bit 63-48		Bit 47-32		Bit 31-16		Bit 15-0	
PDO5:	Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
	(Motor 20)		(Motor 19)		Motor 18		Motor 17	

	Bit 63-48		Bit 47-32		Bit 31-16		Bit 15-0	
PDO6:	Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
	unbelegt		unbelegt		unbelegt		(Motor 21)	

***Hinweis**:* *Die Angaben in Klammern gelten ab V.0.30; bis V.0.20 sind die entsprechenden Felder unbelegt. PDO6 ist erst ab V.0.30 implementiert.*

2.6.5.1 RPDO

Jedes Wort repräsentiert die Steuerdaten für einen Motor; in den 5 Highbits wird eines der Steuerworte für die gewünschte Betriebsart (s. Kap. 2.4.2.2) übergeben, die 11 Lowbits enthalten die gewünschte Geschwindigkeit (im 2er-Komplement: -1024..+1023 U/min; Betriebsarten = OperationStart und OperationMove), den Sollstrom in % des Nennstroms (Betriebsart = OperationHalt) oder die Sollamplitude in % der Maximal-Apmplitude (Betriebsart = OperationDC). Die übergebenen Daten werden ins Controlwort und in die Sollgeschwindigkeit, den Sollstrom oder die Sollamplitude übernommen.

Datenwort für einen Motor, 16 Bit:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Steuerwort					Solldrehzahl in U/min bzw. Sollstrom / Sollamplitude in %										

***Hinweise**:* *Wird im kompletten Wort der Wert 0x0000 übertragen, so wird dieses Datenwort nicht ausgewertet; damit ist es möglich, Information nur für einzelne Motoren per PDO zu übertragen.*
Die Distanz für die Betriebsart "OperationMove" kann nicht per RPDO sondern nur per SDO übergeben werden.

2.6.5.2 TPDO

Jedes Wort repräsentiert den Prozess-Zustand eines Motors; die 5 Highbits enthalten den Status aus den 5 Lowbits des Statusworts (s. Kap. 2.4.1.1), die 11 Lowbits – unabhängig von der gewählten Betriebsart – die Istdrehzahl (im 2er-Komplement: -1024..+1023 U/min).

Datenwort eines Motors, 16 Bit:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Statuswort					Istdrehzahl in U/min										

2.6.6 PDO7 / RPDO9

Die RPDO7 bzw. das RPDO9 werden benutzt, um alle Motoren eines Moduls gleichzeitig zu bedienen und kurze Bewegungssequenzen zu realisieren. Das Ergebnis der Sequenz melden die Module dann jeweils mit dem TPDO7.
Die Funktionalität wird insbesondere benutzt, um die Beladung eines Feldes zu erkennen. Dazu wird mittels RPDO7 oder RPDO9 eine Bewegungssequenz gestartet. Am Ende der Bewegung liefert das TPDO7 einen Bit-Vektor, der die Motoren kennzeichnet, die sich bewegt haben – genauer: deren Drehzahl während der Bewegungssequenz > UMin0 (SDO 0x5100) war.
Die hier beschrieben Funktionalität wird für den Beladungstest – s. folgendes Kap. – verwendet.

2.6.6.1 RPDO7 / RPDO9

Aufbau RPDO7:

RPDO7:	Bit 63-48		Bit 47-32		Bit 31-16		Bit 15-0	
	Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
	Intervall				Modus		Steuerwort / Sollwert	

Steuerwort/Sollwert

Im Lowwort (Byte0/1) werden die Steuerdaten für die Motoren übergeben; die 5 Highbits enthalten eines der Steuerworte für die gewünschte Betriebsart (s. Kap. 2.4.2.2), die 11 Lowbits enthalten die gewünschte Geschwindigkeit (im 2er-Komplement: -1024..+1023 U/min; Betriebsart = OperationStart), den Sollstrom in % des Nennstroms (Betriebsart = OperationHalt) oder die Sollamplitude in % der Maximal-Apmplitude (Betriebsart = OperationDC). Die übergebenen Daten werden ins Controlwort und in die Sollgeschwindigkeit, den Sollstrom oder die Sollamplitude ALLER Motoren übernommen. Die Drehrichtung der Motoren wird im **Modus** festgelegt.

Modus

Der Modus bestimmt die gewünschte Bewegungssequenz. Folgende Sequenzen stehen zur Verfügung:

0x0001 alle Motoren in der gewählten Drehrichtung

0x0002 spaltenweise in alternierender Drehrichtung, 1. Spalte eines jeden Moduls in der gewählten Drehrichtung.

0x0003 wie 0x0002, zusätzlich wird aus der NodeID die Drehrichtung des jeweils 1. Motors so bestimmt, daß sich über das gesamte Motorfeld eine spaltenweise alternierende Drehrichtung ergibt.

0x0004 zeilenweise in alternierender Drehrichtung, 1. Zeile eines jeden Moduls in der gewählten Drehrichtung.

0x0005 wie 0x0004, zusätzlich wird aus der NodeID die Drehrichtung des jeweils 1. Motors so bestimmt, daß sich über das gesamte Motorfeld eine zeilenweise alternierende Drehrichtung ergibt.

0x0006 alternierende Drehrichtung, d.h. die Motoren drehen abwechselnd rechts bzw. links; der 1. Motor eines jeden Moduls dreht in der gewählten Drehrichtung.

0x0007 wie 0x0006; der 1. Motor des 1. Moduls dreht in der gewählten Drehrichtung, zusätzlich wird aus der NodeID die Drehrichtung des 1. Motors der anderen Module so bestimmt, daß sich über das gesamte Motorfeld ein schachbrettartiges Drehmuster der Motoren ergibt.

0x0011..0x0017: wie 0x0001..0x0007, es wird jedoch ein Eichzyklus zur Bestimmung des Parameters „AmpMin“ (SDO-ARRAY 0x3101) durchgeführt; als Betriebsart ist nur „OperationDC“ zugelassen. Weitere Details s.u. (Kap.2.6.7.1 / Eichfunktion)

Intervall

Hier wird die Drehdauer der Motoren in ms angegeben; nach Ablauf dieser Zeit werden die Motoren in die Betriebsart „OperationStop“ geschaltet. Zusätzlich wird am Ende dieses Zeit-Intervalls festgestellt, welche der Motoren sich bewegen (schneller als UMin0 drehen). Am Ende der Bewegung wird das TPDO7 versandt – s. nächstes Kapitel.

Beispiel:

CANID = 0x200 (512):	RPDO1 der NodeID 0 = RPDO9 aller anderen Knoten
8 Datenbyte (Lowbyte first):	0x0A, 0x98, 0x07, 0x00, 0xDC, 0x05, 0x00, 0x00
→ Steuerwort/Sollwert:	0x980A = OperationDC, Sollamplitude 10%
→ Modus:	0x0007 = Drehrichtung alternierend / schachbrettartig
→ Intervall:	0x000005DC = Drehdauer: 1500 ms

Dieses RPDO (RPDO9!) startet alle Motoren auf allen Modulen mit 10% Amplitude in der Betriebsart „OperationDC“, Drehrichtung der Motoren abwechselnd rechts/links. Nach ca. 1500 ms wird die Drehzahl der Motoren festgestellt und die Motoren gestoppt (Betriebsart „OperationStop“).

2.6.6.2 TPDO7

Aufbau TPDO7:

	Bit 63-48		Bit 47-32		Bit 31-16		Bit 15-0	
TPDO7:	Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
	Bewegungsvektor				Modus		Steuerwort / Sollwert	

Am Ende der mit RPDO7 oder RPDO9 gestarteten Bewegungssequenz wird erfasst, welche der Motoren sich gedreht haben (Drehzahl > UMin0). Für jeden Motor, der sich dreht, wird im Bewegungsvektor das entsprechende Bit gesetzt (Motor1 = Bit0...Motor21=Bit20). Steuerwort/Sollwert bzw. Modus werden aus dem RPDO7 bzw. RPDO9 übernommen.
Das TPDO7 wird automatisch am Ende der Bewegungssequenz versandt!

Beispiel:

CAN_ID = 0x3AE (942):	TPDO7, NodeID = 41
Datenbytes (Lowbyte first):	0x00, 0x98, 0x07, 0x00, 0xFF, 0xCF, 0x16, x00
→ Steuerwort/Sollwert:	0x980A = OperationDC, Sollamplitude 10%
→ Modus:	0x0007 = Drehrichtung alternierend / schachbrettartig
→ Bewegungsvektor:	0x0016CFFF: Bit 12,13,16,19 = 0 → Motoren 13,14,17,20 belegt

Dieses TPDO wurde von Modul 41 am Ende der Bewegungssequenz versandt; die Motoren wurden mit 10% im DC-Betrieb mit schachbrettartig alternierender Drehrichtung angesteuert. Die Drehzahl der Motoren 13,14,17 und 20 war < UMin0 (SDO 0x5100)

2.6.7 Beladungstest / Eichfunktion

Für einen Beladungstest sollte die Betriebsart „OperationDC“ benutzt werden; die Motoren drehen dann ungeregelt mit der gewünschten Amplitude.
Beim Beladungstest wird versucht festzustellen, welche der Motoren durch Gegenstände belegt sind. Dazu werden die Motoren mit möglichst geringer Amplitude gedreht; belegte Motoren bleiben dabei im Stillstand. Da aber jeder Motor u.U. eine andere minimale Amplitude benötigt, um unbelastet

überhaupt zu drehen, besteht die Möglichkeit, eine Minimalamplitude für jeden Motor vorzugeben und diese auch automatisch zu bestimmen (SDO-ARRAY 0x3101).

***Hinweis**:* *In den folgenden Beispielen wird der Modus 0x0007 verwendet; natürlich sind aber alle oben beschriebenen Modi zulässig.*

2.6.7.1 Eichfunktion

Die Eichfunktion wird mit RPDO7 bzw. RPDO9 mit (z.B.) folgenden Parametern gestartet:

Beispiel:

CANID = 0x200 (512):	RPDO1 der NodeID 0 = RPDO9 aller anderen Knoten
8 Datenbyte (Lowbyte first):	0x14, 0x98, 0x17, 0x00, 0xDC, 0x05, 0x00, 0x00
→ Steuerwort/Sollwert:	0x9814 = OperationDC, Amplitude 20%
→ Modus:	0x0017 = Drehrichtung alternierend / schachbrettartig, Eichfunktion
→ Intervall:	0x000005DC = Drehdauer: 1500 ms

Beginnend mit der Amplitude AMin0 (SDO 0x5101, Default = 10%) wird jeder Motor mit dieser Amplitude in der Betriebsart „OperationDC“ für die angegebene Dauer (im Beispiel: 1500 ms) angesteuert; anschließend wird festgestellt, ob die Drehzahl des Motor > 2 x UMin0 (SDO 0x5100, Default = 30 U/min) ist, d.h. der Motor **sicher (deshalb 2 x UMin0)** dreht. In diesem Fall wird die Amplitude noch um 1 inkrementiert und **für diesen Motor** als Minimalamplitude gesetzt (SDO-ARRAY 0x3101, AmpMin). Danach beginnt ein neuer Zyklus mit der inkrementierten Amplitude, solange, bis entweder alle Motoren sich drehen oder die im RPDO angegebene Amplitude (im Beispiel: 20%) erreicht ist.

Am Ende meldet jedes Modul mit dem TDO7, für welche Motoren die min. Amplitude erfolgreich bestimmt werden konnte:

Beispiel:

CAN_ID = 0x3AE (942):	TPDO7, NodeID = 41
Datenbytes (Lowbyte first):	0x14, 0x98, 0x17, 0x00, 0xFF, 0xCF, 0x16, x00
→ Steuerwort/Sollwert:	0x9814 = OperationDC, Sollamplitude 20%
→ Modus:	0x0017 = Drehrichtung alternierend / schachbrettartig, Eichfunktion
→ Bewegungsvektor:	0x0016CFFF: Bit 12,13,16,19 = 0 → min. Amplitude der Motoren 13,14,17,20 konnte nicht bestimmt werden.

***Hinweis**:* *Die jeweils gefundene min. Amplitude AmpMin für die einzelnen Motoren kann mit dem SDO 0x3101 ausgelesen werden; ebenso kann AmpMin auch per SDO gesetzt werden.*

2.6.7.2 Beladungstest

Der Beladungstest wird mit RPDO7 bzw. RPDO9 mit (z.B.) folgenden Parametern gestartet:

Beispiel:

CANID = 0x200 (512):	RPDO1 der NodeID 0 = RPDO9 aller anderen Knoten
8 Datenbyte (Lowbyte first):	0x00, 0x98, 0x07, 0x00, 0xDC, 0x05, 0x00, 0x00
→ Steuerwort/Sollwert:	0x9800 = OperationDC, Amplitude 0% → benutze AmpMin (SDO 0x3101) des jeweiligen Motors
→ Modus:	0x0007 = Drehrichtung alternierend / schachbrettartig
→ Intervall:	0x000005DC = Drehdauer: 1500 ms

Es wird eine Bewegungssequenz mit der Betriebsart „OperationDC“ und der Amplitude 0% gestartet: in diesem Fall (und nur in diesem) wird jeder Motor mit seiner Minimalamplitude gestartet. Am Ende der Bewegung meldet wieder jedes Modul mit seinem TPDO7 das Ergebnis der Sequenz.

Beispiel:

CAN_ID = 0x3AE (942):	TPDO7, NodeID = 41
Datenbytes (Lowbyte first):	0x00, 0x98, 0x07, 0x00, 0xFF, 0xCF, 0x16, x00
→ Steuerwort/Sollwert:	0x9800 = OperationDC, Sollamplitude 0% → AmpMin (SDO 0x3101) des jeweiligen Motors
→ Modus:	0x0007 = Drehrichtung alternierend / schachbrettartig
→ Bewegungsvektor:	0x0016CFFF: Bit 12,13,16,19 = 0 → Motoren 13,14,17,20 belegt

Dieses TPDO wurde von Modul 41 am Ende der Bewegungssequenz versandt; die Motoren wurden mit 10% im DC-Betrieb mit schachbrettartig alternierender Drehrichtung angesteuert. Die Drehzahl der Motoren 13,14,17 und 20 war < UMin0 (SDO 0x5100)

3 Inbetriebnahme-Software AC21.EXE

Zur Inbetriebnahme steht das Hilfsprogramm „AC21.EXE“ für den PC unter Windows2000/XP zur Verfügung.

Die Verbindung erfolgt über eine serielle Schnittstelle des PC mit der seriellen Service-Schnittstelle des Moduls oder optional über den CAN-Bus. Für die Kommunikation über den CAN-Bus ist PC-seitig ein entsprechendes CAN-Bus-Modul und eine Schnittstellen-DLL notwendig, die die Kommunikation zwischen AC21.EXE und dem Treiber für das eingesetzte CAN-Bus-Modul implementiert; diese DLL muß für das eingesetzte CAN-Bus-Modul individuell erstellt werden. Details dazu s. Kap. 3.1.3.
Für folgende CAN-Bus-Module steht eine Schnittstellen-DLL zur Verfügung:

- USB-to-CAN compact, Fa. IXXAT: IXXATCAN.DLL
- TinyCAN I, Fa. MHS Elektronik: TinyCAN.DLL
- CAN-PCI, Fa. Pfortner Ltd.: CAN.DLL

Die Schnittstellen-DLL muß sich im gleichen Verzeichnis wie das Programm AC21.EXE befinden.

Folgende Dateien werden benötigt:

- AC21.EXE Programmdatei

Eine der folgenden DLLs zum Zugriff auf das jeweilige CAN-Bus-Modul:

- IXXATCAN.DLL USBtoCAN-Modul der Fa. IXXAT
- TinyCAN.DLL Tiny-CAN-Modul der Fa. MHS
- CAN.DLL CAN-PCI-Karte der Fa. Pfortner

Folgende Verzeichnisse / Dateien werden bei Bedarf automatisch erzeugt:

- AC21.INI Konfigurationsdatei
- *.LOG, *.LO0-*.LO9 Log-Dateien zu Test- und Diagnosezwecken
  Diese Dateien können ggf. gelöscht werden
- Unterverzeichnis „Protokolle“ In diesem Unterverzeichnis werden die erzeugten Protokoll-Dateien abgelegt; dazu wird für jedes Modul ein weiteres Unterverzeichnis mit der 16-stelligen Modul-Seriennummer angelegt. Sollen die Protokoll-Daten in einem anderen Verzeichnis abgelegt werden, so kann dies im Programm geändert werden: s. Kap. 3.3

3.1 Allgemeines

3.1.1 Voraussetzungen

- PC mit Betriebssystem Windows2000/XP
- serielle Schnittstelle (COM:-Anschlüsse): unterstützt werden alle system-konform verfügbaren Anschlüsse. Es ist nur das Modul ansprechbar, das an der seriellen Schnittstelle angeschlossen ist.
- optional CAN-Bus: Dazu ist eine CAN-Bus-Schnittstelle am PC erforderlich. Das Programm erwartet eine DLL, die die Schnittstellenfunktionen implementiert. Alle am CAN-Bus verfügbaren Module sind ansprechbar.

3.1.2 Funktionalität

- Firmware-Update der Einzelprozessoren
- Firmware-Update des Haupt-Prozessors (nur über CAN-Bus)
- Initialisierungslauf der Motoren
- Protokollfunktionen
- Ansteuerung der Einzelprozessoren / Motoren einzeln oder zusammen
- Konfiguration, Bestimmung der Betriebsparameter

Ist die Anwendung über den CAN-Bus angeschlossen, so können alle über den Bus erreichbaren Module bedient werden. Für jedes Modul wir ein eigener SDO-Stack verwaltet.

3.1.3 CAN-DLL – Schnittstellenfunktionen

Für PC-CAN-Bus-Module folgender Hersteller stehen entsprechende DLLs zur Verfügung:

- USB-to-CAN compact, Fa. IXXAT: IXXATCAN.DLL
- TinyCAN I, Fa. MHS Elektronik: TinyCAN.DLL
- CAN-PCI, Fa. Pfortner Ltd.: CAN.DLL

Soll ein anderes PC-CAN-Bus-Modul eingesetzt werden, so muß eine entsprechende CAN.DLL erstellt werden, die die im folgenden beschriebenen Funktionen bereitstellen muß (Delphi-Syntax).

3.1.3.1 Allgemeines

- Alle Funktionen folgen dem Aufrufmodell „**stdcall**“ (C/C++: WINAPI)

- Alle Funktionen geben im Erfolgsfall den Wert „0“, ansonsten einen negativen Fehlercode zurück. Der Code wird vom Programm aber nicht weiter ausgewertet.

- Parameter der Funktionen:

`CANMsgID: LONGWORD`	CAN-ID der Nachricht
`CANData: Pointer`	Pointer auf einen Daten-Puffer für die zu sendenden / empfangenen Daten (mind. 8 Byte!)
`Length: LONGINT`	Anzahl der zu sendenden / empfangenen Daten (max. 8) = Anzahl der gültigen Daten im Daten-Puffer.

Frametyp der zu sendenden / empfangenen CAN-Nachricht:

ExtFrame: BOOL	CANMsgID = 29 Bit-Identifier
RemFrame: BOOL	Die empfangene Nachricht ist ein Remote-Frame bzw. die zu sendende Nachricht soll als Remote-Frame versandt werden

3.1.3.2 CAN_InitEx

```
FUNCTION CAN_InitEx(ABaudRate: INTEGER): INTEGER; stdcall;
```

Initialisiert die gesamte CAN-Schnittstelle mit der angegebenen Baudrate (in Bit/sec).

3.1.3.3 CAN_Done

```
FUNCTION CAN_Done: INTEGER; stdcall;
```

Beendet die CAN-Kommunikation.

3.1.3.4 CAN_SendMessage

```
FUNCTION CAN_SendMessage(CANMsgID: LONGWORD; CANData: Pointer; Length: LONGINT; ExtFrame,RemFrame: BOOL): LONGINT; stdcall;
```

Übergibt eine CAN-Nachricht zum Senden an die CAN-Schnittstelle. Die Funktion wartet nicht(!) bis die Nachricht versandt wurde!

3.1.3.5 CAN_RcvMessage

```
FUNCTION CAN_RcvMessage(VAR CANMsgID: LONGWORD; CANData: Pointer; VAR Length: LONGINT; VAR ExtFrame,RemFrame: BOOL; TimeOut: DWORD): LONGINT; stdcall;
```

Liest die nächste, empfangen CAN-Nachricht; wartet ggf. bis zum Empfang bzw. bis Timeout (in ms) abgelaufen ist!

3.1.3.6 CAN_GetMessage

```
FUNCTION CAN_GetMessage(VAR CANMsgID: LONGWORD; CANData: Pointer; VAR Length: LONGINT; VAR ExtFrame,RemFrame: BOOL): LONGINT; stdcall;
```

Liest die nächste, empfangen CAN-Nachricht, wenn vorhanden.

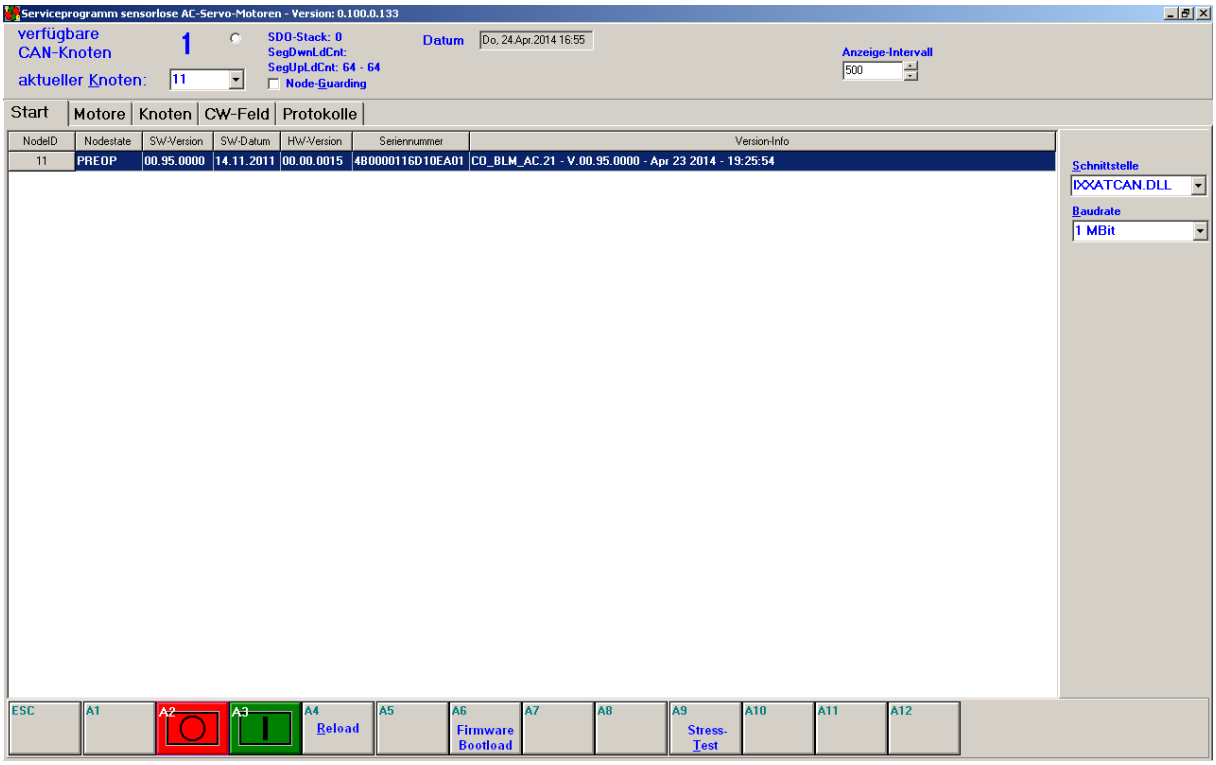
3.1.3.7 CAN_IFCEmpty

```
FUNCTION CAN_IFCEmpty: BOOL; stdcall;
```

Überprüft, ob alle Nachrichten versandt wurden.

3.2 Seite „Start“

Auf der Startseite werden alle angeschlossenen Module mit Node-ID, aktuellem Node-Status und Versions-Information aufgeführt. Zusätzlich wird die Anzahl der verfügbaren Module angezeigt.



Funktionstasten:

- F2: alle Module werden in den Zustand PREOPERATIONAL gesetzt: Alle Motore werden gestoppt und stromlos geschaltet
- F3: z.Zt. ohne Funktion
- F4: Die Anzeige bzw. die interne Knotenverwaltung wird gelöscht und neu aufgebaut
- F9: Test-Zyklus „Stress-Test“ aufrufen, s. Kap. 3.7

Haupfenster:
In diesem Bereich werden alle am CAN-Bus gefundenen Module mit ihren Kenndaten aufgelistet.

Bedienelemente - lokal:

- „Schnittstelle“: Auswahl der gewünschten Schnittstelle (COMx: bzw. CAN-Bus)
- "Baudrate": ab V.0.90: Auswahl der CAN-Baudrate

***Hinweis**:* *Z.Zt unterstützen die AC21-Module die Baudraten 1Mbit, 500kBit und 250kBit*

Anzeigeelemente - global:
Diese Elemente sind auch auf allen folgenden Seiten sichtbar:

- Datum: aktuelles Datum / Uhrzeit

Kommunikationszustand des ausgewählten Knotens:

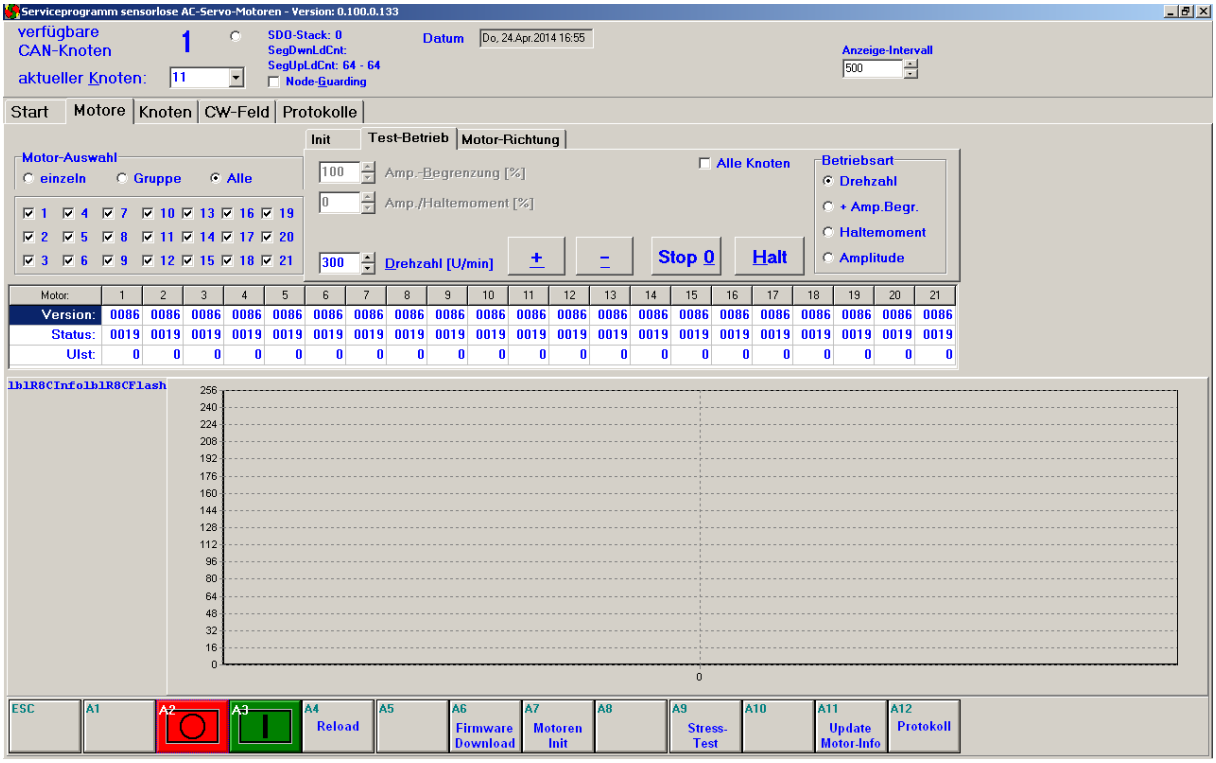
- „SDOStack“: Anzahl der noch zu sendenden SDO-Nachrichten

- „SegDwnLdCnt“: Größe des nächsten zu sendenden Datensegments und Anzahl der bereits gesendeten Datenbytes.
- „SegUpLdCnt“: Größe des nächsten zu empfangenden Datensegments und Anzahl der bereits empfangenen Datenbytes.

Bedienelemente - global:
Diese Elemente sind auch auf allen folgenden Seiten verfügbar:

- „aktueller Knoten“: Auswahl eines Knotens zur Bearbeitung/Bedienung; s. folgende Kapitel
- „NodeGuarding“: Node Guarding ist aktiv, d.h. es wird ca. alle 2 sec eine ERRCTRL-Message versandt, auf die die einzelnen Module reagieren sollten (s. Kap. 2.5.2).

3.3 Seite „Motore“



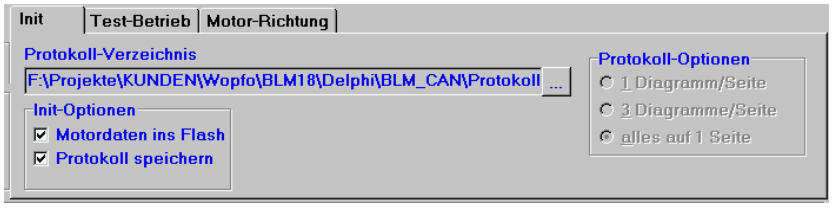
Funktionstasten:

- F2: das aktuelle Modul wird in den Zustand PREOPERATIONAL gesetzt
  Alle Motore werden gestoppt und stromlos geschaltet
- F3: das aktuelle Modul wird in den Zustand OPERATIONAL gesetzt
  In diesem Zustand können die Motore bewegt werden.
- F6: Firmware-Download für die Motorprozessoren
- F7: Initialisierungslauf der Motoren
- F9: Test-Zyklus „Stress-Test“ aufrufen, s. Kap. 3.7
- F11: Auslesen der Initial- und Informationsdaten aus den Motorprozessoren
- F12: Protokoll erzeugen

Bedienelemente:

- „Motor-Auswahl“: Für die verfügbaren Optionen können hier einzelne Motoren ausgewählt werden. Wird „Alle“ angeklickt, so wird im darunterliegenden Auswahlfeld alles angewählt und das Feld für die Eingabe gesperrt.

Register „Init“:

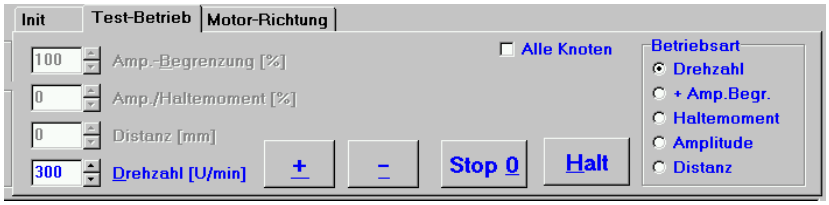


- „Protokoll-Verzeichnis“: In diesem Verzeichnis werden die Protokolle gespeichert, die bei den verschiedenen Tests / Initialisierungen der Module erzeugt werden; ist

das Verzeichnis nicht vorhanden, so wird es angelegt. Für jedes Modul wird ein Unterverzeichnis mit der 16-stelligen Seriennummer des Moduls erzeugt. Wird kein Verzeichnis angegeben, so wird im Programm-Verzeichnis das Unterverzeichnis „Protokolle“ benutzt und ggf. vorher angelegt.

- „Init-Optionen“:
  - Motordaten ins Flash: legt fest, ob die im Initialisierungslauf bestimmten Daten im internen Datenflash gesichert werden
  - Protokoll speichern: Die beim Initialisierungslauf bestimmten Daten werden in eine Protokolldatei geschrieben
- „Protokoll-Optionen“: Ist im Feld „Motor-Auswahl“ eine Gruppe von Motoren ausgewählt, so kann hier festgelegt werden, wie viele Motor-Diagramme auf einer Ausgabeseite ausgegeben werden sollen.
  Ist „Alle“ angewählt so werden immer alle 21 Diagramme verkleinert auf 1 DIN-A4-Seite ausgegeben.
  - 1 Diagramm/Seite: Für jedes Diagramm wird 1 DIN-A4-Seit im Querformat ausgegeben
  - 3 Diagramme/Seite: je 3 Diagramme werden auf einer DIN-A4-Seite im Hochformat ausgegeben
  - alles auf 1 Seite: Die Diagramme werden so verkleinert, daß alle auf eine DIN-A4-Seite im Querformat passen.

Register „Test-Betrieb“



- „Alle Knoten“: Unabhängig vom ausgewählten Modul wirken die Eingaben auf alle angeschlossenen CAN-Module

- Betriebsart: Die im folgenden beschriebenen Betriebsarten können angewählt werden. Die jeweilige Betriebsart wird erst durch einen Klick auf einen der Buttons aktiviert; je nach Betriebsart sind nicht alle Buttons / Eingabefelder aktiv, s. bei der jeweiligen Beschreibung.
  - Drehzahl: Die angewählten Motoren werden mit der eingestellten Drehzahl verfahren.
  - + Amp.Begr. zusätzlich wird die maximale PWM-Amplitude (→ Strombegrenzung) auf den eingestellten Wert begrenzt.
  - Haltemoment: Die angewählten Motoren werden mit der eingestellten Amplitude (Eingabefeld „Prozent“) im Stillstand bestromt (gehalten); um Fehlbedienungen zu vermeiden, wird bei Auswahl dieser Betriebsart **immer** zunächst ein Haltestrom von 0% vorgegeben.
  - Amplitude: Die angewählten Motoren werden mit der eingestellten PWM-Amplitude (Eingabefeld „Prozent“) angesteuert (ungeregelt). Die Motoren verhalten sich damit quasi wie Gleichstrom-(DC-)Motoren.
  - Distanz: ab V.0.90: Die angewählten Motoren werden um die eingestellte Distanz und mit der eingestellten Drehzahl positioniert.
    ***Hinweis**:* *Die max. Distanz beträgt ca. 5000m bei einem Rollendurchmesser von 50mm. Die Drehrichtung ergibt sich aus der Drehzahl*

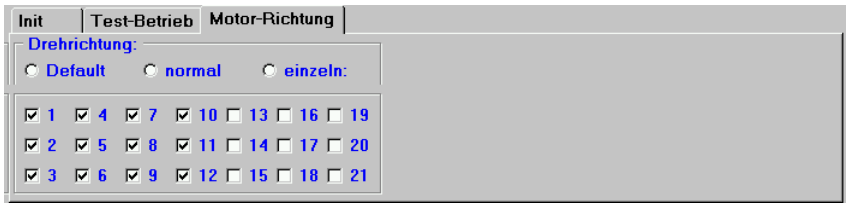
- „Drehzahl“: Hier kann die gewünschte Drehzahl eingegeben oder mit den beiden Pfeil-Buttons gewählt werden; alle angewählten Motoren drehen mit der gleichen Drehzahl, die Richtung ist aber abhängig vom Feld „Drehrichtung“, s.o.

Das Eingabefeld ist nur in den Betriebsarten „Drehzahl“, „Drehzahl + Amp.Begr.“ und "Distanz" aktiv.

- „Prozent“: Eingabefeld für das Haltemoment (% Nennstrom) bzw. die Amplitude (in % der Maximal-Amplitude). Bei Anwahl der Betriebsart „Haltemoment“ wird das Feld zwangsweise zunächst auf 0% gesetzt.
  Das Eingabefeld ist nur in den Betriebsarten „Haltemoment“ bzw. „Amplitude“ aktiv.
- „Amp.-Begrenzung“: Eingabefeld für die Amplituden-Begrenzung (in % der Maximal-Amplitude). Der eingegebene Wert wird in einem Motor-Daten-Objekt abgelegt (s.o.).
  Das Feld ist nur in der Betriebsart „Drehzahl + Amp.Begr.“ aktiv.
- „Distanz“: Eingabefeld für die Distanz (in mm).
  Das Feld ist nur in der Betriebsart „Distanz“ aktiv.

- „+/-“- Buttons: Drehrichtung der angewählten Motoren in den Betriebsarten „Drehzahl“, „Drehzahl + Amp.Begr.“, "Distanz" und „Amplitude“. In der Betriebsart „Haltemoment“ sind die Buttons nicht aktiv.
- „Halt“: Dieser Button hat in den verschiedenen Betriebsarten leicht unterschiedliche Funktion:
  - Drehzahl:
  - + Amp.Begr.
  - Distanz — Die angewählten Motoren werden gestoppt, die Regelung bleibt aktiv.
  - Haltemoment: (Eingabefeld „Prozent“) — Die angewählten Motoren werden mit eingestellten Amplitude im Stillstand bestromt (gehalten).
  - Amplitude: — Die angewählten Motoren werden mit 0% PWM-Amplitude angesteuert.
- „Stop 0“: Die angewählten Motoren werden gestoppt; die Freigabe bleibt aktiv, d.h. die Motoren werden mit 0% PWM bestromt.

Register „Motor-Drehrichtung“



- „Drehrichtung“: Bedingt durch den mechanischen Aufbau muß die tatsächliche Drehrichtung der Motoren 1-12 oder 13-21 invertiert werden, damit sich alle Motoren gleichsinnig drehen. Dieses Verhalten kann hier angepasst werden:
  - Default: Die Drehrichtung der Motoren 1-12 wird invertiert, alle Motoren drehen gleichsinnig (entspr. dem mech. Aufbau).
  - Normal: alle Motoren drehen in die gleiche Richtung
  - Einzeln: Im darunterliegenden Auswahlfeld kann die Drehrichtung für jeden Motor einzeln invertiert werden.

Hinweis:
Die aktuelle Zuordnung wird zunächst aus der AC21 ausgelesen. Im Auslieferungszustand ist die Drehrichtung „Normal“! Werden die Modul-Parameter neu gesetzt (s. Kap. 3.4), so werden die Drehrichtungen der Motoren auf „Default“ gesetzt.

Anzeigeelemente:

- Motor- Tabelle: Für jeden Motor des ausgewählten Moduls werden folgende Daten angezeigt:
  - Version: Versionsnummer der Firmware im Motor-Prozessor
  - Status: Statuswort, s. Kap. 2.4.1

- Drehzahl: aktuelle Drehzahl
  ***Hinweis**:* *u.U. ist die Drehrichtung des Motors invertiert, s.o.* ***„Drehrichtung“**.*

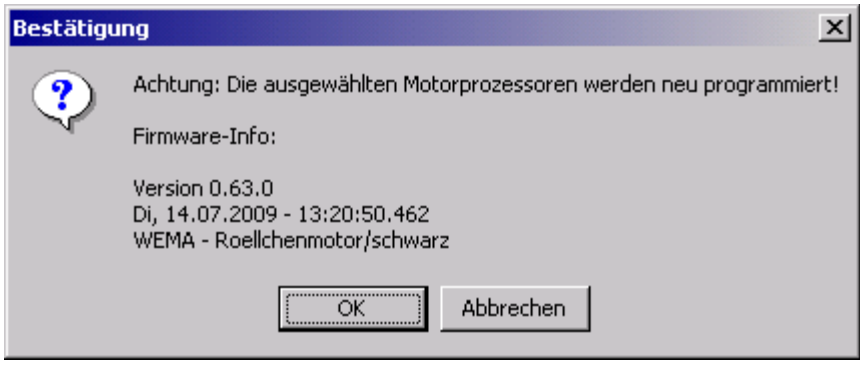
- Grafik-Feld: Hier wird die beim Initialisierungslauf des Motors aufgenommene Messkurve dargestellt. Sie sollte typischerweise so aussehen wie Bild oben. Abweichungen deuten auf einen fehlerhafte Motor.
- Infofeld: In diesem Feld werden einige Kenndaten des angewählten Motors im Klartext ausgegeben

***Hinweis**:* *Die Messkurve und die Kenndaten werden nur nach einem Initialisierungslauf automatisch ausgelesen; sollte bei der Auswahl eines Motors das Feld leer bleiben, so kann mit der Funktionstaste „F11“ eine Aktualisierung veranlasst werden.*

3.3.1 Firmware-Update

Ein Firmware-Update der Motorprozessoren wird mit „F6“ – „Firmware Download“ gestartet.

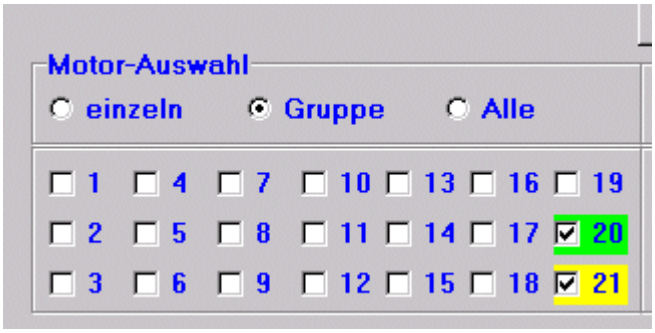
Um ein versehentliches Programmieren zu verhindern, erscheint zu Beginn eine Abfrage und der Information zur neuen Firmware-Version. Danach wird zunächst die Firmware per SDO-Segment-Transfer übertragen – erkennbar am Hochzählen der Anzeige „SegDwnLdCnt“.



Anschließend beginnt für die angewählten Motoren der Programmiervorgang, der pro Motor ca. 10 sec in Anspruch nimmt.

Während des Programmiervorgangs flackert die rote LED am Modul. Der Fortschritt wird außerdem im Motor-Auswahlfeld farblich signalisiert:

- gelb: Programmierung läuft
- grün: Programmierung erfolgreich beendet
- rot Programmierung nicht erfolgreich; Ursache ist hier in der Regel ein Hardware-Fehler auf der Platine.



Am Ende der Programmierung erscheint eine Meldung; die Zahl dahinter dient Protokollzwecken. Das Ergebnis der Programmierung wird außerdem in einer Protokolldatei im Protokoll-Verzeichnis (Unterverzeichnis <Seriennummer> des Moduls) festgehalten.

Anschließend sollte für die neu programmierten Motor-Prozessoren ein Initialisierungslauf durchgeführt werden.

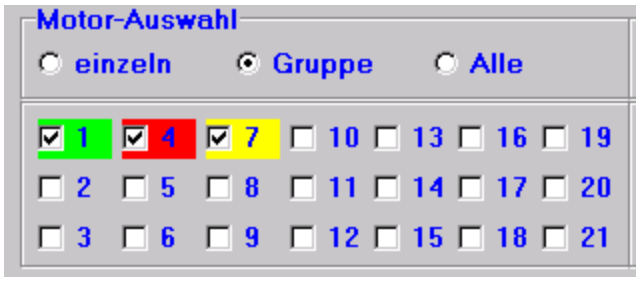
3.3.2 Initialisierungslauf

Die Drehzahlregelung der einzelnen Motoren geschieht durch elektronische Messung ohne Sensoren. Die dazu notwendigen Parameter hängen nicht nur vom Motortyp sondern auch vom konkreten Motor ab und müssen vorher bekannt sein. In einem Initialisierungslauf werden diese Parameter automatisch bestimmt. Die dabei erzeugte Messkurve liefert auch eine Aussage über die Güte des Motors. Da u.U. die Stromaufnahme beim Mess-Vorgangs während der Initialisierung recht hoch sein kann, werden die Motoren eines Moduls prinzipiell nacheinander initialisiert, um Verfälschungen der Messung zu vermeiden.

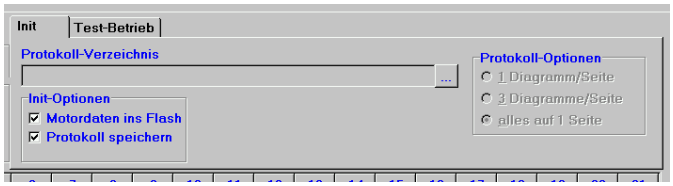
Der Initialisierungslauf wird mit „F7“ – „Motoren Init“ gestartet.

Um ein versehentliches Initialisieren zu verhindern, erscheint zu Beginn eine Abfrage. Danach beginnt für die angewählten Motoren der Initialisierungs-vorgang, der pro Motor ca. 2-3 sec in Anspruch nimmt. Der Fortschritt wird im Motor-Auswahlfeld farblich signalisiert:

- gelb: Initialisierung läuft
- grün: Initialisierung erfolgreich beendet
- rot Initialisierung nicht erfolgreich; dies kann verschiedene Ursachen haben:
  - Motor oder einzelne Phase nicht angeschlossen
  - schlechte Mess-Signal-Qualität
  - Masse- oder Phasenschluß innerhalb des Motors
  - ...



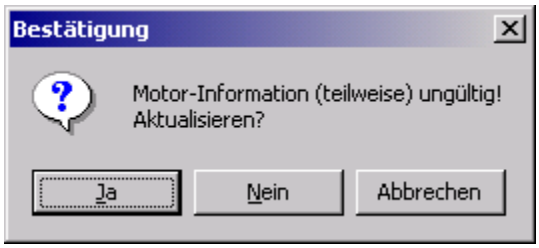
Je nach ausgewählter „Init-Option“ werden die gefundenen Parameter im Motor-Datenflash gespeichert („Motordaten isn Flash“) bzw. als Datei auf dem PC gesichert („Protokoll speichern“. Danach werden alle Messkurven ausgelesen und können direkt über „F12“ – „Protokoll“ angezeigt werden.



3.3.3 Protokoll

Mit der Funktionstaste „F12“ kann ein Protokoll für die ausgewählten Motoren erstellt und ausgedruckt werden. Die Protokolldaten werden in einem Unterverzeichnis des Protokoll-Verzeichnis (s.o.) abgelegt; der Name des Unterverzeichnis entspricht der Seriennummer des Moduls.

Das Protokoll wird aus den Messkurven und den Kenndaten aus den Motorprozessoren erzeugt; wurden die Daten nicht ausgelesen oder sind sie ungültig, so erscheint eine entsprechende Abfrage: „Nein“ erzeugt zwar ein Protokoll, u.U. sind die Daten aber unvollständig bzw. unsinnig!

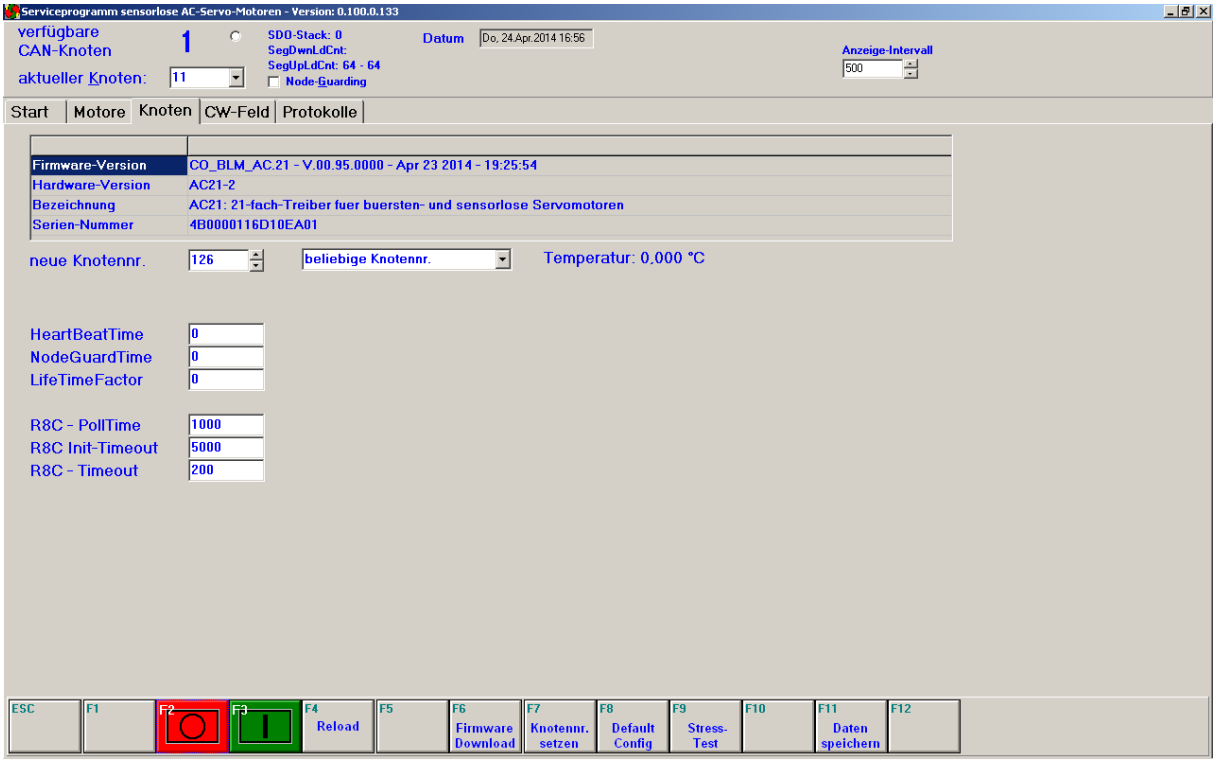


Im einzelnen werden folgende Daten erzeugt:

- Protokolldateien im Unterverzeichnis <Seriennummer>
  Diese Dateien können auch noch nachträglich ausgewertet werden; s. Kap. 3.5, Seite „Protokolle“.
- Protokoll-Ausdrucke
  Entsprechend der angewählten „Protokoll-Optionen“, wird eine Seitenvorschau erzeugt, die auch ausgedruckt werden kann:

3.4 Seite „Knoten“

Diese Seite dient der Konfiguration der einzelnen Module.



Funktionstasten:

- F2: das aktuelle Modul wird in den Zustand PREOPERATIONAL gesetzt
  Alle Motore werden gestoppt und stromlos geschaltet
- F3: z.Zt. ohne Funktion
- F4: Reload: lädt alle Daten neu aus dem ausgewählten Modul
- F6: Firmware-Download für den CAN-Prozessoren
- F7: Die im Feld „neue Knotennr.“ eingegebene Knotennummer wird an das Modul übertragen. Anschließend wird ein Befehl „RESET COMMUNICATION“ an das Modul gesandt. Danach ist das Modul unter der neuen Knotennummer ansprechbar.
  Die Knotennummer wird jedoch nicht im internen Flash gesichert, d.h. sie ist nur bis zum nächsten Reset gültig
- SHIFT-F7: Wird F7 zusammen mit der Hochstelltaste (Shift-Taste) gedrückt, so wechselt die Beschriftung nach "Baudrate setzen": die im Feld "neue Baudrate" ausgewählte Baudrate wird and das Modul übertragen. Anschließend wird ein Befehl „RESET COMMUNICATION“ an das Modul gesandt. Danach erfolgt die Kommunikation mit dem Modul mit der neuen Baudrate.
  Die Baudrate wird jedoch nicht im internen Flash gesichert, d.h. sie ist nur bis zum nächsten Reset gültig
- F8: Alle Einstellungen werden auf ihre Default-Werte gesetzt; insb. wird dir Drehrichtung aller Achsen auf „Default“ gesetzt (s. Kap. 3.3, Register „Motor-Drehrichtung“)
- F9: Test-Zyklus „Stress-Test“ aufrufen, s. Kap. 3.7
- F11: Alle eingegebenen Konfigurationsparameter incl. Knotennummer und Baudrate werden im internen Datenflash gespeichert und sind damit direkt nach dem Einschalten verfügbar.

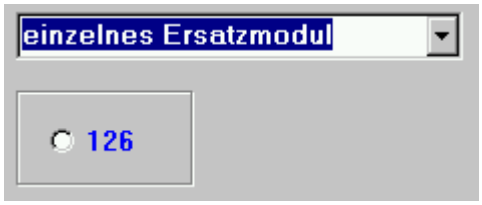
***Hinweise**:* *Beim Ändern der Knotennummer ist besondere Sorgfalt angebracht, da die Knotennummer ein Modul am CAN-BUS eindeutig identifiziert. Wurde eine Knotennummer mehrfach vergeben und auch noch mit „F11“ dauerhaft gespeichert, so kann dies nur durch Abkoppeln aller anderen Moduln vom CAN-Bus oder durch den Anschluß des Moduls an die serielle Schnittstelle geändert werden.*

Bedienelemente:

- „neue Knotennr.“: neue Knotennummer für dieses Modul – Objekt 0x2000.01.
  Die Knotennummer wird erst mit „F7“ – „Knotennummer setzen“ – an das Modul übertragen; danach ist das Modul unter der neuen Knotennr. ansprechbar. Mit „F11“ – „Daten speichern“ – wird die Knotennr. dauerhaft gesetzt, andernfalls ist nach einem Reset bzw. erneuten Einschalten wieder die alte Knotennr. gültig.
- „neue Baudrate“: neue Baudrate für dieses Modul – Objekt 0x2000.02.
  Die Baudrate wird erst mit „Shift-F7“ – „Baudrate setzen“ – an das Modul übertragen; danach erfolgt die Kommunikation mit dieser Baudrate. Mit „F11“ – „Daten speichern“ – wird die Baudrate dauerhaft gesetzt, andernfalls ist nach einem Reset bzw. erneuten Einschalten wieder die alte Baudrate gültig.

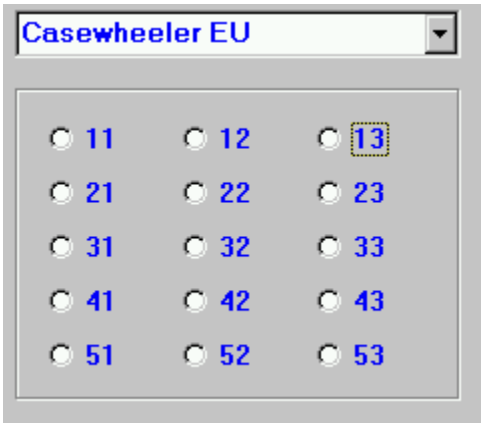
Das Eingabefeld „neue Knotennr.“ ist nur aktiv, wenn im Ausklappfeld rechts daneben die Option „beliebiege Knotennr.“ angewählt ist.
Der Bequemlichkeit halber lassen sich damit verschiedene, oft gebrauchte Modul-Konfigurationen direkt anwählen, so daß die für diese Modul-Konfiguration möglichen/zulässigen Knoten-Nummern direkt gesetzt werden können: Die ausgewählte Knotennummer wird direkt übertragen („F7“ ist also nicht notwendig), jedoch wiederum erst mit „F11“ dauerhaft gespeichert.

- einzelnes Ersatzmodul: Dieses hat immer die Knotennummer 126.

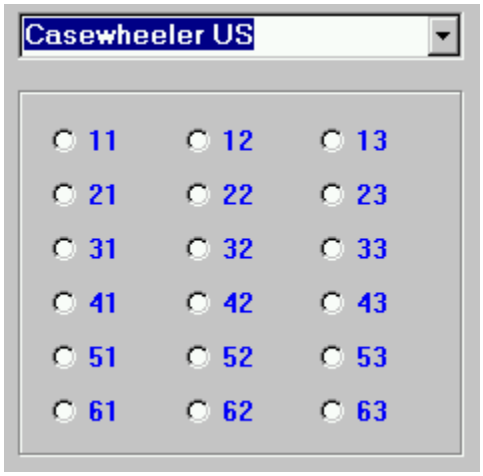


- Casewheeler-Ersatz-Modul: Modul aus 3 montierten AC21-Baugruppen. Hier sind die Knotennummern 111-113 zugelassen.

- Casewheeler – EU: Motoren-Feld aus 5 Casewheeler-Modulen. Für dieses Feld sind die Knotennummern 11-13, 21-23, 31-33, 41-43 und 51-53 zugelassen.



- Casewheeler – US: Motoren-Feld aus 6 Casewheeler-Modulen. Für dieses Feld sind die Knotennummern 11-13, 21-23, 31-33, 41-43, 51-53 und 61-63 zugelassen.



Die folgenden Daten werden direkt übertragen und sind direkt gültig; dauerhaft gespeichert werden sie aber erst mit „F11“ – „Daten speichern“:

- „HeartBeatTime“: – Objekt 0x1017
- „NodeGuardTime“: – Objekt 0x100C
- „LifeTimeFactor“: – Objekt 0x100D

Diese 3 Parameter dienen der Modul-Überwachung; Details dazu s. Kap. 2.5. Da das Programm AC21.EXE eine NMT-ERRCTRL-Message ca. alle 2 sec versendet, sollte dies bei der Konfiguration entsprechend berücksichtigt werden.

- „R8C-PollTime“: Abfrage-Intervall für den Motorstatus – Objekt 0x2420, s. Kap. 2.3.3.4
- „R8C-Init-Timeout“: Timeout für den Initialisierungslauf – Objekt 0x2431 s. Kap. 2.3.3.4
- „R8C-Timeout“: – noch nicht implementiert –

Ab Version 0.90 stehen weitere Konfigurationsmöglichkeitn zur Verfügung:

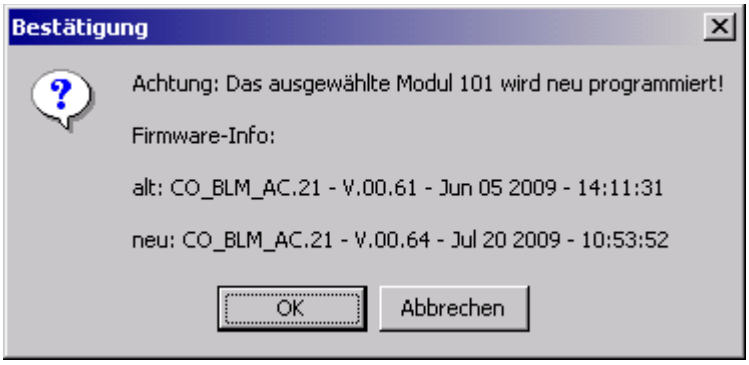
- Rollendurchmesser: Durchmesser der Motoren in mm (alle Motoren!); dieser Wert für die Betriebsart "Distanz" benötigt
- PDONr: Für die PDO5-8 können beliebige IDs vergeben werden; diese werden ebenfalls im Datenflash dauerhaft gespeichert.

Anzeigeelemente:

- Tabelle: Kenndaten des Moduls
- Temperatur: Ist das AC21-Modul eine TK-Version (TK = Tiefkühl), so wird hier die Temperatur des Moduls angezeigt.

3.4.1 Firmware-Update

Ein Firmware-Update des CAN-Bus-Prozessors wird mit „F6“ – „Firmware Download“ gestartet.



Um ein versehentliches Programmieren zu verhindern, erscheint zu Beginn eine Abfrage und der Information zur neuen Firmware-Version. Danach wird die Firmware per SDO-Segment-Transfer übertragen und parallel dazu in den Programm-Speicher des Prozessors geschrieben – erkennbar am Hochzählen der Anzeige „SegDwnLdCnt“. Anschließend wird ein prozessor-interner Reset ausgelöst und es beginnt die normale BOOTUP-Prozedur. Der gesamte Programmiervorgang nimmt ca. 30 sec in Anspruch.

Während des Programmiervorgangs flackert die rote LED am Modul; außerdem ist die Tabelle mit den Modul-Kenndaten gelöscht:

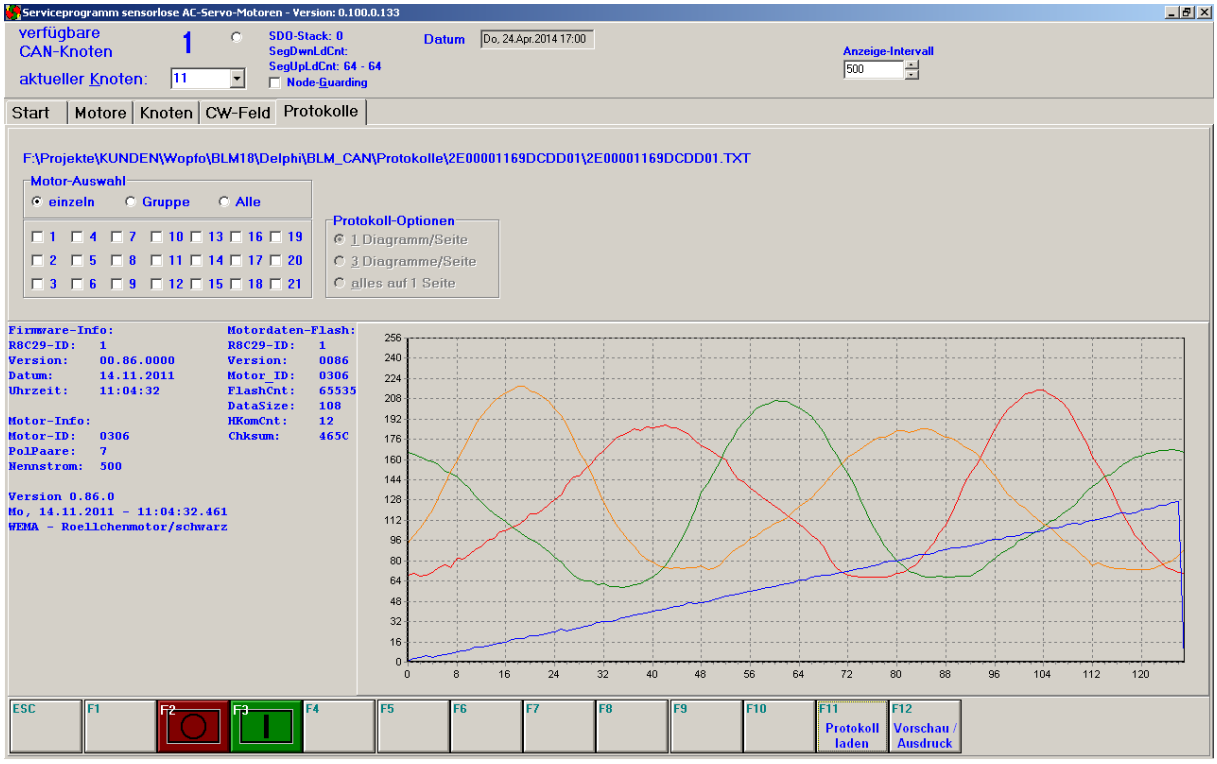
***Hinweise**:*

U.U. sind die im Datenflash des Prozessors gespeicherten Parameter mit der neu aufgespielten Firmware-Version nicht mehr kompatibel; in einem solchen Fall werden alle Parameter auf ihre Default-Werte gesetzt und müssen ggf. neu eingegeben werden. Insbesondere wird die Knotennr. auf ihren Defaultwert (=126) gesetzt. Um zu verhindern, daß sich versehentlich mehr als ein Knoten mit der Nr. 126 im System befindet, ist ein Firmware-Update für andere Knotennummern nicht zulässig, wenn sich ein Knoten mit der Nr. 126 im System befindet.

Da vor der Neu-Programmierung das Programm-Flash des Prozessors gelöscht werden muß, befindet sich die Programmier-Routine im Prozessor-RAM; treten während des Programmierens Fehler auf, die zum Abbruch des Programmier-Vorgangs führen, so muß das Modul ausgebaut und extern über die serielle Service-Schnittstelle neu programmiert werden.

3.5 Seite „Protokolle“

Die manuell oder im Zuge des Initialisierungslaufs erstellten Protokoll-Dateien können geladen und dargestellt werden. Stehen Daten aus der Funktion „Stress-Test“ (s. Kap. 3.7) zur Verfügung, so werden diese auf den Protokoll-Ausdrucken ebenfalls mit ausgegeben



Funktionstasten:

- F2: z.Zt. ohne Funktion
- F3: z.Zt. ohne Funktion
- F11: Protokoll aus Datei laden
  Datei-Auswahl des darzustellenden Protokolls:
  *.TXT: Protokolldatei im Textformat; aus dieser Datei und den zugehörigen Binärdateien für die einzelnen Motorprozessoren wird das Protokoll genau so erstellt, wie wenn die Daten direkt aus dem entsprechenden Modul gelesen werden.
  *.QRP: Protokoll-Vorschau-Datei
- F12: Protokoll anzeigen

Bedienelemente:

- „Motor-Auswahl“: Hier können einzelne Motoren ausgewählt werden. Wird „Alle“ angeklickt, so wird im darunterliegenden Auswahlfeld alles angewählt und das Feld für die Eingabe gesperrt.
- „Protokoll-Optionen“: Ist im Feld „Motor-Auswahl“ eine Gruppe von Motoren ausgewählt, so kann hier festgelegt werden, wie viele Motor-Diagramme auf einer Ausgabeseite ausgegeben werden sollen.
  Ist „Alle“ angewählt so werden immer alle 21 Diagramme verkleinert auf 1 DIN-A4-Seite ausgegeben.
  - 1 Diagramm/Seite:
    Für jedes Diagramm wird 1 DIN-A4-Seite im Querformat ausgegeben
  - 3 Diagramme/Seite: je 3 Diagramme werden auf einer DIN-A4-Seite im Hochformat ausgegeben

– alles auf 1 Seite: Die Diagramme werden so verkleinert, daß alle auf eine DIN-A4-Seite im Querformat passen.

Anzeigeelemente:

- Grafik-Feld: Hier wird die beim Initialisierungslauf des Motors aufgenommene Messkurve dargestellt. Sie sollte typischerweise so aussehen wie Bild oben. Abweichungen deuten auf einen fehlerhafte Motor.
- Infofeld: In diesem Feld werden einige Kenndaten im Klartext ausgegeben

3.5.1 Beispiele für Protokoll-Ausdrucke:

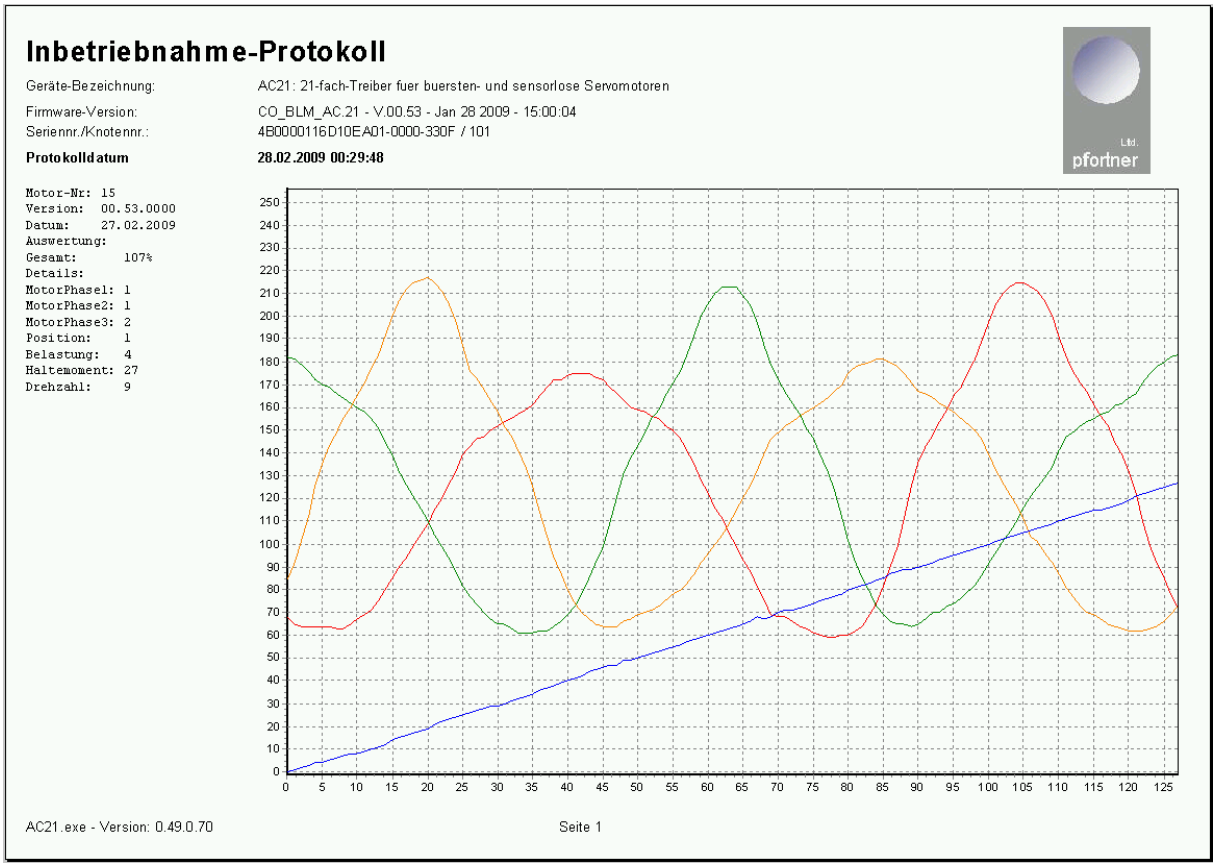
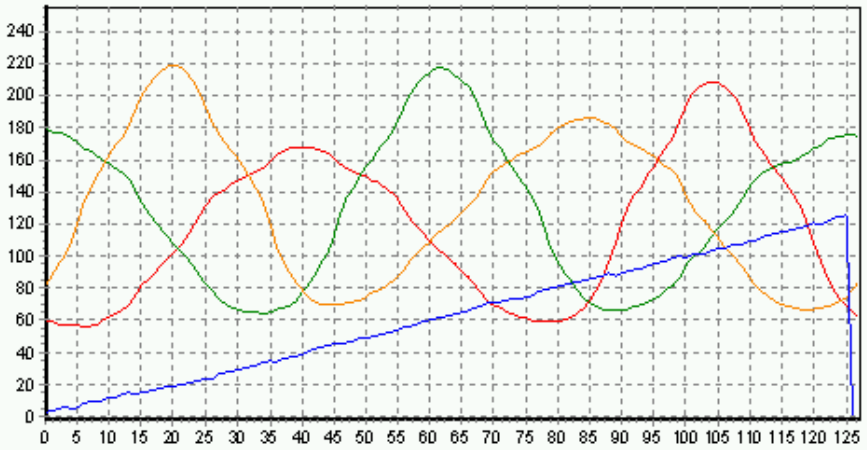


Abbildung 6: 1 Diagramm / Seite

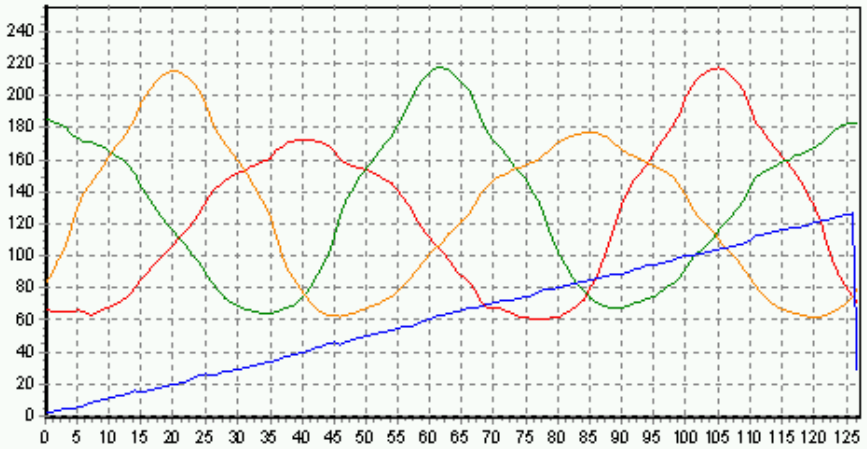
Inbetriebnahme-Protokoll

Geräte-Bezeichnung	AC21: 21-fach-Treiber fuer buersten- und sensorlose Servomotoren
Firmware-Version:	CO_BLM_AC.21 - V.00.53 - Jan 28 2009 - 15:00:04
Seriennr./Knotennr.:	4B0000116D10EA01-0000-330F / 101
Protokolldatum	28.02.2009 00:29:48

Motor-Nr: 16
Version: 00.53.0000
Datum: 27.02.2009
Auswertung:
Gesamt: 107%
Details:
MotorPhasen: 7
Belastung: 4
Haltemoment: 27
Drehzahl: 9



Motor-Nr: 17
Version: 00.53.0000
Datum: 27.02.2009
Auswertung:
Gesamt: 105%
Details:
MotorPhasen: 6
Belastung: 4
Haltemoment: 27
Drehzahl: 9



Motor-Nr: 18
Version: 00.53.0000
Datum: 27.02.2009
Auswertung:
Gesamt: 111%
Details:
MotorPhasen: 9
Belastung: 4
Haltemoment: 27
Drehzahl: 9

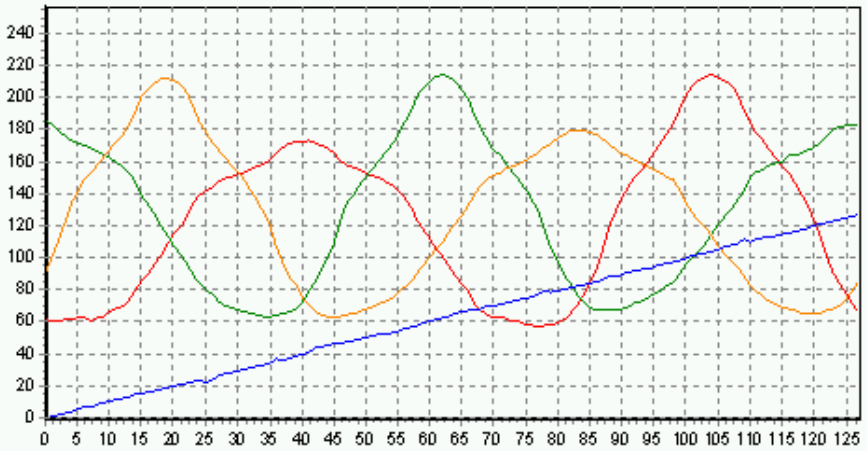


Abbildung 7: 3 Diagramme / Seite

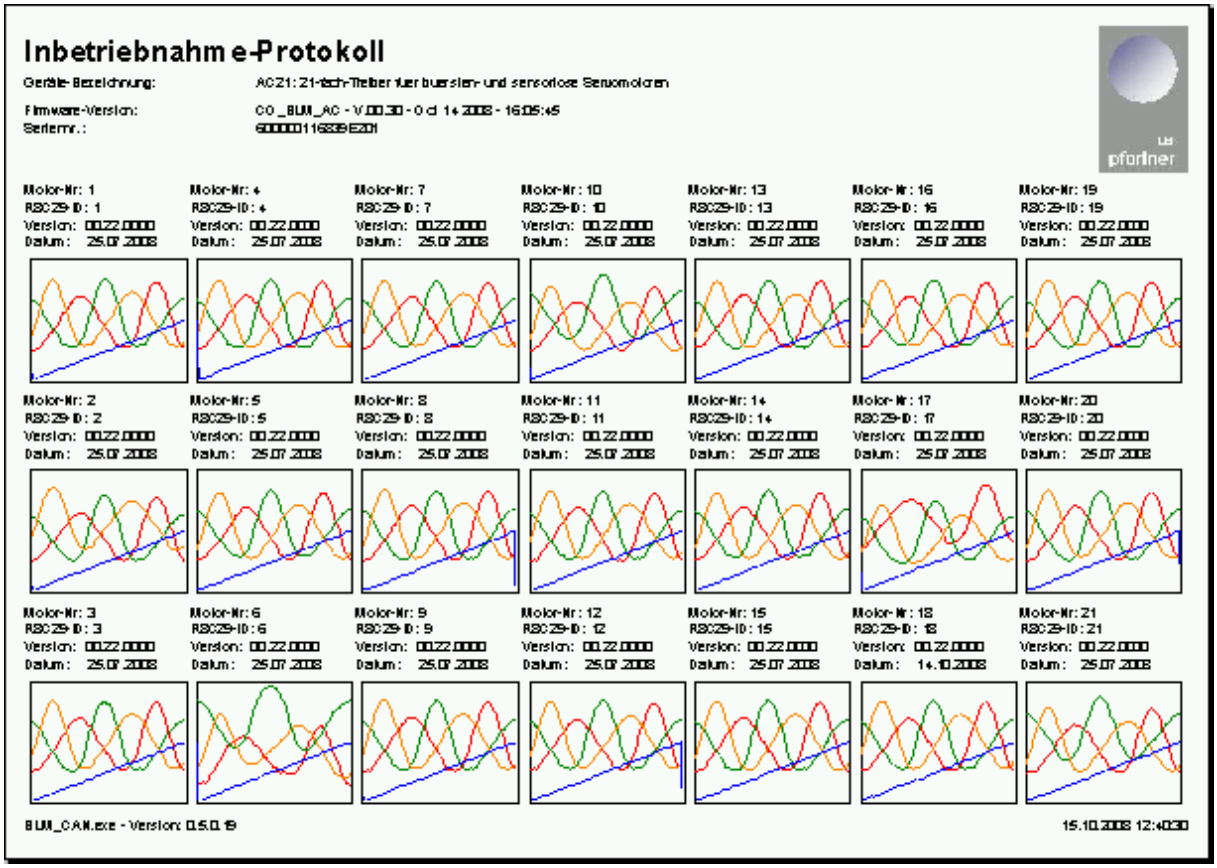
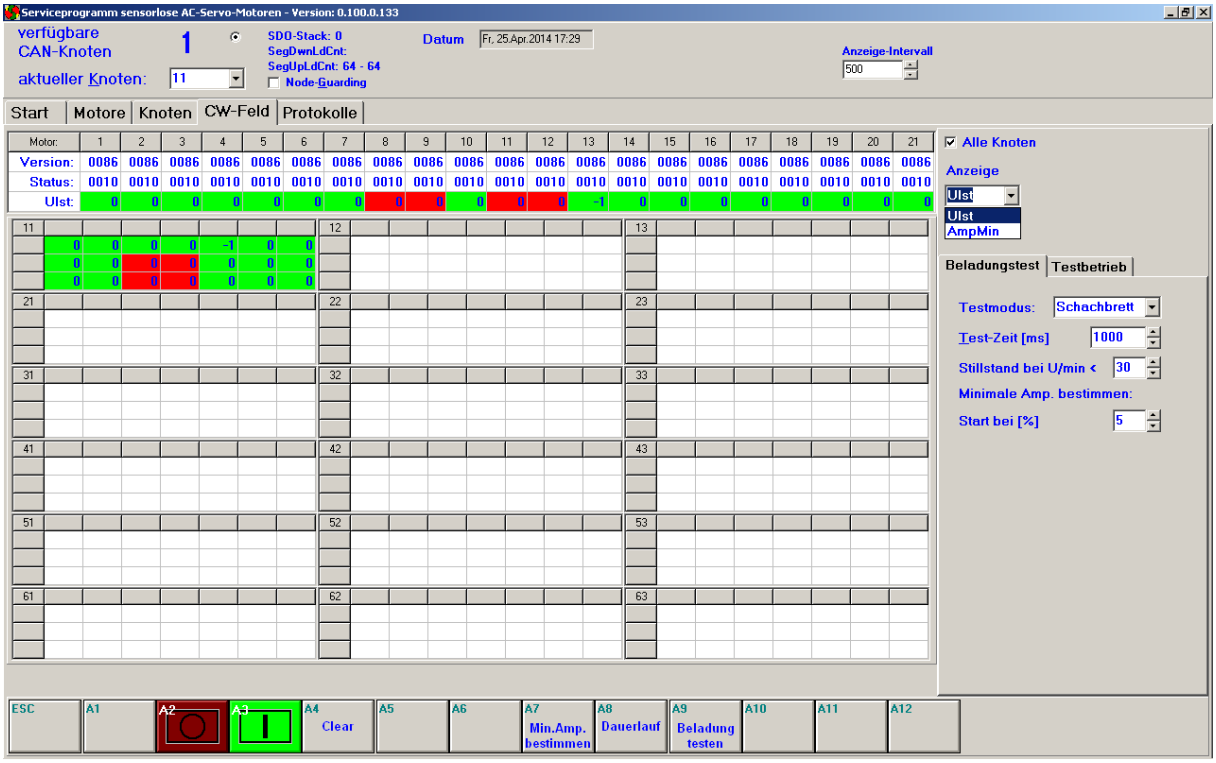


Abbildung 8: alle Diagramme auf 1 Seite

Im letzten Beispiel ist außerdem zu sehen, daß insb. Motor 6 eine deutlich abweichende Messkurve aufweist; auch die Motoren 10, 17 und 21 weichen von der typischen Kurvenform ab. Solche Motoren sollten bei der Inbetriebnahme ausgesondert werden!

3.6 CW-Feld

Das Fenster zeigt ein komplettes CW-(Case-Wheeler-)Feld mit 6 x 3 Motorfeldern und den standardmäßig vorgesehenen Knotennummern. Je nach Auswahl im Ausklappfeld „Anzeige“ wird in den einzelnen Feldern entweder die Drehzahl oder die PWM-Amplitude der Motoren angezeigt.Im Beispiel unten ist nur das Motorfeld mit der Knotennummer 11 angeschlossen.
Zusätzlich werden im oberen Bereich (wie auf der Seite „Motore“) detaillierte Informationen zum jeweils ausgewählte Knoten angezeigt.



Dieser Dialog ermöglicht einen Beladungstest und die Bestimmung der min. Amplitude mit unterschiedlichen Parametern und Modi wie in Kap. 2.6.7 beschrieben.
Zusätzlich können, wie im Dialog-Fenster „Motoren“, alle Motoren bewegt und ein Dauertest durchgeführt werden.

- F2: alle Module werden in den Zustand PREOPERATIONAL gesetzt
  Alle Motoren werden gestoppt und stromlos geschaltet
- F3: alle Module werden in den Zustand OPERATIONAL gesetzt
  In diesem Zustand können die Motoren bewegt werden.
- F4: Die Anzeige wird gelöscht, d.h. die Modulanzeigen werden weiß hinterlegt.
- F7: Eichfunktion zu Bestimmung der min. Amplituden
- F8: Dauerlauf-Funktion (Dauertest)
- F9: Der Beladungstest wird gestartet. Die Anzeige wird zuvor gelöscht.

Bedienelemente:

- „Alle Knoten“: Alle Aktionen betreffen alle Knoten bzw. nur den aktuellen Knoten.
- „Anzeige“: Hier kann ausgewählt werden, ob die Ist-Drehzahl oder (zu Kontrollzwecken) die min. Amplitude in der 3. Zeile der Motortabellen angezeigt wird.

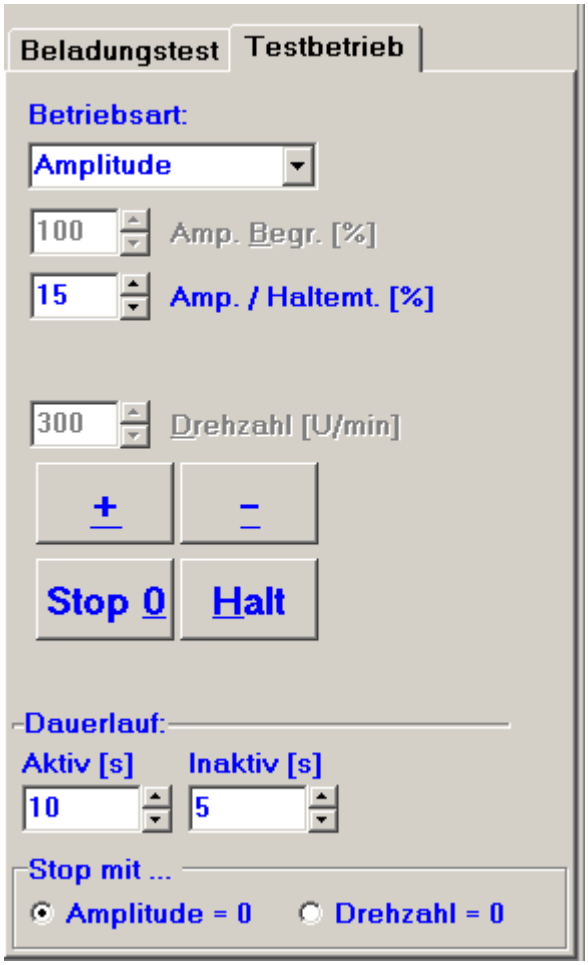
Register „Beladungstest“:

Hier können sämtliche Parameter für den Beladungstest und für die Eichfunktion eingegeben bzw. die relevanten SDOs gesetzt werden.

- Testmodus, Testzeit: Parameter für den Beladungstest bzw. die Eichfunktion
- Stillstand: min. Drehzahl für Stillstanderkennung (SDO 0x5100)
- Start bei [%] Startwert für die Eichfunktion (SDO 0x5101)

Gestartet wird der Beladungstest mit „F9“ und die Eichfunktion (Bestimmung der minimalen Amplitude) mit „F8“

Register „Testbetrieb“:



- Betriebsart, Eingabefelder für Drehzahl und Amplituden, +/-/Stop/Halt-Buttons:
  Diese entsprechen den Bedien-Elementen auf der Dialogseite „Motoren“, Details s. dort.

  Die Motoren können damit genauso bedeint werden, wie dort beschrieben.

Zusätzlich lassen sich die Motoren mit der ausgewählten Betriebsart im Dauerlauf betreiben. Der Dauerlauf ist zyklisch, d.h. die Motoren drehen abwechselnd in positive und negative Drehrichtung mit einer Pause dazwischen.

- Aktiv: Drehzeit
- Inaktiv: Pausenzeit
- Stop mit...
  Das Auswahlfeld bestimmt den Zustand während der Stillstandphase:
  - Amplitude = 0: keine Drehzahl-Regelung
  - Drehzahl = 0: mit Drehzahl-Regelung

3.6.1 F7 – Eichfunktion

Bestimmung der minimalen Amplitude, wie in Kap. 2.6.7.1 beschrieben:

Es wird das entsprechende RPDO an alle oder nur den aktuellen Knoten versandt:
→ Steuerwort/Sollwert: OperationDC, Amplitude aus dem Eingabefeld „Prozent“
→ Modus: aus dem Eingabefeld „Testmodus“
→ Intervall: aus dem Eingabefeld „Testzeit“

Die Eingabefelder „Stillstand bei U/min“ bzw.“Start bei [%]“ bestimmen die Stillstandserkennung und die Startamplitude für die Eichfunktion. Am Ende senden alle Module das TPDO7 mit den Motoren, für die die Minimal-Amplitude bestimmt werden konnte. Die Anzeigefelder werden dann entsprechend rot bzw. grün hinterlegt.

3.6.2 F9 – Beladungstest

Beladungstest wie in Kap. 2.6.7.2 beschrieben:

Es wird das entsprechende RPDO an alle oder nur den aktuellen Knoten versandt:
→ Steuerwort/Sollwert: OperationDC, Amplitude = 0%
→ Modus: aus dem Eingabefeld „Testmodus“
→ Intervall: aus dem Eingabefeld „Testzeit“

Am Ende senden alle Module das TPDO7 mit den Motoren, die sich bewegt haben (> UMin0), d.h. auf denen keine Gegenstände stehen. Die Anzeigefelder werden dann entsprechend rot bzw. grün hinterlegt.

3.6.3 F8 – Dauerlauf

Die Motoren werden in der ausgewählten Betriebsart zyklisch in positiver und negativer Richtung verfahren.
Im Beispiel oben drehen sich die Motoren mit eine PWM-Amplitude von 15% abwechselnd für 10 sec in positiver und negativer Richtung mit jeweils 5 sec Pause dazwischen.

Hinweis: *Alle Parameter (Betriebsart, Amplitude, Drehzahl) können während des Dauerlaufs geändert werden, werden jedoch erst im nächsten Zyklus berücksichtigt.*

3.7 Dialog „Stress-Test“

Der Dialog kann von folgenden Seiten über die Funktionstaste F9 aufgerufen werden:

- Seite „Start“
- Seite „Motore“
- Seite „Knoten“

Testprinzip:

Die 3 Motoren einer Motorspalte werden jeweils zu 3er-Gruppen zusammengefasst; der Test läuft also parallel in den sieben 3er-Gruppen des Moduls.
Innerhalb einer 3er-Gruppe werden verschiedene Test-Zyklen ausgeführt. Dazu werden die 3 Motoren einer Gruppe mit Gummi-Ringen zusammengekoppelt.

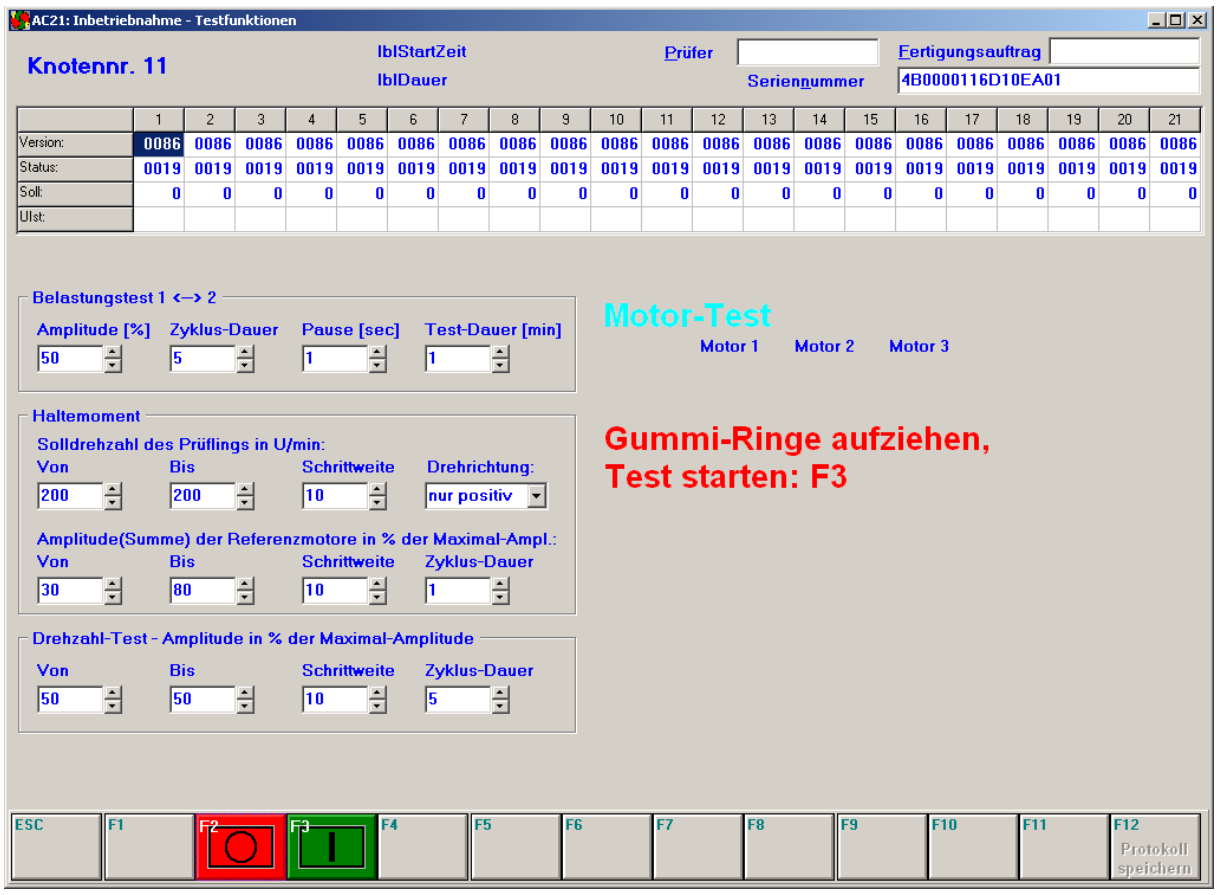
Der Test selbst läuft weitgehend automatisch ab; der Bediener muß zu Beginn je 3 Motoren mit 2 Gummi-Ringen verbinden. Vor einem abschließenden Testlauf müssen die Gummi-Ringe wieder entfernt werden und am Ende muß manuell die Motortemperatur gemessen und eingetragen werden.

Auswertung

Die Auswertung der einzelnen Tests erfolgt nach empirisch gewonnenen Kriterien. Diese sind im Programm fest vorgegeben und so bemessen und gewichtet, daß ein durchschnittlicher Motor einen Qualitätsindex von 100% erreicht. Die Bewertung in % wird am Ende des Tests ausgegeben, so daß die Qualität eine Motors entsprechend beurteilt werden kann.

***Hinweis**:* *Neben den Ergebnissen dieser mechanischen Tests werden auch die Messkurven der Motoren nach messtechnischen Kriterien (Amplitude, X-/Y-Offset) bewertet und haben so Einfluss auf das Gesamt-Ergebnis.*

3.7.1 Startbildschirm



Funktionstasten:

- F2: Abbruch des Tests
- F3: Start des Tests
- F12: Protokoll speichern + Beenden des Tests; Anzeige der Daten

Anzeigeelemente:

- Knotennr. Nummer des CAN-Moduls
- Start / Dauer: Startzeit und bisherige Dauer des testzyklus
- Motor- Tabelle: Für jeden Motor des ausgewählten Moduls werden folgende Daten angezeigt:
  - Version: Versionsnummer der Firmware im Motor-Prozessor
  - Status: Statuswort, s. Kap. 2.4.1
  - Soll: Soll-Drehzahl
  - UIst: aktuelle Drehzahl
- Motor1 / Motor2 / Motor3: Anzeige des gerade aktiven Tests und aktuelle Vorgaben für die Motoren der einzelnen 3er-Motorgruppen
- Infofeld: Bedien-Hinweise **in roter Schrift**.
- Durchschnittstemperatur: Am Ende des Tests muß manuell die Motortemperatur gemessen und hier eingetragen werden.
- Protokoll-Optionen: Am Ende des Tests werden die Ergebnisse in einer Protokoll-Datei gespeichert und ein Protokoll angezeigt bzw. ausgedruckt. Je nachdem, welche Variante gewählt wird, werden die Ergebnisse der Tests unterschiedlich detailliert dargestellt.
  - 1 Diagramm/Seite: Für jedes Diagramm wird 1 DIN-A4-Seit im Querformat ausgegeben

- 3 Diagramme/Seite: je 3 Diagramme werden auf einer DIN-A4-Seite im Hochformat ausgegeben
- alles auf 1 Seite: Die Diagramme werden so verkleinert, daß alle auf eine DIN-A4-Seite im Querformat passen.

***Hinweis**:* *Die Eingabefelder für die Durchschnittstemperatur und die Protokoll-Optionen werden erst am Testende eingeblendet.*

Bedienelemente:

In den Eingabe-Feldern links können die einzelnen Testfunktionen in Grenzen parametriert werden; Details s. folgende Kapitel.
Die Vorgaben sind bereits sinnvoll gewählt, so daß normalerweise keine Änderungen notwendig sind

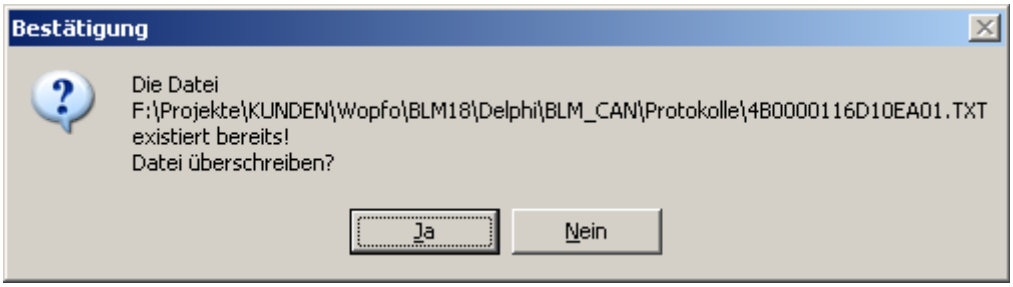
***Hinweis**:* *Sollten Änderungen notwendig sein, so müssen sie VOR dem Start des Test-Zyklus erfolgen. Änderungen während des Zyklus werden nicht oder nicht vollständig berücksichtigt!*

3.7.2 Start des Tests

Mit der Funktionstaste F3 wird der Test gestartet. Bevor der Test beginnt, wird überprüft, ob die Daten zum Modul (Seriennr. und Auftragsnr.) sowie der Name des Prüfers eingegeben wurden.

Als Seriennummer wird standardmäßig die Seriennummer des Moduls vorgegeben. Diese kann jedoch beliebig geändert werden (nicht empfohlen).

Protokolldaten werden als TXT-Datei im Protokoll-Verzeichnis (s. Kap. 3.3) abgelegt. Als Dateiname dient der Eintrag im Eingabefeld „Seriennummer“. Ist diese Datei schon vorhanden, so erscheint eine Warnung:



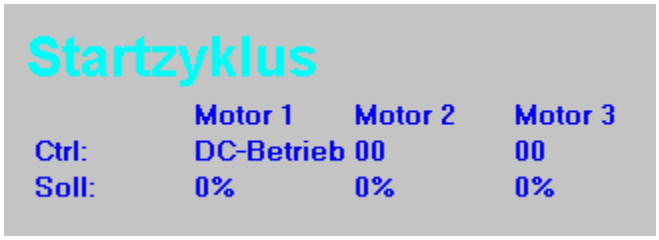
***Hinweis zur Seriennummer**:* *Die im Hauptfenster angezeigten Seriennummern der einzelnen Module sind in der Modul-Hardware fest vorgegeben und nicht änderbar. Im Feld „Seriennummer“ kann dagegen eine beliebige Kennzeichnung eingegeben werden, die kundeneigenen Dokumentations- bzw. Verwaltungszwecken dienen kann und nicht im AC21-Modul selbst gespeichert wird.*

3.7.3 Startzyklus

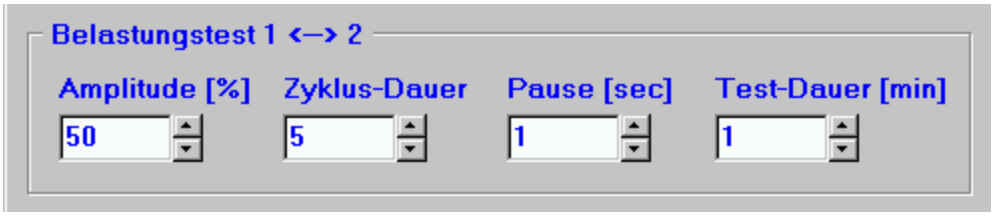
Nachdem die Gummi-Ringe aufgezogen wurden, wird der Test-Zyklus mit „F3“ gestartet.

Für die nächsten Schritte ist keine weitere Benutzereingabe erforderlich.

Zu Beginn werden zunächst alle Motoren in einen definierten Grundzustand gebracht, bevor der eigentliche Test beginnt



3.7.4 Belastungstest



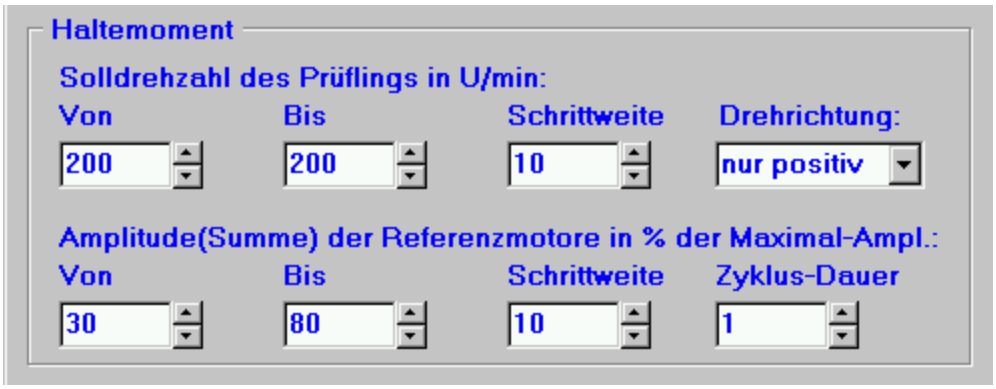
Beim Belastungstest drehen alle Motoren mit vorgegebener Amplitude in der Betriebsart „DC-Betrieb“. Jeweils ein Motor einer 3er-Gruppe dreht dabei in Gegenrichtung. Während des Tests wird die Drehrichtung innerhalb der 3er-Gruppen reihum gewechselt.

Test-Parameter

- Amplitude: PWM-Amplitude in % der Maximal-Amplitude
- Zyklus-Dauer Drehrichtung: Dauer eines Drehzyklus in sec, danach erfolgt der Wechsel der
- Pause: Pause zwischen dem Drehrichtungswechsel
- Test-Dauer: Testschritt: Dauer des gesamten Tests in min; danach folgt der nächste

Hier dreht gerade der 2. (mittlere) Motor der 3er-Gruppen mit –50% Amplitude gegen die beiden äußeren Motoren:

3.7.5 Haltemoment-Test



Beim Test des Haltemoments dreht jeweils ein Motor (Prüfling) der 3er-Gruppe mit vorgegebener Drehzahl, während die beiden anderen Motoren (Referenzmotoren) mit vorgegebener Amplitude (DC-Betrieb) in Gegenrichtung drehen; die PWM-Amplitude wird in Intervallen schrittweise erhöht. In jedem Zyklus wird die Zuordnung der Motoren reihum getauscht. Ist die Vertauschung komplett, so ist der Test beendet und es folgt der nächste Testschritt.

Test-Parameter

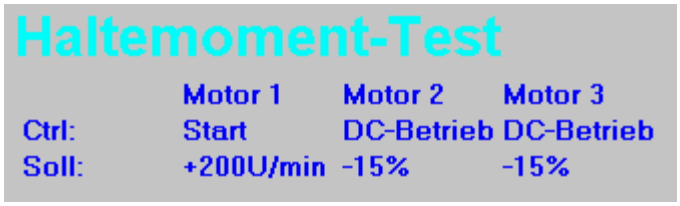
Solldrehzahl des Prüflings

- von / bis: Drehzahlbereich des Prüflings
- Schrittweite: Die Prüfung kann über einen ganzen Drehzahl-Bereich erfolgen; nach jedem Durchgang wird die Drehzahl um die angegebene Schrittweite erhöht.
- Drehrichtung: Die Prüfung kann in positiver, negativer oder in beiden Drehrichtungen des Prüfling erfolgen.

Amplitude der Referenzmotoren

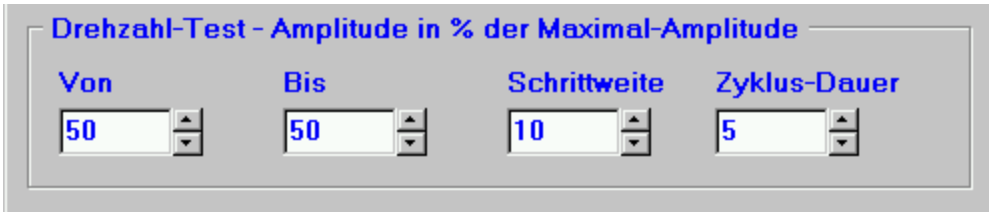
- von / bis: Amplitudenbereich für die Referenzmotoren; angegeben ist jeweils die Gesamt-Amplitude der PWM, d.h. jeder der beiden Referenzmotoren wird mit der Hälfte der angegebenen Amplitude bestromt.
- Schrittweite: Die Prüfung kann über einen ganzen Amplituden-Bereich erfolgen; nach jedem Durchgang wird die Amplitude um die angegebene Schrittweite erhöht.
- Zyklusdauer: Nach der hier angegebenen Zeit in sec wird die Amplitude um die angegebene Schrittweite erhöht.

Hier dreht gerade jeweils der 1.Motor einer 3er-Gruppe mit 200 U/min gegen die beiden anderen, die mit -15% Amplitude im DC-Betrieb in Gegenrichtung drehen.

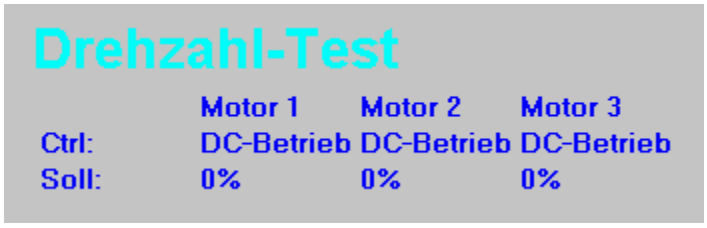


3.7.6 Drehzahl-Test

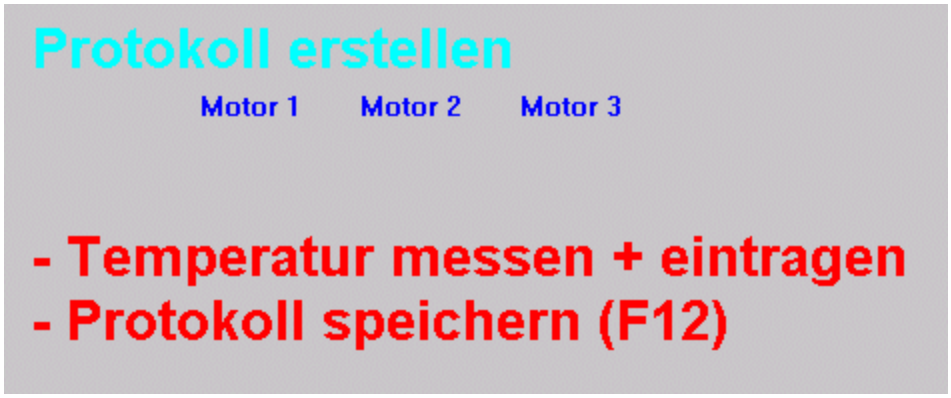
Vor dem abschließenden Drehzahl-Test ist wieder ein Benutzer-Eingriff notwendig: Die Gummi-Ringe müssen jetzt entfernt werden. Mit der Funktionstaste „F3“ wird der Test fortgesetzt.



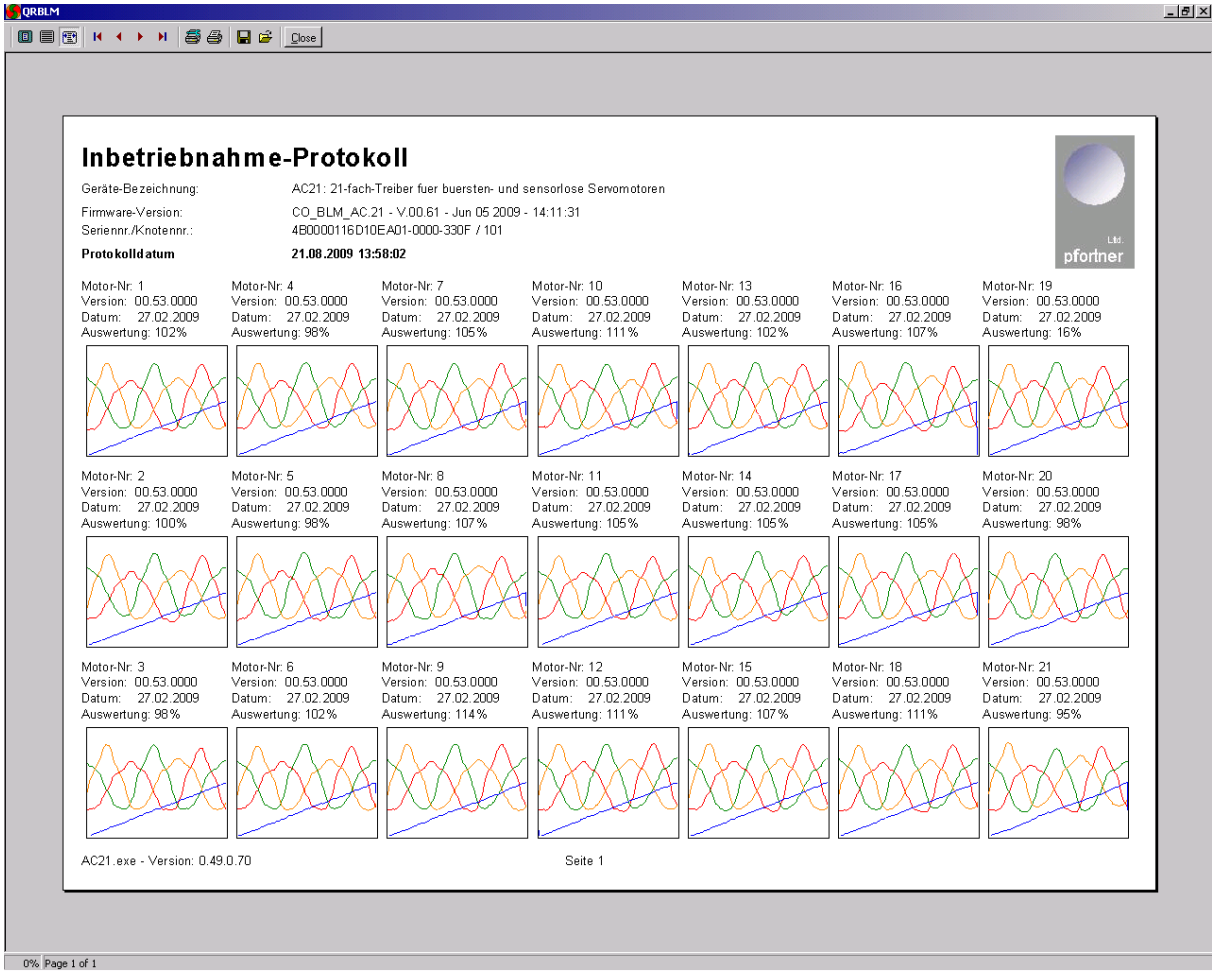
Beim Drehzahl-Test werden die Motoren im DC-Betrieb mit vorgegebener Amplitude in beide Richtungen gedreht und die Drehzahl gemessen. Die Drehzahl sollte in beiden Richtungen in etwa identisch sein. Wird der Test für ein Modul regelmäßig durchgeführt und die Ergebnisse verglichen, so liefert er außerdem Aussagen über die Langzeitstabilität und Qualität / Beanspruchung des Motors und der Lager.



3.7.7 Protokoll



Der letzte Schritt ist das Speichern des Protokolls; zusätzlich wird die Motor-Temperatur erfasst, was allerdings manuell geschehen muß. Die gemessene Temperatur muß ins entsprechende Eingabefeld eingegeben werden.
Mit der Funktionstaste „F12“ wird das Protokoll schließlich gespeichert, das Dialog-Fenster geschlossen und anschließend das Protokoll angezeigt.



4 Verzeichnisse

4.1 Abbildungsverzeichnis

4.2 Tabellenverzeichnis