CAN-CBM-REL4

CANopen
Software Manual

Manual file:	I:\texte\Doku\MANUALS\CAN\Cbm\REL4\Englisch\CBREL4 _12S.en9
Date of print:	15.10.2002

Software described:	CANopen
Revision:	1.3

Changes in the chapters

The changes in the manual below affect changes in the **firmware** as well as changes in the **description** of the facts only.

Chapter	Changes versus previous version
2.	New objects: 1002_h, 1003_h, 1005_h, 1010_h, 1011_h, 1014_h, 1017_h, 1018_h, 1020_h, 1029_h Objects removed: $100B_h$, $100E_h$
3.	New objects: 6202_h, 6206_h, 6207_h, 6208_h

N O T E

The information in this document has been carefully checked and is believed to be entirely reliable. **esd** makes no warranty of any kind with regard to the material in this document, and assumes no responsibility for any errors that may appear in this document. **esd** reserves the right to make changes without notice to this, or any of its products, to improve reliability, performance or design.

esd assumes no responsibility for the use of any circuitry other than circuitry which is part of a product of **esd** gmbh.

esd does not convey to the purchaser of the product described herein any license under the patent rights of **esd** gmbh nor the rights of others.

esd electronic system design gmbh
Vahrenwalder Str. 207
30165 Hannover
Germany

Phone: +49-511-372 98-0
Fax: +49-511-372 98-68
E-mail: info@esd-electronics.com
Internet: www.esd-electronics.com

USA / Canada
esd
PMB 292
20423 State Road 7 #F6
Boca Raton, Florida 33498-6797
USA

Phone: +1-800-732-8006
Fax: +1-800-732-8093
E-mail: sales@esd-electronics.com

Contents

Page

This page is intentionally left blank.

1. General

Apart from basic descriptions of the CANopen, this chapter contains the most important information about the implemented functions.

A complete CANopen description is too extensive for the purpose of this manual.
Further information can therefore be taken from the CAL / CANopen documentations ‘CiA Draft Standard 201...207’, ‘CiA Draft Standard 301’ and ‘CiA Draft Standard Proposal 401’.

1.1 Definition of Terms

COB ...	Communication Object	
Emergency-Id...	Emergency Data Object	
n/a ...	not available	
NMT...	Network Management (Master)	
Rx...	receive	
SDO...	Service Data Object	
SW ...	Software	
Sync...	Sync(frame) Telegram	synchronisation identifier
tbd ...	to be defined	data has yet to be defined
Tx...	transmit	

PDOs (Process Data Objects)
PDOs are used to transmit the process data.
In the ‘Receive’-PDO process data is received by the CAN-CBM-REL4 module.
In the ‘Transmit’-PDO the CAN-CBM-REL4 module (which is always SLAVE in the CANopen network!) transmits data in the CANopen network.

The asynchronous mode of transmitting the PDOs is defined by ‘PDO transmission type’ in the communication parameter of the according process data object.

1.2 NMT Boot-up

The CAN-CBM-REL4 module can be initialized via the 'Minimum Capability Device'-boot-up, described in the CiA-Draft Standard 301 in chapter 8.3.

Usually, only a telegram to switch from *pre-operational* status into *operational* status after power-on is required. For this, you have to transmit for example the 2-bytes telegram '01_h', '00_h' (= Start Remote Node all Devices) to the CAN identifier '0000_h'.

1.3 CANopen Object Directory

The object directory is mainly a (sorted) group of objects which can be accessed via the network. Each object in this directory is addressed by a 16-bit index. This index is given in hexadecimal form in the object directories.

The index can be a 16-bit parameter according to the CANopen specification (CiA-Draft DS-301) or a manufacturer-specific code. By means of the MSBs of the index the object class of the parameter is determined.

Part of the object directory are among others:

Index [HEX]	Object	Example
0001 ... 009F	definition of data types	
1000 ... 1FFF	Communication Profile Area	1001 = error register
2000 ... 5FFF	Manufacturer Specific Profile Area	
6000 ... 9FFF	Standardised Device Profile Area	according to application profile DS-401
A000 ... FFFF	reserved	

1.4 Communication Parameters of the PDOs

The communication parameters of the PDOs (according to DS-301, chap.10.1.2) are transmitted as SDO (Service Data Objects) on ID '**600_h + Node-ID**' (Request). The receiver acknowledges the parameters on ID '**580_h + Node-ID**' (Response).
The **Node-ID** (module No.) is configured via coding switches (SW110, SW111). Please refer to the hardware manual of the CAN-CBM-REL4 module for a detailed description of possible configurations.

1.4.1 Accessing the Communication Parameters via SDO Telegrams

The SDOs (Service Data Objects) are used to access the object directory of a device.
An SDO is therefore a 'channel' to access the parameters of the device. Access via this channel is possible in *operational* and *pre-operational* status in the CAN-CBM-REL4 module.

This chapter does not describe all possible, but only some important modes of access in the CAN-CBM-REL4 module.
Definitions for the modes of access can be taken from CiA DS-202-2 (CMS-Protocol Specification, chap. 6: CMS Multiplexed Domain Protocols).

An SDO is structured as follows:

Identifier	command code	index (low)	index (high)	sub-index	LSB	data field	MSB

Example:
Requesting the device type for module number 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)

The value of device type for module number 19 (13_h) is: 0002.0191_h

Identifier
The parameters are transmitted on ID '**600_h + Node-ID**' (Request).
The receiver acknowledges the parameters on ID '**580_h + Node-ID**' (Response).

Command code

The command code transmitted consists of the command specifier and the length, among others. Frequently used combinations are, e.g.:

$40_h = 64_d$: Read Request, i.e. a parameter is to be read
$23_h = 35_d$: Write Request with 32 bits data, i.e. a parameter is to be written.

The CAN-CBM-REL4 module responds to each received telegram with a response telegram. The response telegram can contain the following command codes:

$43_h = 67_d$: Read Response, this telegram contains the desired parameter
$60_h = 96_d$: Write Response, i.e. a parameter has been successfully configured
$80_h = 128_d$: Error Response, i.e. the CAN-CBM-REL4 module reports a communication error.

Frequently used command codes

The following table gives an overview of frequently used command codes. The command frames always have to have 8 data bytes. Notes on the syntax and further command codes can be taken from CiA DS-202-2 (CMS-Protocol Specification, chap. 6: CMS Multiplexed Domain Protocols).

Command	for number of data bytes	command code [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

Error Codes of the SDO Domain Transfer

The following error codes might occur (according to CiA Work Draft WD-301):

Error code [HEX]	CAN-CBM-REL4 designation	Explanation
0x05030000	SDO_TOGGLE_BIT_NOT_ALTERNATED	Toggle bit not alternated
0x05040001	SDO_CS_UNKNOWN	wrong command specifier
0x06010002	SDO_READ_ONLY	no write access
0x06020000	SDO_WRONG_INDEX	wrong index
0x06070010	SDO_WRONG_LENGTH	wrong number of data bytes
0x06070012	SDO_PARA_TO_LONG	length of service parameter is too long
0x06070013	SDO_PARA_TO_SHORT	length of service parameter is too short
0x06090011	SDO_WRONG_SUBIND	wrong sub-index
0x06090030	SDO_VALUE_RANGE_EXEEDED	-
0x08000000	SDO_OTHER_ERROR	undefined cause of error
0x08000020	SDO_DATA_CANNOT_STORED	Data cannot be transferred or stored to

Index, Sub-Index
Index and sub-index will be described in chapter ‘Communication Profile Area’.

Data field
The maximum 4-bytes long data field is generally structured following the rule ‘LSB first, MSB last’. The LSB is **always** in ‘Data 1’.
In 16-bits values the MSB (bits 8...15) is always in ‘Data 2', and in 32-bits values the MSB (bits 24...31) is in ‘Data 4’.

This page is intentionally left blank.

2. Communication Profile Area

2.1 Terms and Abbreviations Used

The following terms will be used in the tables to describe the communication parameters:

PDO-Mapping	PDO mapping is available for this sub-index of the PDO
Access Mode	permissible modes of access to this parameter: ro... read_only This parameter can only be read. Write access triggers an error code. const.... constant This parameter cannot be changed by the user. It can be read. Write access triggers an error code. rw... read&write This parameter can be read or configured.
Value Range	value range of the parameter
Default-Value	default configuration of parameter when module is shipped
Name/Description	name and short description of parameter

2.2 Overview of Communication Parameters

The format of communication parameters can be taken from CiA DS-301, chap. 10.3. Special features of the CAN-CBM-REL4 implementation will be described in the following chapters of this manual.

The module only supports the communication parameters shown in the table below.

Index [HEX]	Name	Sub-index	Data Type	Access Mode	DefaultValue
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 sec.
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	-	Unsigned32	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*) attention: length > 4 bytes, i.e. upload more complex than with other parameters (see p. 17)
2*) depending on the software revision
3*) depending on default configuration

2.3 Description of Parameters

2.3.1 Device Type 1000_h

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

The value of *device type* is: 0002.0191_h (Digital Output: 0002_h Digital Profile Number: 0191_h)

Example: Reading the Device Type

The CANopen master transmits the Read Request on identifier '603_h' (600_h + Node-ID) to the CAN-CBM-REL4 module with module number 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 Read Request	00_h	10_h Index=1000_h	00_h Sub-index	00_h	00_h	00_h	00_h

CAN-CBM-REL4 module no. 3 responds to the master by means of Read Response on identifier '583_h' (580_h + Node-ID) with the value of the Device Type:

ID	RTR	LEN	DATA							
			1	2	3	4	5	6	7	8
583_h	0_h	8_h	43_h Read Response	00_h	10_h Index=1000_h	00_h Sub-index	91_h dig. Profile No 191	01_h	02_h Digital Output	00_h

value of Device Type: 0002.0191_h

The data field is always structured following the rule 'LSB first, MSB last (see chapter 1.4.1, Data field).

2.3.2 Error Register 1001_h

The CAN-CBM-REL4 module uses the error register to indicate error codes.

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

The following bits of the error register are being supported at present:

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

Bits which are not supported (-) are always returned as ‘0'.

generic	Error
0	no error
1	generic error

The following messages are possible:

00_h	no errors
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

The bits of the status register are set as described below:

Register bit (assembler)	Description	Level Assignment	
D25-D31	reserved	0	(always ‘0’)
D24	I²C Error	0 1	no I²C error I²C error
D3-D23	reserved	0	(always ‘0’)
D2	Watchdog Reset	0 1	no reset Watchdog reset executed
D1	Nodeguard Enabled	0 1	Nodeguarding off Nodeguarding on
D0	Heartbeat Enabled	0 1	Heartbeat function off Heartbeat function on

2.3.4 Pre-defined Error Field 1003_h

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

The *pre-defined error field* provides an error history of the errors that have been signaled via the Emergency Object.
Sub-index 0 contains the number of actual errors that are recorded in the array.
A new error is stored at sub-index 1. If a new error occurs, then the previous error gets stored to sub-index 2 and the new error is stored to sub-index 1and so on. So the older errors move down the list.

The error field works like a ring buffer. If it is full and a new error occures, the older errors move down the list and the oldest error is deleted.
The CAN-CBM-REL4 module supports a maximum of 16 error entries. At the apperance of the 17th error the oldest error is deleted.

Writing a ‘0’ to sub-index 0 deletes the entire error list. Higher values are not allowed. ‘0’ is the only permitted write access for this object.

With every new entry in the list the CAN-CBM-REL4 module sends an emergency frame.

Index [Hex]	Sub-index	Description	Value Range [Hex]	Default	Data Type	Access Mode
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

Parameter Description:

no_of_errors_in_list This parameter contains the actual number of the error codes written down on the list.
n = number of the error occured last
Set *no_of_errors_in_list* to ‘0’ to delete the error list.
If *no_of_errors_in_list* $\neq 0$, then error register (object 1001_h) is set

error-code x The 32 bit error code consists of the CANopen-Emergency-Error-Code (DS-301, Table 21) and the **esd** specific error codes (Manufacturer-Specific Error Field).

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

manufacturer-specific error field: Always '00' for CAN-CBM-REL4 module

emergency-error-code: Only the following error codes are supported:
0000_h - Emergency-Error-Reset or no Error
1000_h - Emergency Generic Error
(Error of I²C-EEPROMs)

Emergency-Frame

The structure of the emergency frame send by the CAN-CBM-REL4:

Byte:	0	1	2	3	4	5	6	7
Content:	*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

Structure of the parameter:

Bit No.	Value	Meaning
31 (MSB)	-	do not care
30	0/1	0: Module does not generate SYNC message 1: Modul generates SYNC message
29	0	always 0, because only 11-bit-IDs are supported by the CAN-CBM-REL4 module
28...11	0	always 0, because no 29-bit-IDs are supported by the CAN-CBM-REL4 module
10...0 (LSB)	x	bit 0...10 of SYNC-COB-ID

The value of the ID can be $0...7FF_h$.

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'

During the upload of the string data has to be exchanged as outlined in the following table (all values are in hexadecimal form). The specified commands must always transmit 8 data bytes.

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Note
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 ack., len = 12
Client:	**60**	00	00	00	00	00	00	00	segment upload request
CBM-REL4:	**0**	**43**	**41**	**4**	**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

Please refer to CiA DS-202-2 (CMS-Protocol Specification) for a detailed description of the Domain Upload.

2.3.7 Manufacturer's Hardware Version 1009_h

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

Similar to reading the Manufacturer Device Name, the hardware version is read via the Domain Upload protocol. A detailed description of the upload can be taken from CiA DS-202-2 (CMS-Protocol Specification).

2.3.8 Manufacturer's Software Version $100A_h$

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

Similar to reading the Manufacturer Device Name, the software version is read via the Domain Upload Protocol. A detailed description of the upload can be taken from CiA DS-202-2 (CMS-Protocol Specification).

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

The CAN-CBM-REL4 module supports the Node Guarding or the Heartbeat function (see p. 25)

Guard Time and Life Time Factor are evaluated together. The multiplication of both values is the Life Time. The Guard Time is shown in milliseconds.

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 $100E_h$

The module only supports 11-bit identifiers. The parameter can only be read.

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

Structure of parameter *node guarding identifier* :

Bit No.	Value	Meaning
31 (MSB) 30	-	reserved
29...11	0	always 0, because 29-bit IDs are not supported
10...0 (LSB)	0 x	bits 0...10 of Node Guarding Identifier

2.3.11 Store Parameters 1010_h

This object stores the parameters to the EEPROM. Only the command 'Save all Parameters' is supported.
Storing is only executed when the specific signature (see below) is written to the appropriate sub-index. On read access to the appropriate sub-index the device provides information about its storage functionality (for further information see CiA DS-301).

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

Index [Hex]	Sub-index	Description	Value Range [Hex]	Data Type	Access Mode
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 which can be saved or loaded:

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

With this object the default values of parameters are restored. The individual choice of parameters, stored to the EEPROM, will get lost. Only the command 'Restore all Parameters' is supported.

In order to avoid the restoring of default parameters by mistake, restoring is only executed when the specific signature (see below) is written to the appropriate sub-index.
On read access to the appropriate sub-index the device provides information about its default parameter restoring capability (for further information see CiA DS-301).

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

Index [Hex]	Sub-index	Description	Value Range [Hex]	Data Type	Access Mode
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

The list of communication parameter and application parameter, which can be stored or loaded is on page 22.

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

This object defines the COB-ID of the Emergency Object (EMCY).

The structure of this object is shown in the following table:

Bit Number	Value	Description
31 (MSB)	0/1	0: EMCY exists / is valid 1: EMCY does not exist / is not valid
30	0	reserved (always 0)
29	0	always 0, since 11-bit ID
28...11	0	always 0, since 29-bit IDs are not supported
10...0 (LSB)	x	bit 0...10 of COB-ID

The value of the ID can be $0...7FF_h$.

2.3.14 Producer Heartbeat Time 1017_h

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

The producer heartbeat time defines the cycle time of the heartbeat. The time has to be a multiple of 1 ms. Setting the producer heartbeat time to a value unequal '0' on a running device the heartbeat producer at that device starts immediately to send heartbeat frames and stops the node-/ life guarding (see page 35). Setting the producer heartbeat time to '0' stops the transmission of the heartbeat by the module.

The heartbeat function can be used for mutual monitoring of the CANopen modules (particularly for recognizing connection failures). Unlike Node Guarding/Life Guarding the heartbeat protocol defines an error control service without need for remote frames.

Description:
A module, the so called heartbeat producer transmits at the node-guarding identifier (see object $100E_h$) cyclically a heartbeat message on the CAN bus. One or more heartbeat consumer receive the message. The heartbeat consumer guards the reception of the heartbeat within the heartbeat consumer time.
If the message is not received in the heartbeat consumer time, a heartbeat event will be generated on the heartbeat consumer module.

Every module can be heartbeat producer and heartbeat consumer. There can be several heartbeat producers and consumers in a CAN net. The CAN-CBM-REL4 module exclusively works as heartbeat producer.

Index [Hex]	Sub-index	Description	Value Range [Hex]	Default Value	Data Type	Access Mode
1017	0	*producer_heartbeat_time*	0...FFFF	0 ms	unsigned 16	rw

Parameter Description:

producer_heartbeat_time Cycle time of the heartbeat producer to send the heartbeat at the node guarding-ID (see object $100E_h$).
The consumer heartbeat time has to be higher than the corresponding *producer_heartbeat_time* configured on the device producing this heartbeat.

2.3.15 Identity Object 1018_h

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

The indentity object returns general information about the CAN module.

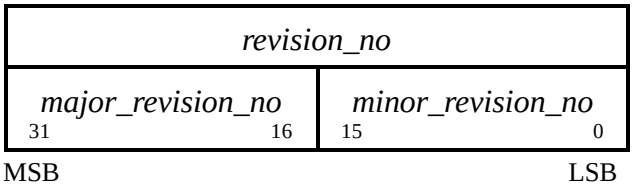
Index [Hex]	Sub-index	Description	Value Range [Hex]	Default Value [Hex]	Data Type	Access Mode
1018	0	*no_of_entries*	1	1	unsigned 8	ro
	1	*vendor_id*	0...FFFFFFFF	0000 0017	unsigned 32	ro
	2	*product_code*	0...FFFFFFFF	2283 3010	unsigned32	ro
	3	*revision_number*	0...FFFFFFFF	'Software revision'	unsigned32	ro
	4	*serial_number*	0...FFFFFFFF	'Hardware revision'	unsigned32	ro

Parameter Description:

vendor_id This parameter returns the **esd**-Vendor-ID. The value is always 00000017_h.

product_code This parameter returns the **esd**-order number of the module.
Example:
In the value '2**2 83 30 10**$_h$' the order number 'C.2833.01' is coded.

revision_number This parameter returns the software version. The upper two bytes return the revision numbers of the major changes according to DS-301 and the lower two bytes return the revision number of minor changes.



serial_number

This parameter returns the serial number code of the PCB-hardware.
The higher two bytes of *serial_no* contain the letters that code the production lot. They return the ASCII-code of the letters with the most significant bit set to '1', to distinguish letters from numbers:
(ASCII-code) + 80_h = returned bytes

The following numbers code the number of each module as BCD-value.

Example:
If the value 'C1 C2 0105_h ' is returned, this will mean the hardware serial number code 'AB 0105'. This value has to match with the value labelled at the module.

2.3.16 Verify Configuration 1020_h

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

In this parameter the date and the time of the last configuration can be stored. This can be used to check if this is the correct configuration version.

Index [Hex]	Sub-index	Description	Value Range [Hex]	Default Value	Data Type	Access Mode
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

Parameter Description:

configuration_date Date of the last configuration.
The value returns the number of days since the 01.01.1984.

configuration_time Time in ms since midnight at the day of the last configuration.

2.3.17 Error Behaviour Object 1029_h

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

If an error event appears (e.g. heartbeat error), the module will switch to the state that is defined in the parameter *communication_error.*

Index [Hex]	Sub-index	Description	Value Range [Hex]	Default Value	Data Type	Access Mode
1029	0	*no_of_error_classes*	1	1	unsigned8	ro
	1	*communication_error*	0, 1, 2	0	unsigned8	rw

Description of Parameter:

no_of_error_classes Number of error classes (here always ‘1’)

communication_error 0 - pre-operational (only if current state is operational)
1 - no state change
2 - stopped

2.3.18 Receive PDO Communication Parameter 1400_h

Object 'Receive PDO Communication Parameter 1400_h' defines the features of a receive PDO (Rx-PDO).

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

Index [Hex]	Sub-index	Description	Value Range [Hex]	Default Value	Data Type	Access Mode
1400	0	*no_of_entries*	2	2_h	unsigned 8	const.
	1	*COB-ID_used_by_ PDO*	0... FFFFFFFF	200_h + Node-ID	unsigned 32	ro
	2	*transmission_type*	0...FF	255_d	unsigned 8	ro

Value Range according to DS-301, table 10-6, 10-7.

2.3.19 Receive PDO Mapping Parameter 1600_h

Object 'Receive PDO Mapping Parameter 1600_h' changes the assignment of receive data to Rx-PDOs.

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

The following table shows the assignment of Receive PDO Mapping parameters for default configuration:

Index [Hex]	Sub-index	Description	Value Range [Hex]	Default Value	Data Type	Access Mode
1600	0	*no_of_entries*	2	2_h	unsigned 8	ro
	1	*object_to_be_mapped*	0... FFFFFFFF	6200 0108_h	unsigned 32	ro

Value Range according to DS-301, table 10-6, 10-7.

2.3.20 Object Transmit PDO Communication Parameter 1800_h

This object defines the features of a transmit PDO.

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

Index [Hex]	Sub-index	Description	Value Range [Hex]	Default Value	Data Type	Access Mode
1800	0	*no_of_entries*	5	-	unsigned8	const
	1	*COB_ID_used_by_PDO*	0... FFFFFFFF	180_h + Node-ID	unsigned32	ro
	2	*transmission_type*	0...FF	255_d	unsigned8	rw

Value Range according to DS-301, table 10-6, 10-7.

2.3.21 Transmit PDO Mapping Parameter $1A00_h$

Object ‘Transmit PDO Mapping Parameter $1A00_h$’ can change the assignment of transmit data to Tx-PDOs.

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

The following table shows the assignment of Transmit PDO Mapping parameters for default configuration A:

Index [Hex]	Sub-index	Description	Value Range [Hex]	Default Value	Data Type	Access Mode
1029	0	*no_of_entries*	1	1	unsigned8	ro
	1	*object_to_be_mapped*	0... FFFFFFFF	60000108_h	unsigned32	ro

Value Range according to DS-301, table 10-6, 10-7.

2.4 Supported Transmission Modes According to DS-301, Table 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

Only transmission modes 253 (manufacturer-specific) and 255 (profile-specific) are used.

2.5 Node Guarding / Life Guarding

The Node-Life Guarding as well as the heartbeat function (object 1017_h) have been designed for a mutual monitoring of control and the CAN modules connected (especially to detect connection failures). For the CAN-CBM-REL4 both functions cannot be chosen simultaneously.
Unlike the the Node Guarding/Life Guarding the heartbeat function doesn' t need any RTR frame.

In CANopen terminology Node Guarding is the monitoring of NMT-slave modules (here CAN-CBM-REL4) by the NMT-master. Life Guarding, however, runs in the NMT-slave and monitors the NMT-master.

Node Guarding:

By means of an RTR-message (CAN request) the NMT-master requests cyclically a special telegram from its NMT-slaves (here CAN-CBM-REL4). If the response of the slaves does not correspond to the response expected, or if the response does not come within the *guard time*, the NMT-master detects an error and reacts accordingly. The response telegram contains the module status and a toggle bit.

Life Guarding:

NMT-slaves monitor, whether they are requested by the NMT-master in the range of Node Guarding. If these requests did not come within the *life time*, a local RESET is triggered.
The Life Time is the product from *guard time* (index $100C_h$) and *life time factor* (index $100D_h$).

This page is intentionally left blank.

3. Device Profile Area

3.1 Overview

The module only supports the objects shown in the following table.

Index [HEX]	Name	Sub-index [HEX]	Data Type	Access Mode	Default [HEX]
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

The figure below shows the relationship between the used *Digital Output Objects* (6200_h-6028_h) according to DS-401 for an 8-Bit access:

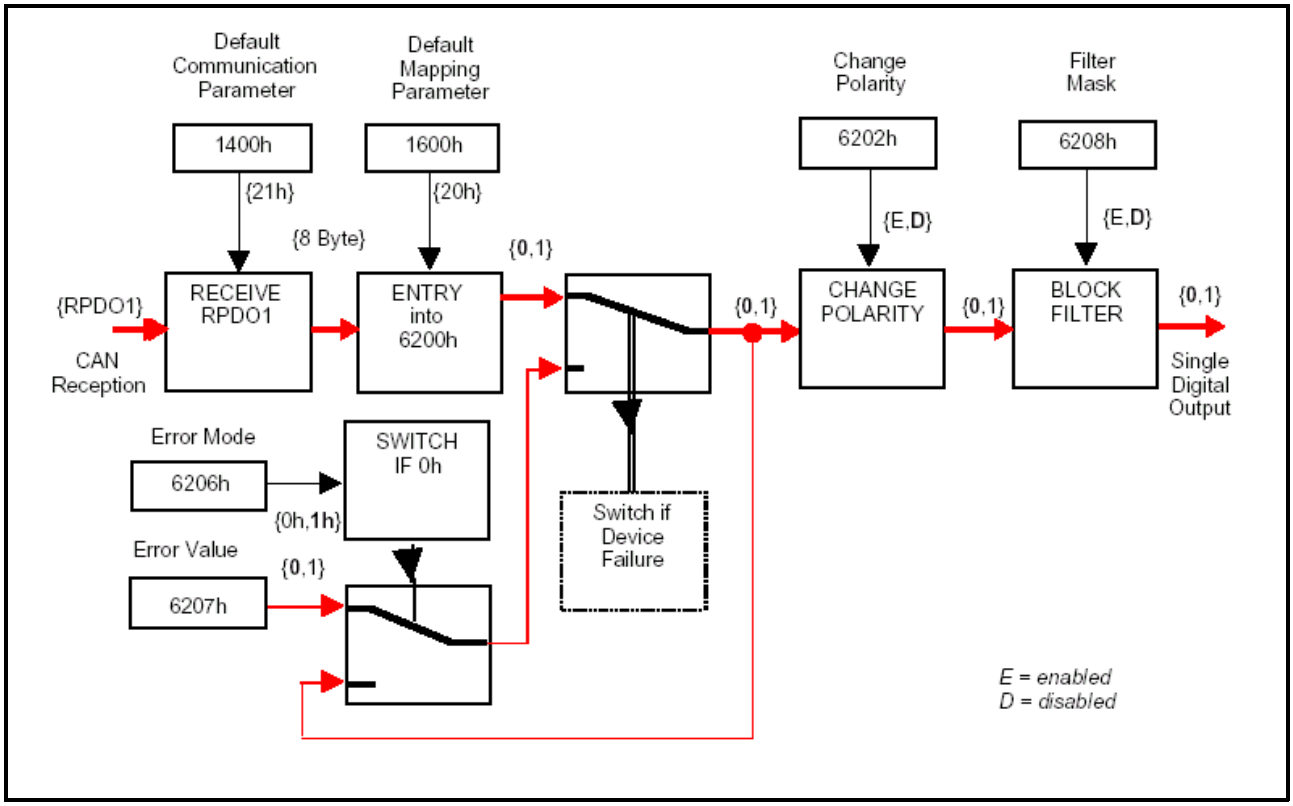


Fig. 3.1: Digital output objects

3.1.1 Change Polarity Output 8-Bit 6202_h

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

This object defines the status of the 4 relais outputs. The status of the four relays can be inverted individually.

Index [Hex]	Sub-index	Description	Value Range [Hex]	Default Value	Data type	Access Mode
6202	0	*number_of_output 8-bit*	1	1	unsigned 8	ro
	1	*change polarity output*	0...F	0	unsigned 8	rw

Description of the parameter:

change polarity output This parameter can invert the status of the four relais.

1 = inverted
0 = not inverted

The values of the four relays are returned in the lower byte:

Bit:	7	6	5	4	3	2	1	0
Content:	reserved				*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

This object indicates whether an relay is set to a pre-defined error value (object 6207_h) in case of an internal device failure.

Index [Hex]	Sub-index	Description	Value Range [Hex]	Default Value	Data Type	Access Mode
6206	0	*number_of_output_8-bit*	1	1	unsigned 8	ro
	1	*error_mode_output*	0...FF	FF_h	unsigned 8	rw

Description of parameter:

error_mode_output This parameter indicates whether an relay is set to a pre-defined error value (object 6207_h) in case of an internal device failure.

1 = Relay shall take the pre-defined condition specified in object 6207_h in case of an internal device failure
0 = Relay status shall be kept in case of an internal device failure

The values of the four relays are returned in the lower byte:

Bit:	7	6	5	4	3	2	1	0
Content:	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

On condition that the corresponding Error Mode (see Objekt 6206_h) is active, device failure sets the relay outputs to the value configured by this object.

Index [Hex]	Sub-index	Description	Value Range [Hex]	Default Value	Data Type	Access Mode
6207	0	*number_of_output_8-bit*	1	1	unsigned 8	ro
	1	*error_value_output*	0...F	0	unsigned 8	rw

Description of parameter

Error Value Output The parameter returns the error values of the four relays.

0 = In case of device failure, the value of the relais is set to ‘0’ (enabled), if object 6206_h is active
1 = In case of device failure, the value of the relais is set to ‘1’ (disenabled), if object 6206_h is active

The values of the four relays are returned in the lower byte:

Bit:	7	6	5	4	3	2	1	0
Content:	reserved				*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

This object defines an additional configurable output filter mask for the 4 relay outputs.

Index [Hex]	Sub-index	Description	Value Range [Hex]	Default Value	Data Type	Access Mode
6208	0	*number_of_output_8-bit*	1	1	unsigned 8	ro
	1	*filter_mask_output*	0..FF	FF_h	unsigned 8	rw

Description of Parameter:

Filter Mask Output This parameter indicates whether the relays can be switched or not

1 = Relay disabled
0 = Relay enabled

The values of the four relays are returned in the lower byte:

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

This page is intentionally left blank.

4. PDO Assignment

PDOs (Process Data Objects) transmit the process data:

CAN identifier	Length	Transmission direction
200_h + Node-ID	1 byte	to the CAN-CBM-REL4 (Rx/receive PDO) setting relay outputs
180_h + Node-ID	1 byte	from the CAN-CBM-REL4 (Tx/transmit PDO) requesting relay status

By means of the Rx-PDO the CANopen master can configure the relay status.
The first data byte of the Rx-PDO is assigned as follows:

Rx-PDO	Data byte							
	7	6	5	4	3	2	1	0
200_h + Node-ID	-	-	-	-	*relay 4*	*relay 3*	*relay 2*	*relay 1*

By means of the Tx-PDO and a remote frame the CANopen master can request the status of relay outputs.
The first data byte of the Tx-PDO is assigned as follows:

Tx-PDO	Data byte							
	7	6	5	4	3	2	1	0
180_h + Node-ID	-	-	-	-	*relay 4*	*relay 3*	*relay 2*	*relay 1*

Relay status:

relay	Maker	Breaker
‘0’	open	closed
‘1’	closed	open

This page is intentionally left blank.

5. Quick Start

5.1 Configuration for Use in CANopen Network

For a quick start with the most basic configuration the following steps have to be observed:

1. Wire CAN (do not forget terminations!)

2. Set baud rate: Only if a baud rate other than the default configuration is desired. The default baud rate is 125 kbaud.
The baud rate can be configured via coding switches (SW110, SW111), as described in the hardware manual (chapter: Configuration Via Rotary Switches).

3. Set Node-ID.: If you desire a module number (Node-ID) different from the default number, or if the baud rate has been changed, the module number has to be configured via coding switches (SW110, SW111), as described in the hardware manual (chapter: Configuration Via Rotary Switches). Permissible values for the module number are between 1 and 127 (01-$7F_h$).
E.g. Set Node-ID 1 (01_h):
power supply off,
set coding switch SW110 (Low) to '1', SW 111 (High) to '0'.
Power supply on, module wakes up in *pre-operational* status.

4. Switch on module: Supply power.
If the module No. has been configured before, power is already supplied and the module is in *pre-operational* status (see hardware manual chapter: Module Status LEDs).

5. Start module with:

ID	Len	Data	
0	2 bytes	01_h	00_h

6. Configure first output (PDO) with:

ID	Len	Data
201	1 byte	01_h

7. Request output status with:

ID	RTR	Len
181	1	0

Module responds with:

ID	Len	Data
181	1 byte	depending on relay status

5.2 Table of Most Important Identifiers and Messages for CANopen

CAN identifier [HEX]	Designation	Length	Data [HEX]	Explanations
0	NMT	2	02 xx	CAN-CBM-REL4 module gets into *stopped* status
0	NMT	2	01 xx	Start (CAN-CBM-REL4 module gets into *operational* status)
0	NMT	2	80 xx	CAN-CBM-REL4 module gets into *pre-operational* status
0	NMT	2	81 xx	Reset CAN-CBM-REL4 module
0	NMT	2	82 xx	Reset Communication (here the same function as '81xx')
80_h + Node-ID	EMCY	8 bytes	see page 14	Emergency frame of CAN-CBM-REL4 module
700_h + Node-ID	NMT	8 bytes	identifier	NMT error control identifier (see DS-301)
580_h + Node-ID	SDO	8 bytes	parameter	Acknowledging communication parameters by CAN-CBM-REL4 module (Tx)
600_h + Node-ID	SDO	8 bytes	parameter	Transmitting communication parameters to CAN-CBM-REL4 module (Rx)
200_h + Node-ID	Rx_PDO_1	1 byte	user data	(Rx/receive PDO)
180_h + Node-ID	Tx_PDO_1	1 byte	user data	(Tx/transmit PDO)

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