CAN-CBM-REL4

CANopen-
Software-Handbuch

Handbuch-Datei:	I:\texte\Doku\MANUALS\CAN\Cbm\REL4\Deutsch\CBREL4_12S.ma9
Datum der Druck-vorlagenherstellung:	07.10.2002

Beschriebene Software:	CANopen
Software Revision:	1.3

Änderungen in den Kapiteln

Die hier aufgeführten Änderungen im Anwenderhandbuch betreffen sowohl Änderungen in der **Firmware** als auch reine Änderungen in der **Beschreibung** der Sachverhalte.

Kapitel	Änderungen gegenüber Vorversion
2.	Neue Objekte eingefügt: 1002_h, 1003_h, 1005_h, 1010_h, 1011_h, 1014_h, 1017_h, 1018_h, 1020_h, 1029_h Objekte die entfallen: $100B_h$, $100E_h$
3.	Neue Objekte eingefügt: 6202_h, 6206_h, 6207_h, 6208_h

Der Inhalt dieses Handbuches wurde mit größter Sorgfalt erarbeitet und geprüft. **esd** übernimmt jedoch keine Verantwortung für Schäden, die aus Fehlern in der Dokumentation resultieren könnten. Insbesondere Beschreibungen und technische Daten sind keine zugesicherten Eigenschaften im rechtlichen Sinne.

esd hat das Recht, Änderungen am beschriebenen Produkt oder an der Dokumentation ohne vorherige Ankündigung vorzunehmen, wenn sie aus Gründen der Zuverlässigkeit oder Qualitätssicherung vorgenommen werden oder dem technischen Fortschritt dienen.

esd electronic system design gmbh
Vahrenwalder Str. 207
30165 Hannover

Tel.: 0511/372 98-0
FAX : 0511/372 98-68
E-Mail: info@esd-electronics.com
Internet: www.esd-electronics.com

Inhalt

Seite

1. Allgemeines

Dieses Kapitel enthält neben Grundlagen zum Thema CANopen die wichtigsten Informationen über die implementierten Funktionen.

Eine komplette CANopen-Beschreibung läge außerhalb des Rahmens dieses Handbuchs.
Weitergehende Informationen sind daher den CAL / CANopen - Dokumentationen ‘CiA Draft Standard 201...207’, ‘CiA Draft Standard 301’ und ‘CiA Draft Standard Proposal 401’ zu entnehmen.

1.1 Begriffsdefinition

COB ...	Communication Object	
Emergency-Id...	Emergency Data Object	Notfalldaten
n/a ...	not available	
NMT...	Network Management (Master)	Netzwerkmanagement
Rx...	receive	empfangen
SDO...	Service Data Object	Service-Daten (Parameter)
SW ...	Software	
Sync...	Sync(frame) Telegramm	Synchronisations-Identifier
tbd ...	to be defined	Daten müssen noch festgelegt werden
Tx...	transmit	senden

PDOs (Process Data Objects)

Die PDOs dienen zur Übertragung der Prozeßdaten.
Im ‘Receive’-PDO werden Prozeßdaten vom CAN-CBM-REL4-Modul empfangen.
Im ‘Transmit’-PDO sendet das CAN-CBM-REL4-Modul (welches im CANopen-Netz selbst immer SLAVE ist !) Daten auf dem CANopen-Netz.

Die asynchrone Übertragungsart der PDOs wird durch den ‘PDO transmission type’ im Kommunikationsparameter des entsprechenden Prozeßdatenobjekts definiert.

1.2 NMT-Boot-up

Das CAN-CBM-REL4-Modul kann mit dem im CiA-Draft Standard 301 in Kapitel 8.3 beschriebenen ‘Minimum Capability Device’ - Boot-up initialisiert werden.

Im einfachsten Fall reicht nach dem Einschalten ein Telegramm zum Umschalten aus dem Zustand *Pre-Operational* in den Zustand *Operational*. Dazu ist an den CAN-Identifier ‘0000_h’ z. B. das 2-Byte-Telegramm ‘01_h’, ‘00_h’ zu senden (= Start Remote Node all Devices).

1.3 CANopen-Objektverzeichnis

Das Objektverzeichnis ist im wesentlichen eine (sortierte) Gruppierung von Objekten, auf die über das Netzwerk zugegriffen werden kann. Jedes Objekt in diesem Verzeichnis wird mit einem 16-Bit-Index adressiert. In den Objektverzeichnissen wird der Index in hexadezimaler Form angegeben.

Der Index kann ein 16-Bit-Parameter nach der CANopen-Spezifikation (CiA-Draft DS-301) oder ein herstellerspezifischer Code sein. Anhand der höherwertigen Bits des Index wird festgelegt, zu welcher Objektklasse der Parameter gehört.

Zum Objektverzeichnis gehören unter anderem:

Index [HEX]	Objekt	Beispiel
0001 ... 009F	Definition von Datentypen	
1000 ... 1FFF	Communication Profile Area	1001 = Fehler-Register
2000 ... 5FFF	Manufacturer Specific Profile Area	
6000 ... 9FFF	Standardised Device Profile Area	nach Anwendungsprofil DS-401
A000 ... FFFF	reserviert	

1.4 Kommunikationsparameter der PDOs

Die Kommunikationsparameter der PDOs (nach DS-301, Kap.10.1.2) werden als SDO (Service Data Objects) auf der ID **'600_h h + Node-ID**' übertragen (Request). Der Empfänger quittiert die Parameter auf der ID '**580_h + Node-ID**' (Response).
Die **Node-ID** (Modul-Nr.) wird über die Kodierschalter (SW110, SW111) eingestellt. Die möglichen Einstellungen sind im Hardware-Handbuch des CAN-CBM-REL4-Moduls ausführlich beschrieben.

1.4.1 Zugriff auf die Kommunikationsparameter über SDO-Telegramme

Die SDOs (Service-Daten-Objekte) dienen dem Zugriff auf das Objektverzeichnis eines Gerätes.
Ein SDO stellt daher einen 'Kanal' zum Zugriff auf die Parameter des Gerätes dar. Der Zugriff über diesen Kanal ist beim CAN-CBM-REL4-Modul im Zustand *Operational* und *Preoperational* möglich.

Dieses Kapitel gibt nicht alle möglichen, sondern nur einige wichtige Zugriffsarten für das CAN-CBM-REL4-Modul wieder.
Die Definitionen für die Zugriffsarten können der CiA DS-202-2 (CMS-Protocol Specification, Kap. 6: CMS Multiplexed Domain Protocols) entnommen werden.

Ein SDO ist wie folgt aufgebaut:

Identifier	Befehls-code	Index (low)	Index (high)	Subindex	LSB	Datenfeld	MSB

Beispiel :
Abfrage des Gerätetyps (Device type) für das Modul Nr. 19 (13_h)
Request:

600_h + Node-ID = 613_h	40_h (read)	00_h	10_h (Index = 1000_h) (Receive-PDO-Comm-Para)	00_h	00_h	00_h	00_h	00_h

Response:

580_h + Node-ID = 593_h	43_h (read)	00_h	10_h (Index = 1000_h) (Receive-PDO-Comm-Para)	00_h	91_h	01_h	02_h	00_h (Device type = 0002.0191_h)

Der Wert des Device type für das Modul Nr. 19 (13_h) beträgt: 0002.0191_h

Identifier
Die Parameter werden auf der ID '**580_h + Node-ID**' übertragen (Request).
Der Empfänger quittiert die Parameter auf der ID '**580_h + Node-ID**' (Response).
Befehlscode
Der gesendete Befehlscode setzt sich unter anderem aus dem Command Specifier und der Länge zusammen. Häufig benötigte Kombinationen sind z.B.:

$40_h = 64_d$: Read Request, d.h. ein Parameter soll gelesen werden
$23_h = 35_d$: Write Request mit 32 Bit Daten, d.h. ein Parameter soll gesetzt werden.

Das CAN-CBM-REL4-Modul antwortet auf jedes empfangene Telegramm mit einem Antworttelegramm. Das Antworttelegramm kann folgende Befehlscodes enthalten:

$43_h = 67_d$: Read Response, dieses Telegramm enthält den gewünschten Parameter
$60_h = 96_d$: Write Response, d.h. ein Parameter wurde erfolgreich gesetzt
$80_h = 128_d$: Error Response, d.h. das CAN-CBM-REL4-Modul meldet einen Kommunikationsfehler.

Häufig verwendete Befehlscodes

Die folgende Tabelle gibt eine Übersicht über häufig verwendete Befehlscodes. Die Kommando-Frames müssen immer 8 Datenbytes enthalten. Hinweise zur Syntax und weitere Befehlscodes sind in der CiA DS-202-2 (CMS-Protocol Specification, Kap. 6: CMS Multiplexed Domain Protocols) nachzulesen.

Kommando	**für Anzahl Datenbytes**	**Befehlscode [HEX]**
Write Request (Initiate Domain Download)	1 2 3 4	2F 2B 27 23
Write Response (Initiate Domain Download)	-	60
Read Request (Initiate Domain Upload)	-	40
Read Response (Initiate Domain Upload)	1 2 3 4	4F 4B 47 43
Error Response (Abort Domain Transfer)	-	80

Fehlercodes des SDO-Domain-Transfers

Die folgenden Fehlercodes können zur Anwendung kommen (gemäß CiA Work Draft WD-301):

Fehlercode [Hex]	CAN-CBM-REL4-Bezeichnung	Erläuterung
0x05030000	SDO_TOGGLE_BIT_NOT_ALTERNATED	Toggle Bit nicht gewechselt
0x05040001	SDO_CS_UNKNOWN	falscher Command Specifier
0x06010002	SDO_READ_ONLY	Schreibzugriff ist hier falsch
0x06020000	SDO_WRONG_INDEX	falscher Index
0x06060000	SDO_HARWARE_ERROR	kein Zugriff durch Hardware Fehler
0x06070010	SDO_WRONG_LENGTH	falsche Anzahl Daten-Bytes
0x06070012	SDO_PARA_TO_LONG	Länge des Service-Parameters zu groß
0x06070013	SDO_PARA_TO_SHORT	Länge des Service-Parameters zu klein
0x06090011	SDO_WRONG_SUBIND	falscher Sub-Index
0x06090030	SDO_VALUE_RANGE_EXEEDED	-
0x08000000	SDO_OTHER_ERROR	nicht definierte Fehlerursache
0x08000020	SDO_DATA_CANNOT_STORED	Daten können nicht transferiert oder gespeichert werden

Index, Sub-Index
Der Index und der Sub-Index werden in dem folgenden Kapitel ‘Communication Profile Area’ beschrieben.

Datenfeld
Das maximal 4 Byte lange Datenfeld ist grundsätzlich nach der Regel ‘niederwertiges Byte zuerst, höherwertiges Byte zuletzt’ aufgebaut. Dabei steht das niederwertige Byte **immer** in ‘Data 1’.
Bei 16-Bit-Werten steht das höchstwertige Byte (Bits 8...15) in ‘Data 2’, und bei 32-Bit-Werten steht das MSB (Bits 24...31) in ‘Data 4’.

Diese Seite ist bewußt unbedruckt.

2. Communication Profile Area

2.1 Verwendete Bezeichnungen und Abkürzungen

Folgende Bezeichnungen werden in den Tabellen zur Beschreibung der Kommunikationsparameter verwendet:

PDO-Mapping	für diesen Sub-Index des PDOs ist PDO-Mapping möglich
Zugriff	zugelassene Zugriffsarten auf diesen Parameter ro... read_only Dieser Parameter kann nur gelesen werden. Schreibzugriffe führen zu einer Fehlermeldung. const...constant Dieser Parameter kann vom Anwender nicht verändert werden. Er ist Lesbar. Schreibzugriffe führen zu einer Fehlermeldung. rw... read&write Dieser Parameter kann gelesen oder gesetzt werden.
Wertebereich	Wertebereich des Parameters
Default-Value	Grundeinstellung des Parameters bei Auslieferung des Moduls
Name	Name und Kurzbeschreibung des Parameters

2.2 Übersicht der Kommunikationsparameter

Das Format der Kommunikationsparameter kann der CiA DS-301, Kap. 10.3 entnommen werden. Besonderheiten der Implementierung der CAN-CBM-REL4 sind in den anschließenden Kapiteln dieses Handbuchs aufgeführt.
Das Modul unterstützt nur die in der folgenden Tabelle aufgeführten Kommunikationsparameter.

Index [HEX]	Name	Subindex	Typ	Zugriff	Default
1000	Device Type	-	Unsigned32	ro	00020191_h
1001	Error Register	-	Unsigned8	ro	Error-Code
1002	Status Register	-	Unsigned32	ro	0
1003	Pre-Defined-Error-Field	16	Unsigned32	rw	Index 0
1005	COB-ID-Sync	-	Unsigned32	rw	80_h
1008	Manufacturer's Device Name 1*)	-	Visible String 1*)	ro	CAN-CBM-REL4 3*)
1009	Manufacturer's Hardware Version 1*)	-	Visible String	ro	1.10
100A	Manufacturer's Software Version 1*)	-	Visible String 2*)	ro	1.10
100C	Guard Time	-	Unsigned16	rw	0 Sek.
100D	Life Time Factor	-	Unsigned8	rw	0
1010	Store Parameter	0, 1, 2, 3	Unsigned32	rw	--
1011	Restore Parameter	0, 1, 2, 3	Unsigned32	rw	--
1014	COB-ID Emergency Object	-	Unsigned 32	rw	80_h + Node-ID
1017	Producer Heartbeat Time	-	Unsigned16	rw	0 ms
1018	Identity Object	0, 1	Unsigned32	ro	00000017_h
1020	Verify Configuration	0, 1, 2	Unsigned32	rw	0
1029	Error Behaviour	0, 1	Unsigned8	rw	0
1400	Receive PDO Communication Parameter	0, 1, 2	PDOCommPar	rw	--
1600	Receive PDO Mapping Parameter	0, 1, 2	PDOMapping	ro	--
1800	Transmit PDO Communication Parameter	0,1,2	PDOCommPar	rw	--
1A00	Transmit PDO Mapping Parameter	0,1	PDOMapping	ro	--

ro - Read Only, rw - Read/Write
1*) Achtung: Länge > 4 Bytes, d.h. Upload aufwendiger als bei anderen Parametern (siehe S. 17)
2*) abhängig vom Revisionsstand der Software
3*) abhängig von der Default-Konfiguration

2.3 Beschreibung der Parameter

2.3.1 Device Type 1000_h

INDEX	**1000_h**
Name	*device type*
Data Type	unsigned 32
Default Value	No

Der Wert des *device type* beträgt: 0002.0191_h (Digital Output: 0002_h
Digital Profile Number: 0191_h)

Beispiel: Auslesen des Device Type

Der CANopen Master sendet unter dem Identifier '603_h' (600_h + Node-ID) den Read Request an das CAN-CBM-REL4-Modul mit der Modul-Nr. 3 (Node-ID=3_h):

ID	RTR	LEN	DATA							
			1	2	3	4	5	6	7	8
603_h	0_h	8_h	40_h	00_h	10_h	00_h	00_h	00_h	00_h	00_h
			Read Request	Index=1000_h		Sub Index				

Das CAN-CBM-REL4-Modul Nr. 3 antwortet dem Master anhand der Read Response unter dem Identifier '583h' (580_h + Node-ID) mit dem Wert des Device Type:

ID	RTR	LEN	DATA							
			1	2	3	4	5	6	7	8
583_h	0_h	8_h	43_h	00_h	10_h	00_h	91_h	01_h	02_h	00_h
			Read Response	Index=1000_h		Sub Index	dig. Profile Nr.191		Digital Output	

ausgegebener Wert des Device Type: 0002.0191_h

Das Datenfeld ist grundsätzlich nach der Regel 'niederwertiges Byte zuerst, höherwertiges Byte zuletzt' aufgebaut (siehe Kapitel 1.4.1, Datenfeld).

2.3.2 Error Register 1001_h

Das CAN-CBM-REL4-Modul nutzt das Error-Register, um Fehlermeldungen anzuzeigen.

INDEX	**1001_h**
Name	*error register*
Data Type	unsigned 8
Default Value	No

Folgende Bits des Error-Registers werden zur Zeit unterstützt:

Bit	7	6	5	4	3	2	1	0
Bedeutung	-	-	-	-	-	-	-	*generic*

Die nicht unterstützten Bits (-) werden immer als ‘0’ zurückgegeben.

generic	Fehler
0	kein Fehler
1	generic Error

Folgende Meldungen sind möglich:
00_h keine Fehler
01_h generic Error

2.3.3 Manufacturer Status Register 1002_h

INDEX	**1002_h**
Name	*manufacturer status register*
Data Type	unsigned 32
Default Value	No

Die Bits des Status-Registers sind wie folgt belegt:

Register-Bit (Assembler)	Beschreibung	Pegelzuordnung	
D25-D31	nicht belegt	0	(immer ‘0’)
D24	I²C-Error	0 1	kein I²C-Error I²C-Error
D3-D23	nicht belegt	0	(immer ‘0’)
D2	Watchdog-Reset	0 1	kein Reset Watchdog-Reset wurde durchgeführt
D1	Nodeguard-Enabled	0 1	Nodeguarding ausgeschaltet Nodeguarding eingeschaltet
D0	Heartbeat-Enabled	0 1	Heartbeat-Funktion ausgeschaltet Heartbeat-Funktion eingeschaltet

2.3.4 Pre-defined Error Field 1003_h

INDEX	**1003_h**
Name	*pre-defined error field*
Data Type	unsigned 32
Default Value	No

Im *pre-defined error field* wird eine Liste der zuletzt aufgetretenen Fehler gespeichert. Der Subindex 0 enthält die aktuelle Anzahl der in der Liste gespeicherten Fehler. Unter Subindex 1 wird der zuletzt aufgetretene Fehler abgelegt. Tritt ein neuer Fehler auf, so wird der vorhergehende Fehler auf Subindex 2 gespeichert und der neue Fehler unter Subindex 1 usw.. So entsteht eine Liste mit der Fehler-Historie.

Der Fehlerspeicher ist wie ein Ringspeicher aufgebaut. Ist er gefüllt, wird der älteste Eintrag gelöscht um Platz für den aktuellen Eintrag zu schaffen.

Dieses Modul unterstützt maximal 16 Fehlereinträge. Beim Eintreten des 17. Fehlers wird der älteste Fehlereintrag gelöscht. Zum Löschen der gesamten Fehler-Liste ist der Subindex ‘0’ auf ‘0’ zu setzen. Dies ist der einzig zulässige Schreibzugriff auf das Objekt.

Mit jedem neuen Eintrag in die Liste sendet das Modul ein **Emergency-Frame**, um den Fehler mitzuteilen.

Index [Hex]	Subindex [Dez]	Beschreibung	Wertebereich [Hex]	Default	Datentyp	R/W
1003	0	*no_of_errors_in_list*	0, 1...10	-	unsigned 8	rw
	1	*error-code n*	0...FFFFFFFF	-	unsigned 32	ro
	2	*error-code (n-1)*	0...FFFFFFFF	-	unsigned 32	ro
	:	*:*	:	:	:	ro
	16	*error-code (n-15)*	0...FFFFFFFF	-	unsigned 32	ro

Bedeutung der Variablen:

no_of_errors_in_list - enthält die aktuelle Anzahl der in der Liste eingetragenen Fehler-Codes
n = Nummer des zuletzt aufgetretenen Fehlers
- zum Löschen der Fehlerliste ist diese Variable auf ‘0’ zu setzen
- ist *no_of_errors_in_list* $\neq 0$, so wird das Error-Register (Object 1001_h) gesetzt

error-code x Der 32-Bit lange Fehler-Code setzt sich aus dem im DS-301, Tabelle 21 aufgeführten CANopen-Emergency-Error-Code und den von esd definierten Fehler-Codes (Manufacturer-Specific Error Field) zusammen.

Bit:	31 16	15 0
Inhalt:	*manufacturer-specific error field*	*emergency-error-code*

manufacturer-specific error field: bei CAN-CBM-REL4 immer ‘00’

emergency-error-code: Es werden nur die folgenden Fehler-Codes unterstützt:
0000_h - Emergency-Error-Reset or no Error
1000_h - Emergency Generic Error
(Fehler des I²C-EEPROMs)

Emergency-Frame

Die Daten des von der CAN-CBM-REL4 gesendeten Emergency-Frames sind wie folgt aufgebaut:

Byte:	0	1	2	3	4	5	6	7
Inhalt:	*emergency-error-code* (siehe oben)		*error-register* 1001_h	*no_of_errors_in_list* $1003,01_h$	*manufacturer status register* 1002_h			

2.3.5 COB-ID of SYNC-Message 1005_h

INDEX	**1005_h**
Name	*COB-ID SYNC message*
Data Type	unsigned 32
Default Value	80_h

Struktur des Parameters:

Bit-No.	Wert	Bedeutung
31 (MSB)	-	do not care
30	0/1	0: keine SYNC Erzeugung 1: Modul erzeugt SYNC
29	0	immer 0, da 11-Bit-ID
28...11	0	immer 0, da keine 29-Bit-ID unterstützt werden
10...0 (LSB)	x	Bit 0...10 der SYNC-COB-ID

Der Identifier kann Werte zwischen $0...7FF_h$ annehmen.

2.3.6 Manufacturer's Device Name 1008_h

INDEX	**1008_h**
Name	*manufacturer's device name*
Data Type	visible string
Default Value	string: 'CAN-CBM-REL4'

Der Datenaustausch beim Upload des Strings muß, wie in der folgenden Tabelle dargestellt, ablaufen (alle Zahlenwerte hexadezimal). In den übergebenen Kommandos müssen immer 8 Datenbytes übertragen werden.

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Bemerkung
Client:	**40**	**08**	**10**	00	00	00	00	00	upload request index 1008_h, sub-index=0
CBM-REL4:	**41**	**08**	**10**	**00**	**0C**	00	00	00	upload ackn., len = 12
Client:	**60**	00	00	00	00	00	00	00	segment upload request
CBM-REL4:	**00**	**43**	**41**	**04**	**2D**	**43**	**42**	**4D**	'CAN-CBM', len = 7
Client:	**70**	00	00	00	00	00	00	00	-
CBM-REL4:	**15**	**2D**	**52**	**45**	**4C**	**34**	00	00	'-REL4', len = 5, ended

Eine ausführliche Beschreibung des Domain Uploads ist der CiA DS-202-2 (CMS-Protocol Specification) zu entnehmen.

2.3.7 Manufacturer's Hardware Version 1009_h

INDEX	**1009_h**
Name	*manufacturer's hardware version*
Data Type	visible string
Default Value	string: z.B. '1.1'

Das Lesen der Hardware-Version erfolgt ähnlich wie das Lesen des Manufacturer Device Names über das Domain Upload Protokoll. Eine ausführliche Beschreibung des Uploads ist der CiA DS-202-2 (CMS-Protocol Specification) zu entnehmen.

2.3.8 Manufacturer's Software Version $100A_h$

INDEX	**$100A_h$**
Name	*manufacturer's software version*
Data Type	visible string
Default Value	string: z.B.: '1.1'

Das Lesen der Software-Version erfolgt ähnlich wie das Lesen des Manufacturer Device Names über das Domain Upload Protokoll. Eine ausführliche Beschreibung des Uploads ist der CiA DS-202-2 (CMS-Protocol Specification) zu entnehmen.

2.3.9 Guard Time $100C_h$ und Life Time Factor $100D_h$

Das CAN-CBM-REL4-Modul unterstützt das Node-Guarding oder alternativ die Heartbeat-Funktion (siehe Seite 25)
Guard Time und Life Time Factor werden zusammen ausgewertet. Die Multiplikation beider Werte ergibt die Life Time. Die Guard Time wird in Millisekunden angegeben.

INDEX	**$100C_h$**
Name	*guard time*
Data Type	unsigned 16
Default Value	0 [ms]
Minimum Value	0
Maximum Value	$FFFF_h$ (65,535 s)

INDEX	**$100D_h$**
Name	*life time factor*
Data Type	unsigned 8
Default Value	0
Minimum Value	0
Maximum Value	FF_h

2.3.10 Node Guarding Identifier $100E_h$

Das Modul unterstützt nur 11-Bit-Identifier. Der Parameter kann nur gelesen werden.

INDEX	**$100E_h$**
Name	*node guarding identifier*
Data Type	unsigned 32
Default Value	700_h + Node-ID

Struktur des Parameters *node guarding identifier* :

Bit-No.	Wert	Bedeutung
31 (MSB) 30	-	reserved
29...11	0	immer 0, da keine 29-Bit-ID unterstützt werden
10...0 (LSB)	0 x	Bit 0...10 des Node Guarding Identifiers

2.3.11 Store Parameters 1010_h

Mit diesem Kommando werden die Parameter im EEPROM gespeichert. Es wird nur das Kommando ‘Save all Parameters’ unterstützt.

Beim Schreiben des Index muss die unten angegebene Byte-Folge gesendet werden. Das Lesen des Index gibt Informationen über die implementierten Speicher-Funktionen zurück (näheres hierzu siehe CiA DS-301).

INDEX	**1010_h**
Name	*store parameters*
Data Type	unsigned 32

Index [Hex]	Subindex	Beschreibung	Wertebereich	Datentyp	R/W
1010	0	*number_of_entries*	3_h	unsigned 8	ro
	1	*save_all_parameters*	no default, write: 65 76 61 73_h (= ASCII: ‘e’ ‘v’ ‘a’ ‘s’)	unsigned 32	rw
	2	*save_communication_parameter*	no default, write: 65 76 61 73_h (= ASCII: ‘e’ ‘v’ ‘a’ ‘s’)	unsigned 32	rw
	3	*save_application_parameter*	no default, write: 65 76 61 73_h (= ASCII: ‘e’ ‘v’ ‘a’ ‘s’)	unsigned 32	rw

Parameter die abgespeichert bzw. geladen werden können:

Communication Parameter:

- 1005_h *COB-ID Sync*
- $100C_h$ *Guard Time*
- $100D_h$ *Life Time Factor*
- 1017_h *Producer Heartbeat Time*
- 1020_h Verify Configuration: *Configuration_Date, Configuration_Time*
- 1029_h Error Behaviour Object: *Communication_Error*

Application Parameter:

- 6202_h Change Polarity Output 8-Bit: *Change_Polarity_Output*
- 6206_h Error Mode Output 8-Bit: *Error_Mode_Output*
- 6207_h Error Value Output 8-Bit: *Error_Value_Output*
- 6202_h Filter Mask Output 8-Bit: *Filter_Mask_Output*

2.3.12 Restore Default Parameters 1011_h

Mit diesem Kommando werden die bei der Auslieferung aktiven Default-Parameter wieder aktiviert. Alle individuellen Einstellungen, die im EEPROM gespeichert worden sind, gehen verloren. Es wird nur das Kommando 'Restore all Parameters' unterstützt.

Beim Schreiben des Index muss die unten angegebene Byte-Folge gesendet werden. Das Lesen des Index gibt Informationen über die implementierten Restore-Funktionen zurück (näheres hierzu siehe CiA DS-301).

INDEX	**1011_h**
Name	*restore default parameters*
Data Type	unsigned 32

Index [Hex]	Subindex	Beschreibung	Wertebereich	Datentyp	R/W
1011	0	*number_of_entries*	3	unsigned 8	ro
	1	*load_all_default_parameters*	no default, write: 64 61 65 $6C_h$ (= ASCII: 'd' 'a' 'o' 'l')	unsigned 32	rw
	2	*load_communication_parameter*	no default, write: 64 61 65 $6C_h$ (= ASCII: 'd' 'a' 'o' 'l')	unsigned 32	rw
	3	*load_application_parameter*	no default, write: 64 61 65 $6C_h$ (= ASCII: 'd' 'a' 'o' 'l')	unsigned 32	rw

Die Communication Parameter und die Application Parameter, die gespeichert, bzw. geladen werden können, sind auf Seite 22 aufgelistet.

2.3.13 COB_ID Emergency Object 1014_h

INDEX	**1014_h**
Name	*COB-ID emergency object*
Data Type	unsigned 32
Default Value	80_h + Node-ID

Dieses Objekt bestimmt die COB-ID des Emergency Objektes (EMCY).

Die Struktur des Objektes ist in der folgenden Tabelle beschrieben:

Bit-No.	Wert	Bedeutung
31 (MSB)	0/1	0: EMCY existiert / ist gültig 1: kein EMCY / EMCY ist nicht gültig
30	0	reserviert (immer 0)
29	0	immer 0, da 11-Bit ID
28...11	0	immer 0, da keine 29-Bit-ID unterstützt werden
10...0 (LSB)	x	Bit 0...10 des COB-ID

Der Identifier kann Werte zwischen 0...$7FF_h$ annehmen.

2.3.14 Producer Heartbeat Time 1017_h

INDEX	**1017_h**
Name	*producer heartbeat time*
Data Type	unsigned 16
Default Value	0 ms

Hier wird die Zeit eingetragen, mit der das CAN-CBM-REL4-Modul zyklisch einen Heartbeat-Frame auf dem Node-Guarding-ID sendet.

Wird für die Producer-Heartbeat-Time ein Wert größer Null eingesetzt, so ist sie aktiv und unterbindet das Node-/ Life-Guarding (siehe Seite 35).
Wird die Producer-Heartbeat-Time auf '0' gesetzt, so wird das Senden des Heartbeats durch dieses Modul beendet.

Zur gegenseitigen Überwachung der CANopen-Module (insbesondere zum Erkennen von Verbindungsausfällen) kann die Heartbeat-Funktion genutzt werden. Im Gegensatz zum Node Guarding/Life Guarding kommt die Heartbeat-Funktion ohne RTR-Frames aus.

Funktionsweise:
Ein Modul, der sog. Heartbeat-Producer sendet auf dem Node-Guarding-Identifier (siehe Object $100E_h$) zyklisch eine Heartbeat-Nachricht auf dem CAN-Bus. Ein oder mehrere Heartbeat-Consumer empfangen die Nachricht. Die Nachricht muss innerhalb der auf dem Heartbeat-Consumer gespeicherten Heartbeat-Time empfangen werden, sonst wird auf dem Heartbeat-Consumer-Modul ein Heartbeat-Event ausgelöst.

Jedes Modul kann Heartbeat-Producer und Heartbeat-Consumer sein. Es kann in einem CAN-Netz mehrere Heartbeat-Producer und mehrere Heartbeat-Consumer geben. Das CAN-CBM-REL4-Modul arbeitet ausschließlich als Heartbeat-Producer.

Index [Hex]	Subindex	Beschreibung	Wertebereich [Hex]	Default	Datentyp	R/W
1017	0	*producer-heartbeat_time*	0...FFFF	0 ms	unsigned 16	rw

producer-heartbeat_time Zykluszeit des Heartbeat-Producers zum Senden des Heartbeats auf dem Node-Guarding-ID (siehe Object $100E_h$).
Die Consumer-Heartbeat-Time der überwachenden Module muss immer größer sein als die Producer-Heartbeat-Time dieses Heartbeat-sendenden Moduls.

2.3.15 Identity Object 1018_h

INDEX	**1018_h**
Name	*identity object*
Data Type	unsigned 32
Default Value	No

Das Indentity Object enthält allgemeine Informationen zum CAN-Modul.

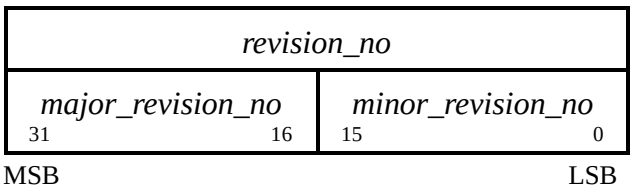
Index [Hex]	Subindex	Beschreibung	Wertebereich [Hex]	Default	Datentyp	R/W
1018	0	*no_of_entries*	1	1	unsigned 8	ro
	1	*vendor_id*	0...FFFFFFFF	0000 0017_h	unsigned 32	ro
	2	*product_code*	0...FFFFFFFF	2283 3010_h	unsigned32	ro
	3	*revision_number*	0...FFFFFFFF	'aktuelle SW-Version'	unsigned32	ro
	4	*serial_number*	0...FFFFFFFF	'aktuelle HW-Version'	unsigned32	ro

Bedeutung der Variablen:

vendor_id Diese Variable enthält die esd-Vendor-ID. Hier ist immer der Wert 00000017_h eingetragen.

product_code Hier ist die esd-Artikelnummer des Produkts abgelegt.
Beispiel:
Der Wert '2**2833.01**0_h' entspricht der Artikelnummer 'C.2833.01'.

revision_number Hier ist die Software-Version abgelegt. In den oberen zwei Bytes sind gemäß DS-301 die Revisionsnummern der wesentlichen (major) Änderungen aufgeführt und in den unteren zwei Bytes die Revisionsnummern einfacher Korrekturen oder Änderungen (minor).



serial_number

Hier wird die Seriennummer-Kennung der Hardware gelesen. Die ersten beiden Zeichen der Seriennummer sind Buchstaben, die das Fertigungslos kennzeichnen. Die folgenden Zeichen geben die eigentliche Seriennummer wieder.

In den beiden höherwertigen Bytes von *serial_no* sind die Buchstaben des Fertigungsloses kodiert. Sie enthalten jeweils den ASCII-Code des Buchstaben mit dem hochwertigsten Bit auf '1' gesetzt, um Buchstaben und Zahlen unterscheiden zu können:
(ASCII-Code) + 80_h = gelesenes_Byte

In den beiden niederwertigen Bytes ist die Nummer des Moduls als BCD-Wert enthalten.

Beispiel:
Wird der Wert 'C1C2 0105_h' gelesen, so entspricht dies der Hardware-Seriennummer-Kennung 'AB 0105'. Dieser Wert muß der aufgeklebten Seriennummer des Moduls entsprechen.

2.3.16 Verify Configuration 1020_h

INDEX	**1020_h**
Name	*verify configuration*
Data Type	unsigned 32
Default Value	No

In diesem Objekt können Datum und Tageszeit der letzten Konfiguration abgelegt werden, um zu einem späteren Zeitpunkt prüfen zu können, ob die gespeicherte Konfiguration der erwarteten entspricht.

Index [Hex]	Subindex	Beschreibung	Wertebereich [Hex]	Default	Datentyp	R/W
1020	0	*no_of_entries*	2	2	unsigned 8	ro
	1	*configuration_date*	0...FFFFFFFF	0	unsigned 32	rw
	2	*configuration_time*	0...FFFFFFFF	0	unsigned 32	rw

Bedeutung der Variablen:

configuration_date Datum der letzten Konfiguration des Moduls, angegeben in Tagen seit dem 01.01.1984.

configuration_time Zeit in ms seit Mitternacht am Tag der letzten Konfiguration.

2.3.17 Error Behaviour Object 1029_h

INDEX	**1029_h**
Name	*error behaviour object*
Data Type	Byte
Default Value	No

Tritt ein Error-Event ein (z.B. Heartbeat-Error), so wechselt das Modul in den Zustand, der in der Variablen *communication_error* definiert ist.

Index [Hex]	Subindex	Beschreibung	Wertebereich [Hex]	Default	Datentyp	R/W
1029	0	*no_of_error_classes*	1	1	Byte	ro
	1	*communication_error*	0, 1, 2	0	Byte	rw

Bedeutung der Variablen:

no_of_error_classes Anzahl der Fehler-Klassen (hier immer ‘1’)

communication_error 0 - pre-operational (nur, wenn aktueller Zustand = operational)
1 - no state change
2 - stopped

2.3.18 Receive PDO Communication Parameter 1400_h

Mit dem Objekt ‘Receive PDO Communication Parameter 1400_h’ werden die Eigenschaften eines Empfangs-PDOs (Rx-PDO) definiert.

INDEX	**1400_h**
Name	*receive PDO parameter*
Data Type	PDOCommPar

Index [Hex]	Subindex	Beschreibung	Wertebereich [Hex]	Default	Datentyp	R/W
1400	0	*no_of_entries*	2	2_h	unsigned 8	const
	1	*COB_ID used by PDO*	0...FFFFFFFF	200_h + Mod.-Nr	unsigned 32	ro
	2	*transmission type*	0...FF	255d	unsigned 8	rw

Wertebereich nach DS-301, Tabelle 10-6, 10-7.

2.3.19 Receive PDO Mapping Parameter 1600_h

Mit dem Objekt 'Receive PDO Mapping Parameter 1600_h' kann die Zuordnung der Empfangsdaten zu den Rx-PDOs verändert werden.

INDEX	**1600_h**
Name	*receive PDO mapping*
Data Type	PDO Mapping

Die folgende Tabelle zeigt die Belegung der Receive PDO Mapping Parameter für die Default-Konfiguration:

Index [Hex]	Subindex	Beschreibung	Wertebereich [Hex]	Default	Datentyp	R/W
1600	0	*no_of_entries*	1	1_h	unsigned 8	ro
	1	*object_to_be_mapped*	0...FFFFFFFF	6200 0108_h	unsigned 32	ro

Wertebereich nach DS-301, Tabelle 10-6, 10-7.

2.3.20 Objekt Transmit PDO Communication Parameter 1800_h

Mit diesem Objekt werden die Eigenschaften eines Sende-PDO definiert.

INDEX	**1800_h**
Name	*transmit PDO parameter*
Data Type	PDOCommPar

Index [Hex]	Sub-index	Beschreibung	Wertebereich [Hex]	Default	Datentyp	R/W
1800	0	*number_of_entries*	5	-	unsigned8	const
	1	*COB-ID used by PDO*	0...FFFFFFFF	180_h +Node-ID	unsigned32	ro
	2	*transmission type*	0...FF	255d	unsigned8	rw

Wertebereich nach DS-301, Tabelle 10-6, 10-7.

2.3.21 Transmit PDO Mapping Parameter $1A00_h$

Mit dem Objekt 'Transmit PDO Mapping Parameter $1A00_h$' kann die Zuordnung der Sendedaten zu den Tx-PDOs verändert werden.

INDEX	**$1A00_h$**
Name	*transmit PDO mapping*
Data Type	PDO Mapping

Die folgende Tabelle zeigt die Belegung der Transmit PDO Mapping Parameter für die Default-Konfiguration A:

Index	Subindex	Beschreibung	Wertebereich [Hex]	Default	Datentyp	R/W
1A00	0	*number of entries*	1	1_h	unsigned 8	ro
	1	*object_to_be_mapped*	0...FFFFFFFF	60000108_h	unsigned 32	ro

Wertebereich nach DS-301, Tabelle 10-6, 10-7.

2.4 Unterstützte Übertragungsarten nach DS-301, Tabelle 10-7

Transmission Type	PDO transmission					Supported by CAN-CBM-REL4
	cyclic	acyclic	synchronous	asynchronous	RTR only	
0	-	X	X	-	-	Yes
1...240	X	-	X	-	-	Yes
241...251	reserved					No
252	-	-	X	-	X	Yes
253	-	-	-	X	X	Yes
254	-	-	-	X	-	Yes
255	-	-	-	X	-	Yes

Es werden nur die Übertragungsarten 253 (Manufacturer Specific) und 255 (Profile Specific) verwendet.

2.5 Node Guarding / Life Guarding

Zur gegenseitigen Überwachung der CANopen-Module (insbesondere zum Erkennen von Verbindungsausfällen) können entweder die Heartbeatfunktion (Objekt 1017_h) oder das Node-Life-Guarding genutzt werden. Beim CAN-CBM-REL4-Modul schließen sich die Überwachungsfunktionen gegenseitig aus und können nicht gleichzeitig genutzt werden.
Im Gegensatz zum Node Guarding/Life Guarding kommt die Heartbeat-Funktion ohne RTR-Frames aus.

In der CANopen-Terminologie ist das Node-Guarding die Überwachung der NMT-Slave-Module (hier CAN-CBM-REL4) durch den NMT-Master, das Life-Guarding läuft dagegen im NMT-Slave ab und überwacht den NMT-Master.

Node-Guarding:

Der NMT-Master ruft mit einer RTR-Message (CAN-Request) zyklisch ein besonderes Telegramm von seinen NMT-Slaves (hier CAN-CBM-REL4) ab. Stimmt die Antwort der Slaves nicht mit der erwarteten Antwort überein, oder erfolgt die Antwort nicht innerhalb der *guard time*, erkennt der NMT-Master einen Fehler und reagiert entsprechend. Im Antworttelegramm sind der Modulstatus und ein Toggle-Bit enthalten.

Life-Guarding:

Die NMT-Slaves überwachen, ob sie im Rahmen des Node-Guardings vom NMT-Master abgefragt werden. Bleiben diese Abfragen für die Zeitdauer der *life time* aus, so wird ein lokaler RESET ausgelöst.
Die Life Time ist das Produkt aus *guard time* (Index $100C_h$) und dem *life time faktor* (Index $100D_h$).

Diese Seite ist bewußt unbedruckt.

3. Device Profile Area

3.1 Übersicht

Das Modul unterstützt nur die in der folgenden Tabelle aufgeführten Objekte.

Index	Name	Subindex	Typ	Zugriff	Default
6000_h	Read Input 8 Bit	0	unsigned 8	ro	No
		1	unsigned 8	ro	No
6200_h	Write Output 8 Bit	0	unsigned 8	ro	No
		1	unsigned 8	rw	No
6202_h	Change Polarity Output 8 Bit	0, 1	unsigned 8	rw	FF_h
6206_h	Error Mode Output 8 Bit	0, 1	unsigned 8	rw	0_h
6207_h	Error State Output 8 Bit	0, 1	unsigned 8	rw	FF_h
6208_h	Filter Constant Output 8 Bit	0, 1	unsigned 8	rw	0_h

ro - Read Only, rw - Read/Write

Die folgende Grafik verdeutlicht die Zusammenhänge der verwendeten *Digital Output Objects* (6200_h-6028_h) gemäß DS-401 für einen 8-Bit Zugriff:

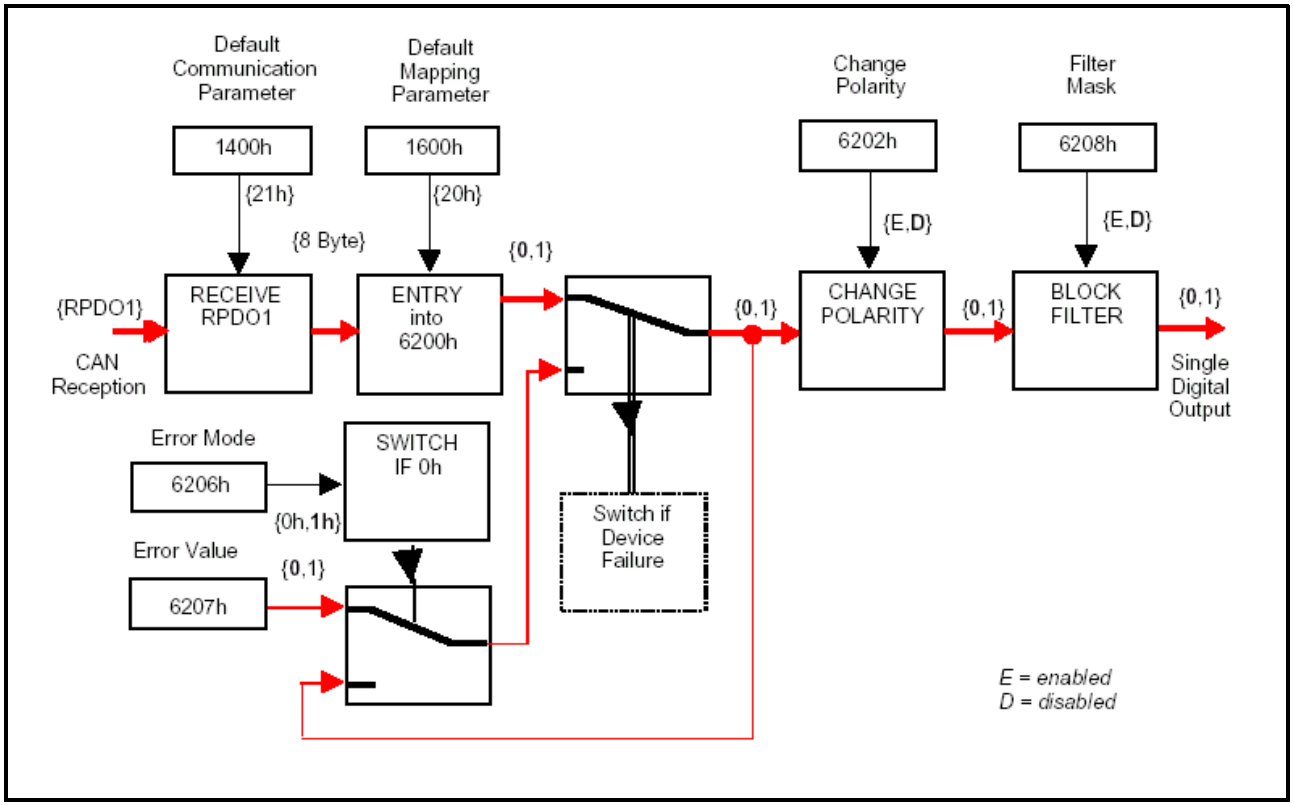


Abb. 3.1: Digital Output Objects

3.1.1 Change Polarity Output 8-Bit 6202_h

INDEX	**6202_h**
Name	*change polarity output*
Data Type	unsigned8

Dieses Objekt definiert den Zustand der 4 Relaisausgänge. Die Relaisstellung kann für jedes Relais einzeln invertiert werden.

Index [Hex]	Subindex	Beschreibung	Wertebereich [Hex]	Default	Datentyp	R/W
6202	0	*number_of_output 8-bit*	1	1	unsigned 8	ro
	1	*change polarity output*	0...F	0	unsigned 8	rw

Bedeutung der Variablen:

change polarity output Diese Variable bestimmt, bei welchem Relais die Stellung invertiert wird.

1 = invertiert
0 = nicht invertiert.

Die Werte für die vier Relais werden in dem untersten Byte übergeben:

Bit:	7	6	5	4	3	2	1	0
Inhalt:	reserviert				*change_polarity relais_4*	*change_polarity relais_3*	*change_polarity relais_2*	*change_polarity relais_1*

3.1.2 Error Mode Output 6206_h

INDEX	**6206_h**
Name	*error mode output 8-bit*
Data Type	unsigned8

Dieses Objekt gibt an, ob die Relais beim Auftreten eines internen Device-Fehlers auf einen bestimmten Error-Wert (Objekt 6207_h) gesetzt werden sollen oder nicht.

Index [Hex]	Subindex	Beschreibung	Wertebereich [Hex]	Default	Datentyp	R/W
6206	0	*number_of_output_8 bit*	1	1_h	unsigned 8	ro
	1	*error_mode_output*	0...FF	FF_h	unsigned 8	rw

Bedeutung der Variablen:

error_mode_output Diese Variable zeigt an, welches Relais auf einen vordefinierten Error-Wert (siehe Objekt 6207_h) gesetzt wird, wenn ein interner Device-Fehler auftritt.

1 = Relais soll auf einen in Objekt 6207_h definiertenWert gesetzt werden.
0 = Relais soll auch bei aufgetretenem Fehler nicht verändert werden.

Die Werte für die vier Relais werden in dem untersten Byte übergeben:

Bit:	7	6	5	4	3	2	1	0
Inhalt:	reserviert				*error_mode relais_4*	*error_mode relais_3*	*error_mode relais_2*	*error_mode relais_1*

3.1.3 Error Value Output 8-Bit 6207_h

INDEX	**6207_h**
Name	*error value output 8-bit*
Data Type	unsigned8

Bei internen Gerätefehlern werden die Relais auf den in diesem Objekt definierten Wert gesetzt, wenn der zugehörige Error Mode (Objekt 6206_h) aktiviert ist.

Index [Hex]	Subindex	Beschreibung	Wertebereich [Hex]	Default	Datentyp	R/W
6207	0	*number_of_output 8 bit*	1	1	unsigned 8	ro
	1	*error_value_output*	0...F	0	unsigned 8	rw

Bedeutung der Variablen:

Error Value Output Diese Variable enthält den Wert auf den die Relais gesetzt werden, wenn ein Fehler aufgetreten ist.

0 = Im Fehlerfall wird das Relais auf ‘0’ (enabled) gesetzt, wenn Objekt 6206_h aktiviert ist.
1 = Im Fehlerfall wird das Relais auf ‘1’ (disabled) gesetzt, wenn Objekt 6206_h aktiviert ist.

Die Werte für die vier Relais werden in dem untersten Byte übergeben:

Bit:	7	6	5	4	3	2	1	0
Inhalt:	reserviert				*error_value relais_4*	*error_value relais_3*	*error_value relais_2*	*error_value relais_1*

3.1.4 Filter Mask Output 8-Bit 6208_h

INDEX	**6208_h**
Name	*filter mask output 8-bit*
Data Type	unsigned8

Mit diesem Objekt kann eine Filter-Maske für die 4 Relais definiert werden.

Index [Hex]	Subindex	Beschreibung	Wertebereich [Hex]	Default	Datentyp	R/W
6208	0	*no_of_output_8-bit*	1	1_h	unsigned 8	ro
	1	*filter_mask_output*	0..FF	FF_h	unsigned 8	rw

Bedeutung der Variablen:

Filter Mask Output Diese Variable bestimmt, ob die Relais geschaltet werden können oder nicht.

1 = Relais disabled
0 = Relais enabled

Die Werte für die vier Relais werden in dem untersten Byte übergeben:

Bit:	7	6	5	4	3	2	1	0
Inhalt:	reserviert				*filter_mask relais_4*	*filter_mask relais_3*	*filter_mask relais_2*	*filter_mask relais_1*

Diese Seite ist bewußt unbedruckt.

4. PDO-Belegung

Die PDOs (Process Data Objects) dienen zur Übertragung der Prozeßdaten:

CAN-Identifier	Länge	Übertragungsrichtung
200_h + Node-ID	1 Byte	zum CAN-CBM-REL4 (Rx/Empfangs-PDO) Setzen der Relais-Ausgänge
180_h + Node-ID	1 Byte	vom CAN-CBM-REL4 (Tx/Sende-PDO) Abfragen des Relais-Status

Über das Rx-PDO kann der CANopen-Master die Relais-Zustände setzen.
Das erste Datenbyte des Rx-PDOs ist folgt belegt:

Rx-PDO	Datenbyte							
	7	6	5	4	3	2	1	0
200_h + Node-ID	-	-	-	-	*relais_4*	*relais_3*	*relais_2*	*relais_1*

Der CANopen-Master kann über das Tx-PDO mit Hilfe eines Remote-Frames die Zustände der Relais-Ausgänge abfragen.
Das erste Datenbyte des Tx-PDOs ist wie folgt belegt:

Tx-PDO	Datenbyte							
	7	6	5	4	3	2	1	0
180_h + Node-ID	-	-	-	-	*relais_4*	*relais_3*	*relais_2*	*relais_1*

Zustände der Relais:

relais	Schließerkontakt	Öffnerkontakt
‘0’	offen	geschlossen
‘1’	geschlossen	offen

Diese Seite ist bewußt unbedruckt.

5. Schnellstart

5.1 Konfiguration für den Einsatz im CANopen-Netzwerk

Für einen schnellen Start mit der einfachsten Konfiguration sind zunächst folgende Schritte nötig:

1. CAN-Verdrahtung aufbauen (Abschlüsse nicht vergessen!)

2. Baudrate einstellen: Nur wenn eine andere als die Default-Einstellung gewünscht wird. Die Default-Baudrate beträgt 125 kBaud.
Die Baudrate kann mit den Kodierschaltern (SW110, SW111), wie im Hardware-Handbuch (Kapitel: Konfiguration über die Drehschalter) beschrieben, gesetzt werden.

3. Node-ID einstellen: Sofern eine andere als die eingestellte Modul-Nr. (Node-ID) gewünscht wird, oder die Baudrate verstellt wurde, muß die Modul-Nr. mit den Kodierschaltern (SW110, SW111), wie im Hardware-Handbuch (Kapitel Konfiguration über die Drehschalter) beschrieben, gesetzt werden.
Die Modulnummer darf Werte zwischen 1 und 127 (01-7F$_h$) annehmen.
z.B. Node-ID 1 (01$_h$) einstellen:
Versorgungsspannung aus,
Kodierschalter SW110 (Low) auf ‘1’, SW 111 (High) auf ‘0’ drehen. Versorgungsspannung ein, Modul wacht im Zustand *Preoperational* auf.

4. Modul einschalten: Versorgungsspannung anlegen
Wurde zuvor die Modul-Nr. (Node-ID) eingestellt, liegt die Versorgungsspannung bereits an, und das Modul befindet sich im Zustand *Preoperational* (siehe Hardware-Handbuch Kapitel: Module-Status-LEDs).

5. Modul starten mit:

ID	Len	Data	
0	2 Byte	01$_h$	00$_h$

6. Ersten Ausgang setzen (PDO) mit:

ID	Len	Data
201	1 Byte	01$_h$

7. Status Ausgänge abfragen mit:

ID	RTR	Len
181	1	0

Modul antwortet mit:

ID	Len	Data
181	1 Byte	abhängig vom Zustand der Relais

5.2 Tabelle der wichtigsten Identifier und Nachrichten für CANopen

CAN-Identifier [HEX]	Bezeichnung	Länge	Daten [HEX]	Bemerkungen
0	NMT	2	02 xx	CAN-CBM-REL4-Modul geht in Zustand *Stopped*
0	NMT	2	01 xx	Start (CAN-CBM-REL4-Modul geht in Zustand *Operational*)
0	NMT	2	80 xx	CAN-CBM-REL4-Modul geht in Zustand *Preoperational*
0	NMT	2	81 xx	Reset CAN-CBM-REL4-Modul
0	NMT	2	82 xx	Reset Communication (hier gleiche Funktion wie ‘81xx’)
80_h + Node-ID	EMCY	8 Bytes	siehe Seite 14	Emergency Frame von CAN-CBM-REL4-Modul
580_h + Node-ID	SDO	8 Bytes	Parameter	Quittierung der Kommunikations-parameter von CAN-CBM-REL4-Modul (Tx)
600_h + Node-ID	SDO	8 Bytes	Parameter	Übertragung der Kommunikations-parameter zur CAN-CBM-REL4-Modul (Rx)
700_h + Node-ID	NMT	8 Bytes	Identifier	NMT error control identifier (siehe DS-301)
200_h + Node-ID	Rx_PDO_1	1 Byte	Nutzdaten	(Rx/Empfangs-PDO)
180_h + Node-ID	Tx_PDO_1	1 Byte	Nutzdaten	(Tx/Sende-PDO)

xx = Node-ID (Modulnummer)
Node-ID = $1...7F_h$