CANbloc-Mini
CBM-AO4

4 Analog Outputs

Manual Supplement

Document file:	I:\TEXTE\DOKU\MANUALS\CAN\CBM\AO4\AO4-01.EN6
Date of print:	04.01.2000

Changes in the chapters

The changes in the user's manual listed below affect changes in the *hardware* as well as changes in the *description* of the facts only.

Chapter	Changes versus previous version
-	1. edition
-	

Technical details are subject to change without notice.

Contents

Page

1. Overview

1.1 Preface

This supplement to system manual ‘CBM-DIO8’ contains **additional** information about analog outputs of the analog module ‘CBM-AO4’. General information about the modules can be found in system manual ‘CBM-DIO8’.

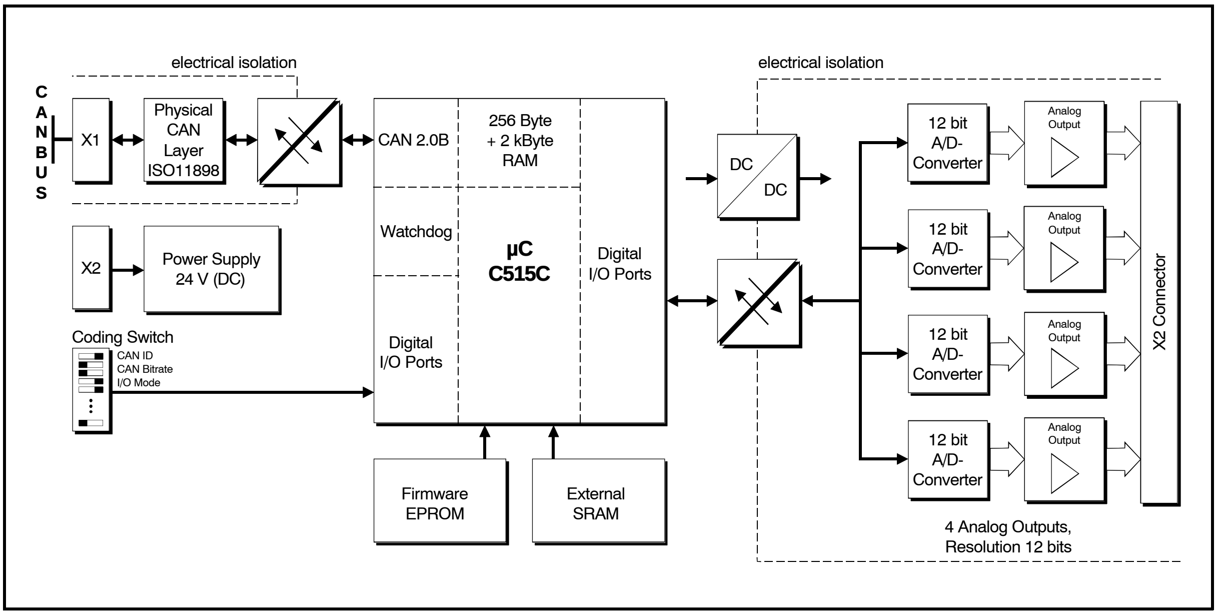
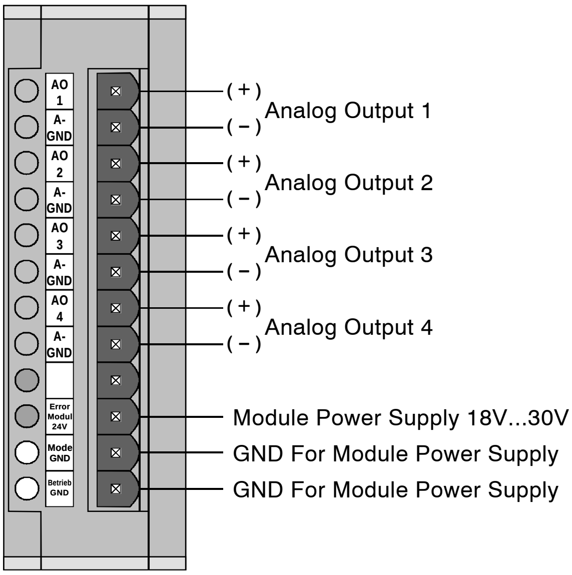


Fig. 1.1.1: Block circuit diagram of CBM-AO4 module

1.2 Connector Assignment



2. Analog Outputs of the CBM-AO4 Module

2.1 Electrical Features of Analog Outputs

All four equal analog outputs of this module are bipolar voltage outputs with following features:

- voltage range -10 V to +10 V
- resolution 12 bits **plus** sign
- minimum load resistor 2kΩ
- lowpass for noise reduction, realized by means of hardware, time constant ca. 0.5 ms
- outputs are calibrated via hardware by means of precision resistors
- after switching on supply voltage, all outputs have 0 V first

3. Access to Analog Outputs

The analog outputs are serially controlled by the processor of the I/O unit via several digital/analog converters. The outputs can be updated every 5 milliseconds (depending on the kind of access via CAN-bus).
Principally, controlling the first and last analog output takes about 0.5 milliseconds within a cycle, even if all four outputs are to be controlled simultaneously via CAN-bus.

3.1 Controlling the Analog Outputs via SDO

The outputs can be controlled via **SDO** (**S**ervice **D**ata **O**bject) or **PDO** (**P**rocess **D**ata **O**bject). The CANopen-Dictionary-Objects which are **required** for this can be found in appendix B.

The most important object for controlling the analog outputs is object 0x6411 in accordance with CiA-Draft Specification 401. By means of this object one analog output can be controlled each by SDO-CAN telegram by writing the desired output level as ‘16-bit integer value’ into object 0x6411. -10 Volt correspond to value ‘-32768' and +10 Volt correspond to value ‘+32767’.

‘+32767’ is generally the coding of the maximum possible output level, no matter if this corresponds to +10 V or +20 mA and whether the converter has an accuracy of 10, 13 or 16 bits. In order to clarify this, the term ‘unconverted’ has been added to the description of object 0x6411 in DS401.
This does not mean that the numerical values in object 0x6411 are ‘unconverted’. Even in ‘unconverted’ status the ‘left-aligned’ CANopen format is converted into ‘right-aligned’ converter format (here: -2047 ... +2047)!

Normally, however, electrical outputs are not controlled via SDO, because this kind of access is quite slow and not synchronous. Furthermore, only **one** output can be set per SDO write access.

3.2 Controlling the Analog Outputs via PDO

If you want to update all four analog outputs via **one** CAN telegram, you have to transmit the analog data via PDO. For this you can map, e.g., the subindices 1...4 from object 0x6411 into the process data channel. The principal function of ‘PDO mapping’ is described in the general system manual (chapter ‘CANopen’, subchapter ‘PDO Mapping’).

The PDO mapping introduced here is part of the default setting of the CBM-AO4 module. The following list helps you to set it yourself (please observe the order):

1. Object 1600h, subindex 0: at **04h**
   => four objects are mapped into PDO.

2. Object 1600h, subindex 1: at **64110110h**
   => object 6411, map subindex 1 into PDO with 16 bits.

3. Object 1600h, subindex 2: at **64110210h**
   => object 6411, map subindex 2 into PDO with 16 bits.

4. Object 1600h, subindex 3: at **64110310h**
   => object 6411, map subindex 3 into PDO with 16 bits.

5. Object 1600h, subindex 4: at **64110410h**
   => object 6411, map subindex 4 into PDO with 16 bits.

Save this configuration permanently in the CBM-AO4 module, if necessary!
Modified parameters are permanently saved by object 0x1010 in the *CANopen ‘Communication Profile’*.

After this, the CBM-AO4 module expects a 16-bit value for the first analog output (low byte first) in the first two bytes of the process data channel, in the following two bytes the value for the second analog output and so on.

In contrast to CiA DS401 recommendation the CBM-AO4 uses the PDO channel for transmitting analog process data, although it is usually used for transmitting digital process data. The CBM-DIO8 has **always** just one transmit and one receive PDO. The **‘2nd PDO’** for analog data, as mentioned in DS401, **does not exist in the CBM-AO4**. If your application required the analog process data to be transmitted with the CAN identifier of the 2nd PDO, you can set the identifier basis of the PDO to the required value by means of code position 50 (see system manual and CiA DS401).

Note: Please consider that PDO transmissions only work in module status ‘operational’. By means of SDO you can set the analog outputs already in ‘pre-operational’ status!

3.3 Controlling the Analog Outputs in Case of Error and after Power-Up

It is possible to set the analog outputs to a pre-programmed value in case of error. CANopen-Dictionary Objects 0x6443 and 0x6444 (in accordance with CiA DS401), as described in appendix B, can be used to configure the outputs.
The only error recognized by the module (at the moment) is the failure of the CAN connection between CBM-AO4 module and superior control. In order to monitor the failure of the CAN connection the ‘Extended NMT Bootup’ with ‘Nodeguarding Activation’ in accordance with CANopen standard has to be used, which is quite complex, unfortunately. A short description of the CAN protocol required for this can be found in the general system manual in chapter ‘CANopen’.

Via object 0x6443 you define the ‘Output Fault MODE’, i.e. the general performance of the analog outputs in case of error (can be set individually for each channel). At the moment the following ‘Fault Modes’ are implemented in the firmware:

Fault Mode (Obj. 0x6443)	Designation in accordance with DS401	Meaning, notes
0	actual value rest	retain current value
1	revert to default value, unconverted	set value defined by object 0x6444
2	revert to default value, converted	is currently treated like Fault Mode 1 [19.05.99]

Table 3.3.1: Fault Mode of object 0x6443

In order to define the error performance of channel 1 use subindex 1 in object 0x6443, subindex 2 is responsible for channel 2, etc.

By means of object 0x6444 you can determine values with which the analog outputs are to be controlled in case of error. The format of the entries in object 0x6444 corresponds to the format of object 0x6411.

As a small addition (which did not exist in this form in DS401) object 0x6444 can also be used as **default value after switching on supply voltage** of the CBM-AO4 module. In default setting all outputs are 0 until they are controlled via SDO or PDO.

By means of object 0x6444, however, you can set the analog outputs after power-on to any ‘start value’.
A certain delay in milliseconds cannot be avoided, because the module configuration is first read from a serial controlled EEPROM after power-on. During the first milliseconds after power-on the analog outputs therefore always have value 0.

Program the desired start values of the analog outputs 1...4 in object 0x6444.1...4 and save the configuration permanently!

4. Appendix A: Additional Code Positions of the CBM-AO4 Module

Code No.	**Contents**	**Data type [unit]**	**Access r=read w=write**	**Setting / display possibilities**	**Default setting**	**Notes references**
Code positions for the basic module configuration (additional to the code positions described in the manual of the CBM-DIO8)						
18	hardware model	8 bits	r	0x40 = ‘D/A4 installed’ 0x00 = ‘no extension’	0x40	only for manufacturer
19	module options	8 bits	r	0x00 = ‘Lenze mode’ 0x01 = ‘DS401 mode’	depending on order, generally 0x01	only for manufacturer
Code positions for the CBM-AO4 with four analog outputs Please use the objects specified in CANopen DS401 for controlling the analog outputs!						
640.0	number ‘DA4’-registers	8 bits	r	0 (not installed) ... <number registers>	4	
640.1	current value of channel 1	16 bits	rw	-32767 ... +32767	0, is overwritten by obj. 0x6444	only for diagnostical matters
640.2	current value of channel 2	16 bits	rw	-32767 ... +32767	’’	
640.3	current value of channel 3	16 bits	rw	-32767 ... +32767	’’	
640.4	current value of channel 4	16 bits	rw	-32767 ... +32767	’’	
640.x	reserved	-	r			

5. Appendix B: Objects in Accordance with DS401 for Controlling the Analog Outputs

Index (hex)	Designation in accordance with DS401 (notes)	Subindices	Type	Access
6411	**'Writes value of the output channel (not converted)'** Write accesses to this object control the channel (according to subindex). Subindex 0 is read-only and contains the number of channels, here: 4 The subindices 1...4 of this object can be mapped into the PDO channel (as 16-bit parameters) (see 'PDO Mapping' in system manual). **Scaling:** -32767 = -10 V (min.) 0 = 0 V +32767 = +10 V (max.)	0 1 ... 4	Unsigned 8 Signed 16	ro rw
6443	**'Output Fault Mode'** This object defines how the analog outputs are to react in case of error. Default setting for subindex 1...4 is value 0.	0 1 ... 4	Unsigned 8 Unsigned 16	ro rw
6444	'**Default Output Fault value (not converted)'** This object defines the level which the analog outputs are to adopt in case of error, if object 6443 triggers according actions. In addition, for the CBM-AO4 they define the status of the analog outputs after power-on (before CAN-bus activity). Default setting for subindex 1...4 is value 0.	0 1 ... 4	Unsigned 8 Signed 16	ro rw

CANbloc-Mini
I/O Modules DIO8

For The Use In
CANopen - Networks

Manual

CAN-CBM-DIO8 Manual Rev. 0.83

Document file:	I:\texte\Doku\MANUALS\CAN\CBM\DIO8\Englisch\DIO8-083.en9
Date of print:	16.03.04

Changes in the chapters

The changes in the user's manual listed below effect changes in the **hardware**, as well as changes in the **description** of the facts only.

Chapter	Changes versus previous version
3.6	Order information changed
4.2.5	Schematic diagram of input and output circuits inserted

Further technical changes are subject to change without notice.

NOTE

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 electronics Inc.
12 Elm Street
Hatfield, MA 01038-0048
USA

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

Contents

Page

1. Preface and General

This manual is a complex technical information about the I/O-modules of the **C**AN**b**lock-**M**ini-series (CBM) CAN-CBM-DIO8.

A large part of the descriptions is valid also for other CBM-modules. Special data which concern only one of these CBM-modules are described in the additional manual.

A few words on how to put the module into operation can be found in the chapter 'Quick Start' on page 44.

1.1 Purpose of Use

If the module described in this manual is to be used with a motor controller, it is important also to follow the definitions, regulations and safety information from the manuals of the servo converters used.

This page is intentionally left blank.

2. Safety Information

The operator or security officers are committed to

- controlling that all important regulations, information and rules are followed,
- guaranteeing that only qualified personnel is working at and with the device components,
- guaranteeing that the manual is available to the personnel for all according operations, and
- forbidding unqualified personnel working at and with the device.

3. Short Description

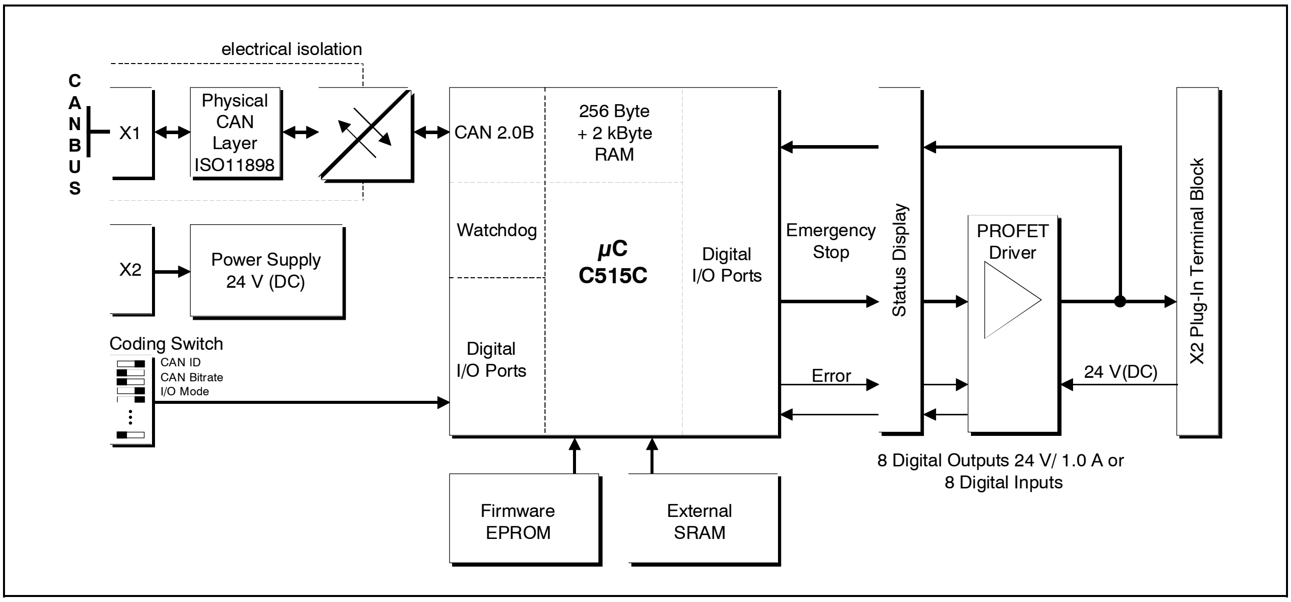


Fig. 3: Block-circuit diagram of the CAN-CBM-DIO8 module

3.1 Components

- decentral I/O-unit for CAN with eight connecting terminals which can be freely used as inputs or outputs.
- top-hat rails mounting
- connectors: screw-type/plug terminals
- two green status LEDs (‘Operation’ (below) and ‘Mode’)
- two red error LEDs (module- and output error)
- yellow LEDs for the I/O-signal display
- processor: SAB 80C515C-L
- CAN transceiver: PCA82C250T (SMD) electrically insulated

3.2 Mechanical Data

- case: UEGM-MSTB by Phoenix Contact
  dimensions: height: 79 mm
  depth: 90.5 mm (+10 mm for I/O - connector)
  width: 25 mm
- ambient temperature: 0...50 ˚C or extended temperature range -20...+70˚C
- position of the DIP switch at the upper case side (when snapped-open)
- position of the CAN connector at the lower case side
- position of the green operation LED below

3.3 Electrical Data

- voltage supply: 18 to 30 Volts DC, typical 24 V
- current: typical 30 mA, max. 80 mA (when outputs are switched-off)

3.3.1 Digital Inputs

- not electrically insulated
- input resistance: 3 kΩ ≤ Ri ≤ 4 kΩ
- input-voltage range ‘low’: 0...5 V
- input-voltage range ‘high’: 13...30 V

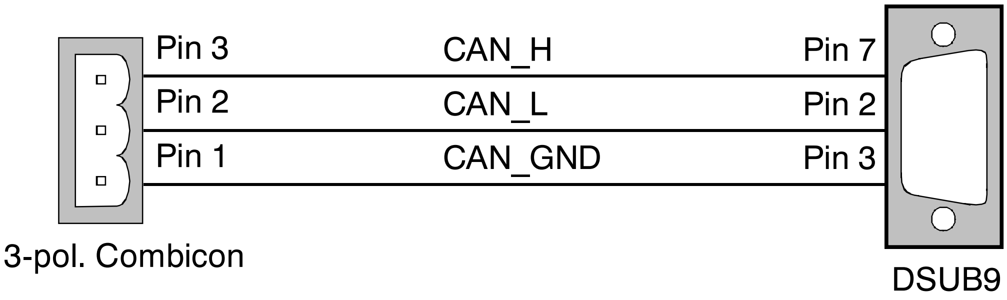
3.3.2 Digital Outputs

- not electrically insulated
- type: ‘PNP’ (high-side driver: high side switch)
- short-circuit proof
- maximum output current: ca. 1 A (per channel, driver specification!)
- short circuit recognition/ overload from about 4 A (per channel)
- output voltage ‘low’: 0 V via input resistances
  (with open output)
- output voltage ‘high’: > Ub - 1,8 V
  (with maximum load, supply with 24 V)
- the current output status can be read back
- shared short-circuit-status display by a red LED
- display by red LED, if total current is exceeded

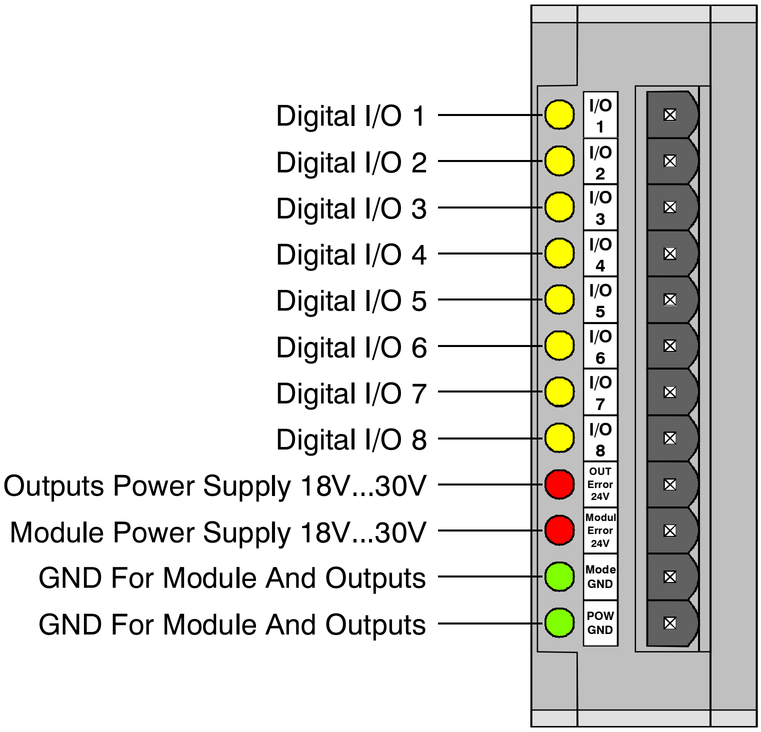
3.3.3 CAN Connection

3-pole COMBICON connectors (male) by Phoenix without shroud, pitch 5.00 mm, designation **MSTBA 2.5/3-G-5.00**

Assignment of the adaptor cable 3-pole Combicon <-> DSUB9:



3.3.4 Connection of Supply Voltage and I/O Connections



3.4 Case Design

- Case type: UEGM-MSTB by Phoenix Contact

3.5 Software

- CANopen with special extensions
- one receive- and one transmit-PDO
- one SDO (expedited protocol, max. 4 user-data bytes)
- node- and life guarding, emergency messages
- linking up to four modules to form a 'logical' module with 32 I/O-lines
- SYNC-frame evaluation (which is restricted when modules are linked)
- operating mode ‘DS401’ or ‘Servo’ can be selected via DIP switch

3.6 Order Information

Type	Properties	Order No.
CAN-CBM-DIO8	CANbloc-Mini module with 8 digital inputs/ outputs, 24 V / 1 A, 24 VDC, CANopen, 0...50 ˚C ambient temperature	C.2830.02
CAN-CBM-DIO8-12V	as C.2830.02, but 12 VDC power supply	C.2830.04
CAN-CBM-DIO8-T	as C.2830.02, but -20...+70 ˚C ambient temperature	C.2830.05
CAN-CBM-DIO8-MD	User manual in English 1*) (this manual)	C.2830.20
CAN-CBM-DIO8-ENG	Engineering manual in English 2*) Contents: circuit diagrams, PCB top overlay drawing, data sheets of significant components	C.2830.25

1*) If module and manual are ordered together, the manual is free of charge.
2*) This manual is liable of costs, please contact our support.

4. Digital Inputs/Outputs

Each I/O-connection is either defined as an input or an output via the DIP switch.
The status of the I/O-connections is each indicated by a yellow LED.

4.1 Transmission Rates of Digital Inputs

The transmission rates for the digital inputs can be distinguished:

- *cyclical but not synchronous*: The transmission-rate interval is only predetermined by the I/O-module (not designed for CANopen standard, but useful for simple applications!)

- *acyclical, synchronous*: The transmission is made after a SYNC message has been received (transmission type 0)

- *cyclical, synchronous*: The transmission is made after a certain number of SYNC messages each has been received (PDO-transmission type 1...240)

- *event-controlled, asynchronous*: The transmission is made when the status of certain digital inputs changes (PDO-transmission type 255)

If up to four I/O-modules are linked to form a 'logic' module, the I/O-slave modules transmit their digital input information first to the I/O-master. The master combines them, e.g., in a 4-byte message (PDO) for the motor controller (or for the master control) and then transmits them via the process-data object.

4.2 Digital Outputs

Each output driver has a nominal current of 1 A. If about 4 A per output are exceeded, the according output is switched off at once and an error is reported.
Factory-set all I/O-connections are only defined as inputs.

For the I/O-module type 'DIO8’ it is advisable to define the inputs and outputs always in pairs of two, as will be described on the following page.
In order to be able to drive them as outputs, the connection which is to be an output also has to be defined via DIP-switch or parameter channel (see chapter Parameterization, from page 26).

4.2.1 Restrictions in Defining I/O-Directions

The output driver for the 8 digital outputs has only 4 status lines which means that the error status of two output lines each cannot be distinguished. For technical reasons the error status of connections defined as INPUT has to be ignored (open-load detection of the driver).
If for instance I/O-connection 1 is used as digital input and I/O-connection 2 is used as digital output, possible errors (e.g. overload) of I/O-connection 2 cannot be recognized. Therefore it is advisable to use the I/O-lines 1 and 2, 3 and 4, 5 and 6 as well as 7 and 8 always as 'pair of two' with the same direction.

Example for an **advisable** combination:

1+2 = inputs, 3+4=outputs, 5+6=inputs, 7+8=outputs

Example for a **not** advisable combination:

1=output, 2=input, 3+4=outputs, 5+6=inputs, 7+8=outputs

In this example an overload of I/O-connection 1, e.g., will not be recognized, because I/O-connection 2 is defined as input and the internal status line of I/O 1+2 has therefore to be ignored.
Errors at outputs 3,4,7,8 however will still be recognized and shown.

4.2.2 Reading Back the Digital Outputs

Each I/O-line defined as an output always retains its function as a digital input as well, i.e. if an output is enabled, the current status will be transmitted as described in 4.1.

4.2.3 Reception Monitoring of the Digital Outputs (for Servo Mode)

The monitoring which will be described here is normally only useful for the use in servo mode (or if for other reasons no node guarding in accordance with CANopen is possible):
If the telegrams for the digital outputs (RX-PDO) are transmitted periodically, the I/O-module can monitor the receive cycle.

If no telegrams are received for a certain time (e.g. for 500 ms), the modules concerned switch all outputs to the predefined error status (e.g. all outputs off) and report an error via the red LED. The module remains in its normal operating mode, however. As soon as a telegram for the digital outputs is received again, the error display switches off.
The default setting of the according parameter (code position 30, see appendix) is 0, i.e. there is no reception monitoring, if the default setting is used.

4.2.4 Output Failures

If a short-circuit occurs or the maximum permissible output current of a channel is exceeded, only the output concerned is switched off driver-internally, the other outputs remain unaffected for the time being.

The output driver used, also signalizes errors, if an external voltage instead of a load conducting to ground is applied to its outputs. This is a fatal error and causes a reset of the whole outputs of the module. Therefore it must be avoided to apply an external voltage to an I/O-line which has been defined as an output!

Errors occurring at the outputs are shown by the red 'Out Error' LED.
Furthermore a CANopen-standard-compliant emergency telegram is transmitted.

4.2.5 Input and Output Circuits

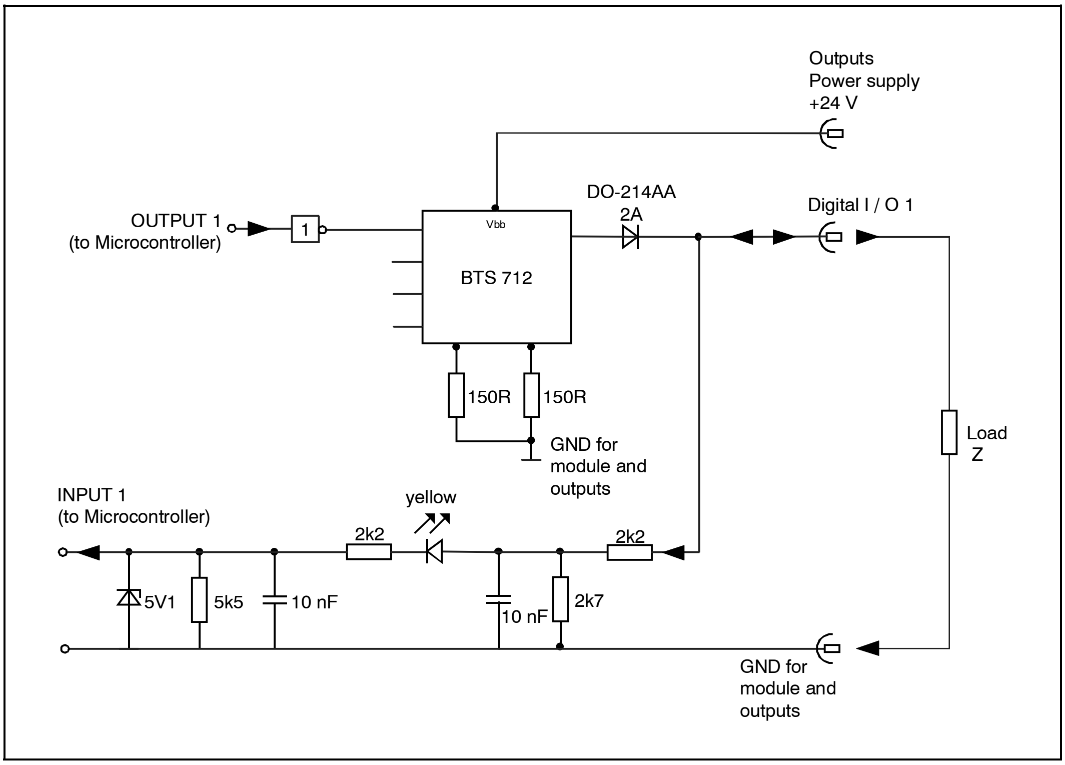


Figure 4.2.1: Schematic diagram of input and output circuits

5. Installation

Generally all guidelines regarding EMC-compliant installation, wiring, conductor cross sections, materials to be used, minimum clearances, lightning protection, etc. have to be strictly followed.

This section will only explain a few features in detail.

5.1 I/O-Module Protection

Note:

The I/O-modules contain components which are electrostatic-sensitive.

- Before working on the area of the connections, the personnel has to discharge itself electrostatically, e.g. by touching earthed metal or PE-screws.
- When connecting an I/O-module, first the earth connection is to be made and then the connection of the signal lines in order to degrade possibly existing differences in potential without risk.

5.2 CAN Wiring

The CAN is wired with lines which are ISO 11898 compliant. The following electrical features are required:

Parameter	Value			Description
	min.	nom.	max.	
impedance (Z)	108 Ω	120 Ω	132 Ω	natural impedance (HF), measured between two signal lines
resistivity (r)		70 mΩ/m		
specific line delay		5 ns / m		has to be added to the delays of the transmit and receive circuit

The CAN has to be terminated at both sides with a correct impedance, in other words, by a low-inductance 120 Ω - resistor. The I/O-modules of the 'DIO8'-series do not have a terminal resistance. The maximum possible bus length depends on the bit rate (bit timing). This will be explained in the chapter 'CAN'

6. LEDs

8 I/O-status display LEDs, 2 operating display LEDs and 2 error display LEDs are used:

- operating display ‘Operation GND’(green)
- mode display ‘Mode GND’(green)

- output-error display ‘OUT Error 24 V’(red)
- module-error display ‘Module Error 24 V’(red)

Normally only green LEDs should be on. A red LED indicates an error or a very special system status (e.g. programming mode).

If errors occur **during the initialisation phase**, the operating, mode and module-error LEDs start blinking at the same time in order to distinguish this error distinctly from 'normal' errors (which occur during operation).

This error can for instance be caused by:

- wrong setting of the DIP-switch, e.g. CAN address 0
- EEPROM defect
- other error during initialisation-automatic test

Errors which occur **during normal operation** are shown by the red error LEDs.
As long as the operating LED does not switch off, the error is not very serious.

Among these errors are, e.g.:

- no periodical messages from the motor controller
- a to high error rate on the CAN, caused for instance by a faulty CAN termination
- extreme busload so that messages cannot be transmitted
- temporary overload of individual outputs

If serious errors occur, the operating LED is switched off. Serious errors are for instance:

- CAN OFF
- hardware error

In this case the module does not work anymore, in other words, it does not accept commands by the CAN anymore. The module can only (if at all) be restarted by switching the module supply off and on again.

The various LED combinations will be explained in the following chapter.

6.1 LED - Display Combinations

Operating display	Mode display	Module error	Output error	Function
ON	ON	OFF	OFF	normal operation
ON	OFF (temporary)	OFF	OFF	received CAN message
BLINKING	ON	OFF	X	operating mode is not ‘operational’ (see below)
OFF	ON	BLINKING	OFF	programming mode
X	X	OFF	BLINKING	error of output driver
ON	ON	BLINKING	X	CAN error, protocol error etc. (see below) but module still in operation
OFF	ON	BLINKING	X	CAN OFF, module not in operation anymore
BLINKING	BLINKING	BLINKING	OFF	configuration or automatic-test error

Blinking of the operating display LED (green):

Blink code	Meaning	Error cure, notes
short-long-long-short	*preoperational*, module is configured as I/O-master	make sure that the CAN connections are correct, transmit 'Start Node', servo mode: motor controller has to operate as BUS master
short-short-short	*preoperational*, module is configured as I/O-slave	see above
regular, slow blinking	*preoperational*, I/O-master / slave cascading is switched off	see above

Blinking of the module error LED (red)

Classifying *some* blinking codes for precise diagnosis:

Blink code	Meaning	Error cure, notes
long - short-short	timeout by RX-monitor	check CAN connection, switch off RX-monitor of I/O-module (code no. 30) servo mode: activate periodical transmission of CAN-OUT3 of the motor controller
long-short	*AI4 only*: Error of the analog input module (e.g.:overcharging)	check clamp connections, check if the analog input signals lie in the defined range
	node guard error (node guard RTRs from NMT-master fail to arrive)	check CAN connection, increase lifetime factor according to DS203-2, (see description 'Extended NMT-Boot up')
long-short-long-short	CAN error or CAN OFF	correct bit rate for all bus users? CAN terminated correctly? CAN shielded correctly?
short-short-short	slave error (only if ‘internal' Master-Slave cascading is activated)	The I/O-master module has detected an error at one of its I/O-slave modules: check CAN lines and supply lines
(other codes)	CAL-error register is not 0	read out CAL-error register and keep on investigating (object 1001_h, see CANopen)

'Operation', 'Mode' and 'Error' are blinking at the same time

Blink code	Meaning	Error cure, notes
3 * blinking	automatic-test error type 3	check voltage supply, maybe a hardware defect
4 * blinking	EEPROM error	see page 25 , maybe hardware defect
5 * blinking	error at DIP configuration	see page 22, e.g. set CAN address > 0

7. CAN

This chapter will describe physical features of the CAN and the most important protocols required for the use of the module, implemented into the I/O-modules.
Further information on the higher protocol layers can be taken from the CAL/CANopen documentation ‘CiA Draft Standard 201 ... 207’, ‘CiA Draft Standard 301’ and ‘CiA Draft Standard Proposal 401’. There the communication profile(301) and the device profile (401) will be explained. If you want to know exactly to what extent the 'objects' described in these documentation are implemented in the I/O-module family 'DIO8x', please consult the data sheets of the special modules. The objects generally implemented will be described starting from page 31.

The CAN identifiers used for the data transfer are listed in the appendix.

7.1 CAN Timing

The CAN timing has been set in compliance with the CiA-guidelines (certain deviations are due to the hardware used). Depending on the CAN-bit rate set (see also page 23), the CAN controller is programmed according to the following table. Furthermore the specified maximum bus lengths apply depending on the bit rate:

Bit rate	Tseg 1	Tseg 2	Tsync	max. bus length (cable: 5 ns/m)
1000 Kbit / s	700 ns	200 ns	100 ns	10 m
500 Kbit / s	1600 ns	300 ns	100 ns	80 m
250 Kbit / s	3400 ns	400 ns	200 ns	250 m
125 Kbit / s	6.5 µs	1 µs	500 ns	500 m
100 Kbit / s	8 µs	1.5 µs	500 ns	700 m
50 Kbit / s	16 µs	3 µs	1 µs	1000 m
20 Kbit / s	40 µs	7.5 µs	2.5 µs	1500 m
10 Kbit / s	80 µs	15 µs	5.0 µs	2000 m

The values Tsync, Tseg1 and Tseg2 refer to the CAN specification of the CAN controller used, they signify time intervals within a bit from the CAN telegram.

Tsync: Time of the synchronization segment.

Tseg1: The sum from ‘propagation time segment’ and ‘phase buffer segment 1’. Tseg 1 is the time between the Sync-segment and the sample point of a bit.

Tseg2: The ‘remaining’ time after the sample point (‘phase buffer segment 2’) to the beginning of the next bit.

Attention: The maximum reachable bus length of the CAN-CBM-DIO8, CAN-CBM-AI4, CAN-CBM-AO4, CAN-CBM-DIO8-Counter and CAN-CBM-OP1 modules is shorter than the reachable bus length of other **esd** CAN modules because of the type of optocoupler in the CAN interface of these modules.
This has to be taken into account at the network planning and installation!

7.2 CAN Channels

The following channels are available for the data transfer between the control (e.g. motor controller) and the I/O-units:

- a process-data object (PDO) with 8 to 32 bits, depending on the number of I/O-slave modules
- a parameter channel (SDO) to access manufacturer-specific code positions

Depending on the requirements up to four I/O-modules with 8 I/O-connections each can be linked via the CAN to form a 'big' module with 32 I/O-connections. Modules linked in such a way will be called 'I/O-master module' and 'I/O-slaves' in this manual. The I/O-master module is not to be confused with a CANopen master! For such a module the following channels are used:

- an internal parameter channel from the I/O-master module to its I/O-slave modules
- a maximum of three internal parameter and process-data objects from the I/O-slave module to the I/O-master module

Before using the internal module compound, you have to check whether the required identifiers are really available in the CAN network (see appendix, starting on page 47).

7.3 Process-Data Objects

Each I/O-module configured as master is connected to the control or the motor controller via a process-data object (PDO).
For the I/O-module each process-data object consists of a receive branch (PDO-RX) and a transmit branch (PDO-TX).
Via the process-data object the I/O-module receives programs to enable its digital outputs and transmits the status of its digital inputs.
In 'DS-401'-mode the contents of the process-data telegrams can be set flexibly. More on this can be taken from chapter 'PDO Mapping' starting on page 40.
In servo mode the PDO mapping is configured in default setting in such a way that the CAN telegrams have the following structure:

process-data telegram (e.g. address 2) transmitted by the I/O-module in servo mode

Identifier	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8
768+2 = 770	input 1...8	input 9...16	input 17...24	input 25...32	not used (0)	not used (0)	not used (0)	not used (0)

Inputs 1...8 are the I/O-connections of the master module, inputs 9...16 are at slave no. 1, inputs 17...24 are at slave no. 2, inputs 25...32 are at slave no. 3.
Input 1 is inserted into the LSB of data 1, input 8 into the MSB, etc.
If some slaves are not available, the according inputs transmit the value 0.
The transmission of digital input information is initiated by the I/O-module in servo mode, event-controlled and/or time-controlled transmissions are possible (see chapter 'Parameterization', starting on page 26).

process-data telegram (e.g. address 2) received by the I/O-module in servo mode

Identifier	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8
767+2 = 769	output 1...8	output 9...16	output 17...24	output 25...32	not used (0)	not used (0)	not used (0)	not used (0)

This telegram is equally structured than the transmitted telegram.
If a bit is set, the according output is enabled (provided that it is defined as an output).
Attempts to enable connections which have not been defined as outputs are ignored by the I/O-modules.

7.3.1 Parameter Channel

Each I/O-master module can be configured by a parameter channel. Important parameters are transmitted from the I/O-master module to the I/O-slaves. A detailed description of the parameter channel will follow on page 28.
For standard applications the use of the parameter channel is not necessary, because the most important parameters can also be set via the DIP-switch.

7.3.2 Internal Master-Slave Communication

For the communication between I/O-master module and its I/O-slave modules further CAN identifiers are covered which can be taken from the appendix
For the CAN transmission from the I/O-slaves to the I/O-master each slave has its own identifier (because 'quick' process data which must not collide has also to be transmitted via this identifier).

The telegrams for the monitoring of the transmission from the slave modules to the master module are structured as follows:

Identifier	**Data 1**	**Data 2**	**Data 3**	**Data 4**	**Data 5**	**Data 6**	**Data 7**	**Data 8**
896 ... + x	ident. (see below)	index (low)	index (high)	sub-index	user data	user data	user data	user data

For identification the following codes are used (following the SDO channel, which, however, has nothing in common with the 'internal' parameter channel):

Code	**Meaning**
0x00	telegram with process data and status
0x40	read command
0x2B	write command with data (maximum 32 bits)
0x4A	response to read command with user data (maximum 32 bits)
0x60	acknowledgment for write command
0x80	acknowledgement: „error“

Only one identifier is used for the CAN transmission from the I/O-master to its I/O-slaves. The slave number is written into the CAN-user data in order to 'save' identifiers. This way parameters and process data are transmitted which the I/O-master module passes on to its I/O-slaves after analysing the PDO-telegrams.

If the 'internal' communication between I/O-master module and I/O-slave module is disturbing or is not required, it can be switched off by a special programming option by the DIP-switch (see page 27). This might be necessary, if no identifiers for the internal master/slave communication are available in the CAN network.

8. Parameterization

The module can be parameterized without external aids (bit rate, CAN addresses).
Connection to the power supply is only needed for programming the inputs and outputs and further internal parameters.

8.1 Parameterization Via DIP-Switch

8.1.1 Assignment of the DIP-Switch (in normal status):

Switch number	Function	Explanations
1	CAN address (bit 0)	OFF=0, ON= 1
2	CAN address (bit 1)	OFF=0, ON= +2
3	CAN address (bit 2)	OFF=0, ON= +4
4	CAN address (bit 3)	OFF=0, ON= +8
5	CAN address (bit 4)	OFF=0, ON=+16
6	CAN address (bit 5)	OFF=0, ON=+32

7 (*)	master/slave number (L)
	OR
	CAN address (bit 6)
		OFF=0, ON= 1
		
		OFF=0, ON=+64

8 (**)	master/slave number (H) OR reserve	OFF=0, ON= +2
9	CAN-bit rate (bit 0)	
10	CAN-bit rate (bit 1)	
11	CAN-bit rate (bit 2)	
12	programming switch	normal status ‘OFF’

The CAN address results from the addition of the squares (see column 'Description').

(*) The assignment of switch 7 depends on the choice whether the I/O-modules are to operate with 'internal master/slave cascading' (see page 26) or not. The setting whether the modules operate with or without cascading can be made in the programming mode (see page 25).
With 'internal cascading’ only 63 different node identifiers are possible, without 'internal cascading' however 127.

(**) If the 'internal master/slave cascading' is not used, switch 8 has no function at the time being. It should be set to 'OFF' so that it does not collide with possibly future extensions.

8.1.2 Setting the CAN-Bit Rate

Switch 9	Switch 10	Switch 11	CAN-bit rate
OFF	OFF	OFF	1000 Kbit / s
OFF	OFF	ON	500 Kbit / s
OFF	ON	OFF	250 Kbit / s
OFF	ON	ON	125 Kbit / s
ON	OFF	OFF	100 Kbit / s
ON	OFF	ON	50 Kbit / s
ON	ON	OFF	20 Kbit / s
ON	ON	ON	10 Kbit / s

8.1.3 Test Mode (only for the manufacturer)

The test mode is used for a fast test of the I/O-connections and for the default setting. Current information can be taken from the special document.

In order to activate the test mode, all DIP-switches have to be set to 'ON' before switching on the supply voltage and then (max. 2 seconds after a power-on) one of the switches 8...12 has to be set to 'OFF' again. The I/O-module then switches into test mode, which can only be left again by switching off the supply voltage. Furthermore **all EEPROM cells are set to the default values**.

In test mode the two last DIP-switches define the test which is to be carried out:

S11 ON, S12 OFF: ‘running’ test of all LEDs and reading back the digital inputs (directly after activating the test). If there is an error at reading-back, the red module-error LED is additionally driven.

S11 OFF, S12 OFF: reserved

S11 ON, S12 ON: reserved

S11 OFF, S12 ON: ‘DIP’-test: Switches 1...10 are linked by software to the outputs 1...8 and both error LEDs.

Manufacturer-default settings which have been implemented previously
(during power-ON all DIP-switches ON, then after about 2 seconds...):

Switch 12 OFF: ‘Servo’ mode, no A/D-converter installed

Switch 11 OFF: ‘DS401’, no A/D-converter installed

Switch 10 OFF: ‘DS401’, 1 channel A/D + D/A, voltage feed (±10 V)

Switch 9 OFF: ‘DS401’, 1 channel A/D + D/A, current feed (±20 mA)

Switch 8 OFF: ‘DS401’, input 1 and 2 as counter inputs (C0256)

8.1.4 Programming Mode

As long as the programming switch (S12) is in 'OFF' position, the module is in 'normal operation'. By switching S12 to 'ON' the module gets into 'Programming' mode, no matter which current operating mode it had been in.

If the programming switch is already on 'ON' during power-ON, the default settings become active and all possibly stored 'special configurations' will be overwritten. If the programming switch is only put to 'ON' after power-on, however, the parameters which can only be set via SDO remain stored.

The programming mode is signalized as follows:

- the green operation LED is switched off
- the green mode LED is permanently on
- the red module-error LED blinks slowly as long as the programming switch is 'ON'

In programming mode all outputs are switched off.
During this phase the switches S1 ... S9 can be set in accordance with the following chapters. Each time the switch is turned the green operation LED blinks.

Only when the programming switch is turned from ‘ON’ to ‘OFF’ the set configuration (definition of inputs and outputs) is taken over into a non-transient memory.

After a successful programming the blinking interval of the red module-error LED becomes shorter. For safety reasons the programming mode can only be left by switching off the supply voltage.
Before switching-on the supply voltage again, the DIP-switches for the address setting and the CAN bit rate (S1...S11) have to be set to their current values again!

All parameters which can be set via DIP-switch in the programming mode can also be set via SDO (see page 28). For this, however, you need a suitable configuration tool.

8.1.5 Assignment of the DIP-Switch in Programming Mode

Note: The Assignment of the DIP-Switch in Programming Mode is not valid for the CBM-AO4 module and the CBM-AI4 module (see 'CBM-AI4 manual supplement)

Switch number	Function	Description

1	port direction I/O 1	OFF = input,
		ON = output

2	port direction I/O 2	...
3	port direction I/O 3	...
4	port direction I/O 4	...
5	port direction I/O 5	...
6	port direction I/O 6	...
7	port direction I/O 7	...
8	port direction I/O 8	...

9	CAN-ID basis, software model	OFF = servo mode
		ON = DS401-mode

10	internal I/O-cascading (I/O-M/S compound)	OFF = cascading ON = no cascading.
11	'reserve'	should remain 'OFF'!
12	programming switch	normal status 'OFF'

8.1.5.1 Programming the Input/Output Functions

In programming mode the switches 1 to 8 of the DIP-switch have the following special functions: The switch position 'ON' defines the according I/O-line as output, the position 'OFF' defines an input. For the I/O-module 'DIO8' you have to observe the notes in chapter 'Digital Outputs' on page 10!

8.1.5.2 Programming the CAN Identifiers and the Software Model

In programming mode switch 9 has the following meaning:

Switch 9 'OFF': The CAN identifiers listed in the appendix starting on page 47 are used ('servo mode').

Switch 9 'ON': *For the time being* CAN identifiers from the 'Predefined Connection Set' are used ('DS 401'-mode, see appendix).

8.1.5.3 Inhibiting the I/O-Master/Slave Compound

If the compound of the I/O-master to three I/O-slaves is not desired, it is possible to inhibit this functionality. This can be achieved by switch 10 in programming mode:

Switch 10 'OFF': Linking an I/O-master module with a maximum of three slave modules is possible (this is normal).

Switch 10 'ON': An I/O master-Slave compound is not possible (special case, only required if the 'internal' transmission between I/O-master and I/O-slaves as described on page 26 is disturbing).

8.1.5.4 Examples for Programming by the DIP-Switch

Servo mode with all I/O-pins as outputs:

- voltage supply off
- DIP-switches 1 to 8 'ON'
- DIP-switches 9 to 11 'OFF'
- DIP-switch 12 'ON'
- voltage supply on: red LED is slowly blinking
- DIP-switch 12 'OFF': red LED is slowly 'flashing' (programming is finished)
- voltage supply off again
- before switching on again:
  set device number and addresses again (see pages 22, 23) !

DS401-mode without internal I/O-cascading, all I/O-pins as inputs:

- voltage supply off
- DIP-switches 1 to 8 'OFF'
- DIP-switch 9 'ON'
- DIP-switch 10 'ON'
- DIP-switch 11 'OFF'
- DIP-switch 12 'ON'
- voltage supply on: red LED is slowly blinking
- DIP-switch 12 'OFF': red LED is slowly 'flashing' (programming finished)
- voltage supply off again
- before switching on again:
  set device number and addresses again (see pages 22, 23) !

8.2 Parameterization by the Parameter Channel (SDO)

By means of a PC with CAN interface and configuration tool or another control system the most important internal parameters of the I/O-module can be configured. For this an SDO-channel is available in each I/O-module defined as master. This SDO-channel will be explained in detail in chapter ‘CANopen’.

8.2.1 Access to I/O-Code Positions Via the Parameter Channel

The I/O-module uses its own ‘code positions’ for accessing internal parameters. These code positions can be taken from the appendix. The subindex is not used within this ‘manufacturer-specific profile’ (CANopen designation) and the according telegram field should be set to ‘0’.

8.2.1.1 Time Performance at SDO-Write Accesses

Write accesses to I/O-modules without ‘internal master/slave compound’ are acknowledged within 30 ms, if ‘automatic storing’ is used.

A ‘normal’ write access is much quicker acknowledged, if *no* automatic storage is uses. In this case a write access is acknowledged within 10 ms. If you want to retain the parameters even after switching off the supply voltage, they have to be saved by a special ‘save’ command to CANopen object 1. The time the ‘save’ command needs to be executed depends on the number of changed memory cells in the EEPROM. In the worst case the EEPROM used needs up to 20 ms per write access, however, such long delay times have never been witnessed in tests.

An SDO-write access can take especially long time (100 ms), if you work with ‘internal master/slave compound’ and code positions have been written on which have to be passed from the I/O-master module to its I/O-slaves first. The internal procedure during this access will be explained in the following chapter.

8.2.1.2 Internal Procedure During SDO-Write Accesses

The following example is used to explain the ‘most difficult’ internal procedure during SDO-write accesses with ‘internal master/slave compound’:

Example 1: Definition of 32 inputs/outputs via the parameter channel

All I/O-lines of the master module and the three slave modules connected to it are to be defined as outputs. The device address of the master module is 2.

The appropriate code position 10_{dec} (port direction) can be taken from the appendix.
From this the following CAN (SDO) telegram to be transmitted from the control system results:

identifier	command code	index (low)	index (high)	subindex	data 1	data 2	data 3	data 4
1536 + 2 = 1538	(write) 0x23	0xFF -10_{dec} = 0xF5	0x5F-0 = 0x5F	0x00	(master) 0xFF	(slave1) 0xFF	(slave2) 0xFF	(slave3) 0xFF

Only the I/O-master module with device address 2 receives this telegram. Since this telegram contains also data for the slaves connected to the master, however, the master then transmits according telegrams to its slaves via the internal master->slave channel (identifiers 1664+2) and waits for the response of the slaves.

Note: (Provided that not all three slaves are available, the master stops waiting after a certain time (50 ms), but does not report an error!).

The slaves store their individual port-direction definitions into the non-transient memory and acknowledge this after successfully storing via the slave->master channels.
Afterwards the master, too, stores its own ‘new’ port-direction definition into the non-transient memory. This procedure requires also some time (max. 20 ms).
Only after completing programming, the I/O-master module reports the acknowledgement to the control system:

identifier	command code	index (low)	index (high)	subindex	data 1	data 2	data 3	data 4
1408 + 2 = 1410	(w.resp) 0x23	0xFF -10_{dec} = 0xF5	0x5F-0 = 0x5F	0x00	0x00	0x00	0x00	0x00

The actual reaction time can only be estimated, because it depends on the following factors, for instance:

- CAN bit rate
- CAN load and part of CAN telegrams with a higher grade of priority
- CPU-demand in the I/O-modules by other tasks, e.g. fast status changes at the digital inputs.

8.2.2 How to Use the Parameter Channel

Generally the following things have to be observed when using the SDOs:

- SDOs are used to configure the I/O-modules. They must not be used to change internal parameters during 'normal' operation. This would reduce the lifetime of the EEPROMs used for storing!

- When changing a module configured via SDO or using the module in a different place, the configuration should be set to its default values again. This concerns especially the change in I/O-directions mentioned above.

- For parameters which can be configured via DIP-switch *and* via SDO the setting which had been executed last applies.

- Changes of identifier-'basis addresses' by means of write accesses to code positions starting from 50 become only active after initializing the modules again for instance by a network management or by switching off the supply voltage. This possibility of 'redefining' the CAN-ID basis addresses is only to be used, if the addresses cannot be set anymore by means of the DIP-switch!

- Accessing the code positions does not depend on the status of the module (operational, preoperational).

- When linking modules to one 'logical' module, the I/O-slave modules should also be connected when parameterizing the I/O-master module, because at SDO accesses it exchanges several parameters via the internal parameter channel with its I/O-slave modules (e.g. the port-direction definition).

- Only if serious errors occurred an SDO access is not possible anymore.

- The automatic storing of parameters into the EEPROM after each write access can be suppressed via access to a manufacturer-specific code position. This can be requested in object 1010_h in the CANopen communication profile. The automatic storing of parameters is default-set, however, in other words, no 'save' command has to be transmitted.

9. CANopen

This chapter offers fundamental information on 'CANopen' and the most important information on the functions implemented.

A complete CANopen description would be too extensive for this manual.
Interested readers are advised to refer to the CAL/CANopen manual, which is not easy to understand, however.

9.1 The CANopen Object Directory

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

The index can be a 16-bit parameter according to the CANopen specification (CiA-Draft DS301, DS401) or a manufacturer-specific code. By means of the MSB 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	5FF5 = port directions
6000 ... 9FFF	standardised device profile area	(here : according to DS401) 6000 = status of 8 inputs
A000 ... FFFF	reserved	

9.1.1 Communication Profile Area

Within this passage the following objects are implemented :

Index (hex)	Name	Subindices	Type	Access
1000	Device Type	-	Unsigned32	ro
1001	Error Register	-	Unsigned8	ro
1003	Predef'd Error... (Error History)	variable	Unsigned32	r, w (only subindex 0)
1004	Number of PDOs	0,1,2	Unsigned32	ro
1005	COB-ID of Sync Message	-	Unsigned32	rw (*)
1008	Manufacturer's Device Name	-	Visible String	ro
1009	Manufacturer's Hardware Version	-	Visible String	ro
100A	Manufacturer's Software Version	-	Visible String	ro
100B	Node-ID	-	Unsigned32	ro (=DIP-switch)
100C	Guard Time	-	Unsigned16	rw (*)
100D	Life Time Factor	-	Unsigned8	rw (*)
100	Node Guarding ID	-	Unsigned32	rw (*)
1010	Store Parameters	0,1	Unsigned32	rw
1011 in preparation	Restore Default Parameters	0,1	Unsigned32	rw
1400	Receive PDO Communication Parameter	0,1,2	PDOCommPar	rw
1600	Receive PDO Mapping Parameter	0, 1...max.8	PDOMapping	rw
1800	Transmit PDO Communication Parameter	0,1,2	PDOCommPar	rw
1A00	Transmit PDO Mapping Parameter	0, 1...max.8	PDOMapping	rw

(*) these parameters **are to be** set only by means of the 'extended NMT-boot up'.

9.1.2 Standardised Device Profile Area

The objects implemented are based on the 'CiA Draft Standard Proposal 401' Version 1.4. Some objects, e.g. the 16- and 32-bit-access objects are only available, if several CAN-CBM-DIO8 modules with 8 I/O-connections each are linked to form a compound.

9.1.2.1 Digital Inputs and Outputs

Index (hex)	Name	Subindices	Type	Access
6000	Read State [of] 8 Input Lines	0,1...max.4	Unsigned8	ro
6006	Input Interrupt Mask [for] 8 Input Lines, any change	0, 1...max.4	Unsigned8	rw
6020	Read State [of] 1 Input Line	0, 1...max.32	Unsigned8, Boolean	ro
6100	Read State [of] 16 Input Lines	0, [1, 2]	Unsigned16	ro
6120	Read State [of] 32 Input Lines	0, max. 1	Unsigned32	ro
6200	Write State [for] 8 Output Lines	0, 1...max.4	Unsigned8	rw
6206	Fault Mode [for] 8 Output Lines	0, 1...max 4	Unsigned8	rw
6207	Fault State [for] 8 Output Lines	0, 1...max 4	Unsigned8	rw
6220	Write State [for] 1 Output Line	0, 1... max.32	Unsigned8, Boolean	rw
6250	Fault Mode [for] 1 Output Line	0, 1...max.32	Unsigned8, Boolean	rw
6260	Fault State [for] 1 Output Line	0, 1...max.32	Unsigned8, Boolean	rw
6300	Write State [for] 16 Output Lines	0, 1...max.2	Unsigned8, Unsigned16	rw
6306	Fault Mode [for] 16 Output Lines	0, 1...max 2	Unsigned8, Unsigned16	rw
6307	Fault State [for] 16 Output Lines	0, 1...max 4	Unsigned8, Unsigned16	rw
6320	Write State [for] 32 Output Lines	0, max.1	Unsigned8, Unsigned32	rw
6326	Fault Mode [for] 32 Output Lines	0, max 1	Unsigned8, Unsigned32	rw
6327	Fault State [for] 32 Output Lines	0, max 1	Unsigned8, Unsigned32	rw

9.1.2.2 Analog Inputs and Outputs

(Only CBM-AI4 and CBM-AO4)

Index (hex)	Name	Subindices	Type	Access
6401	Reads value of the input channel (not converted)	0 1	Unsigned 8 Unsigned16	ro
6411	Writes value of the output channel (not converted)	0 1	Unsigned8 Unsigned 16	rw
6420	Set Analogue Input Range	0 1	Unsigned8 Unsigned 16	rw
6421	Determines which events cause an interrupt for a specific channel	0 1	Unsigned 8 Unsigned 8	rw
6423	Globally enable/disable Interrupt	0	Boolean	rw
6424	When enabled, interrupt triggered when analogue input rises above this value (not converted)	0 1	Unsigned8 Unsigned 32	rw
6425	When enabled, interrupt triggered when analogue input falls below this value (not converted)	0 1	Unsigned8 Unsigned 32	rw
6426	When enabled, interrupt triggered when analogue changes by more than this value from previous reading (rising or falling) (not converted)	0 1	Unsigned8, Unsigned 32	rw
6443	Output Fault Mode	0,1	Unsigned8	rw
6444	Default Output Fault value (unconverted)	0 1	Unsigned 8 Unsigned 32	rw

9.1.3 Manufacturer-Specific Profile Area

Important is the manufacturer-specific profile area from 0x2000-0x5FFF, where special system parameters can be read and set.
In the I/O-module CAN-CBM-DIO8 the index is determined by means of the formula

Index = 0x5FFF - code_number_of_the_I/O-module

The code numbers of the I/O-modules can be taken from the appendix on page 47.

9.2 Service-Data Objects (SDOs)

Service-data objects are used to access the object directory of a device.
Therefore an SDO is a channel which is used to access the parameters of the device.
The access via this channel is possible in the I/O-module CAN-CBM-DIO8 in ‘operational’ and ‘preoperational’ status. Please observe the notes on page 30!

9.2.1 Structure of SDO Telegrams

Identifier	command code	index (low)	index (high)	subindex	data 1	data 2	data 3	data 4

The command code consists of the command specifier and the length. Combinations often used are for instance:
0x40 = 64 dec: read request, i.e. a parameter is to be read
0x23 = 35 dec: write request with 32 bit data, i.e. a parameter is to be set

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

The I/O-module responds to **every** telegram received with a response telegram, which can consist of the following command codes:
0x42 = 66 dec: read response, this telegram contains the desired parameter
0x60 = 96 dec: write response, i.e. a parameter has been set successfully
0x80 = 128 dec: error response, i.e. the I/O-module reports a communication error.

Further information can be taken from the CiA-specifications.

9.2.2 Non-Transient Storing of Parameters

All communication relevant parameters (PDO mapping + communication) can be saved in the EEPROM generally. The objects in the device profile, which determine the behaviour of the module in the error case, can be stored to the EEPROM as well. The value of the digital and analog outputs cannot be saved.

At each write access the parameters are stored into a non-transient memory via default-setting without the need to send a special 'save' command to object 1010h. The response to an SDO-telegram then follows only after the non-transient memory has been updated.

If this 'automatic' storing and the delay this is causing should be bothering, the 'automatic' storing can be switched-off by reprogramming the manufacturer-specific code position 17.

The contents of object 1010h represent the current memory mode:

- If bit 1 and 2 are set subindex 1 by object 1010_h, the I/O-module automatically stores the configuration after each change via SDO-channel.
- If bit 1 is set subindex 1 by object 1010_h, the I/O-module does not save the configuration 'automatically', but has to be arranged by writing the characters 'save' (73_h 61_h 76_h 65_h, arrangement from CAN telegram) into object 1010_h, subindex 1.

Note: The CANopen standard has no possibility to change into the memory mode. The possibility to change via code position 17 has been designed for a greater flexibility of the I/O-modules.
If many write accesses via SDO are made when booting a network, the 'automatic' saving should be switched off by setting code position 17 to zero and the EEPROM should be updated once after the initialization. By doing this the I/O-module does not always have to determine the EEPROM checksum and the many SDO-write accesses can be acknowledged much quicker (see also page 28).
The EEPROM used has a lifetime of 1 million delete/write cycles.

9.3 NMT-Boot-Up

9.3.1 Basic Boot-Up

The I/O-modules of the ‘DIO8’-series can be initialized with the ‘Minimum Capability Device’-boot-up described in the CiA-Draft Standard 301 in chapter 8.3. The easiest way would be to send a telegram after switching-on to change from preoperational status into operational status. Transmitting the 2-bytes telegram ‘01 00’ (=start remote node) to the CAN identifier 0000_h should be sufficient.

9.3.2 Extended Boot-Up

Additionally the extended boot-up with node guarding activation is available.
For this the NMT-master can transmit the sequence which will be shown on the following page to the NMT-slave.
(here: module-ID = 1, COB-IDs decimal, CAN data hexadecimal):

The parameters entered in the CAN telegrams have the following meaning:

node-ID: ‘node identifier’; in the CAN-CBM-DIO8 this is the ‘CAN address’ set via DIP-switch in the area of 1...63 (without ‘I/O-cascading’ 1...127).

req. guard time: Guard time in milliseconds requested by the slave. The CANbloc-Mini-module CAN-CBM-DIO8 does not request a guard time, and the entry is therefore 0.

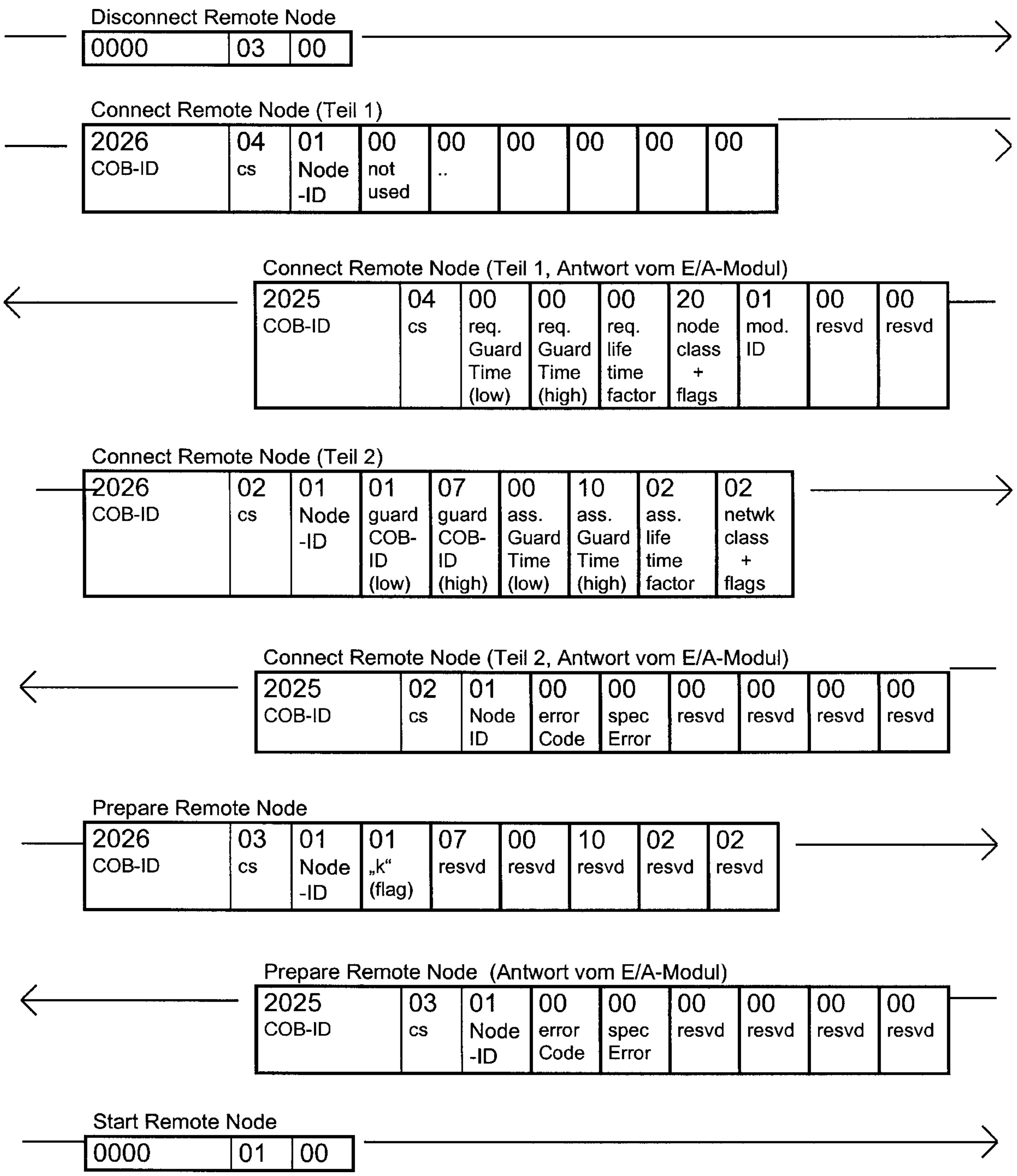
req. lifetime factor: Lifetime factor requested by the slave. Is being overwritten by the assigned lifetime factor.

node class+flags: Represents the features of the I/O-module as NMT-slave. In CAN-CBM-DIO8 ‘2’. In bit 7 of this byte the NMT-slave could request a configuration download. This is not possible on the CAN-CBM-DIO8, and therefore bit 7 is always ‘0’.

mod. ID: In CAN-CBM-DIO8 identical with the node-ID.

guard COB-ID: The CAN identifier for the node-guarding protocol assigned from the master to the slave. Has to be between 1761 and 2015.

NMT-Master NMT-Slave



Disconnect Remote Node

0000	03	00

Connect Remote Node (Teil 1)

2026	04	01	00	00	00	00	00	00
COB-ID	cs	Node -ID	not used	..				

Connect Remote Node (Teil 1, Antwort vom E/A-Modul)

2025	04	00	00	00	20	01	00	00
COB-ID	cs	req. Guard Time (low)	req. Guard Time (high)	req. life time factor	node class + flags	mod. ID	resvd	resvd

Connect Remote Node (Teil 2)

2026	02	01	01	07	00	10	02	02
COB-ID	cs	Node -ID	guard COB-ID (low)	guard COB-ID (high)	ass. Guard Time (low)	ass. Guard Time (high)	ass. life time factor	netwk class + flags

Connect Remote Node (Teil 2, Antwort vom E/A-Modul)

2025	02	01	00	00	00	00	00	00
COB-ID	cs	Node ID	error Code	spec Error	resvd	resvd	resvd	resvd

Prepare Remote Node

2026	03	01	01	07	00	10	02	02
COB-ID	cs	Node -ID	„k“ (flag)	resvd	resvd	resvd	resvd	resvd

Prepare Remote Node (Antwort vom E/A-Modul)

2025	03	01	00	00	00	00	00	00
COB-ID	cs	Node -ID	error Code	spec Error	resvd	resvd	resvd	resvd

Start Remote Node

0000	01	00

After this procedure the I/O-module is in operational status and is ready to transmit and receive process-data objects. Furthermore the node guarding is now active so that the node-guarding master **must** request the module status periodically via RTR-telegram from now on (more details on this in chapter ‘Node Guarding’).

9.4 Process-Data Objects (PDOs)

PDOs are used for the transmission of process data.
The ‘receive’ PDO normally transmits the data for the digital outputs of the control system (e.g. static frequency changer) to the I/O-modules.
The‘transmit’ PDO transmits the status of the digital inputs of the I/O-master module (which is always a SLAVE in the CANopen network!) to the control system (or - in special cases - to other I/O-devices).

The PDO structure has already been explained for an easy example in chapter ‘Process-Data Objects’. For general use a PDO telegram can be structured in (nearly) any way, however. For this the PDO mapping, which will be described in the following chapter, can be used.

9.4.1 PDO Mapping

The variable PDO mapping is used to 'insert' certain objects from the CANopen-object directory into a maximum 8 bytes long process-data telegram.
For this, object 1800_h is responsible for the receive PDO (digital inputs) and object $1A00_h$ is responsible for the transmit PDO (digital outputs). Both objects have principally the same structure. These PDO mapping parameter objects comply with a list in which all objects in the PDO are represented.
Each list entry shows the index of a represented object, the subindex of the represented object and the number of data bits inserted by this object in the PDO.

Example:

In the transmit PDO only the status of the first 8 digital inputs of an I/O-module is to be represented. The object which contains the digital inputs in groups of 8 inputs each (i.e. 'byte by byte) is object 6000_h from 'DS 401' (at the moment only 'DSP 401 V1.4', chapter 8.1.1).
Subindex 1 from object 6000h always contains the status of the first 8 digital inputs, if necessary it could also be read out via SDO. In order to map this object into the PDO, index, subindex and bit number have to be combined to a 32-bit value ('unsigned 32') as follows:

MSB, Bits 31...24	Bits 23...16	Bits 15...8	LSB, Bits 7...0
index (high)	index (low)	subindex	number of data bits
60	00	01	08

Before this 32-bit value can be written into the mapping object 1A00h via SDO, the list length of the mapping object has to be set to at least '1'. This is made by writing a '1' into the mapping object 1A00h, subindex 0.
Now there is space in the list for exactly one entry which is then written to subindex 1. Attempting to access a list object which does not fit into the list (because of the contents of subindex 0) causes an error response during SDO access to the mapping object.
After successfully programming the mapping object it has the following contents:

object 1A00h, subindex 0: **01_h** (i.e. one entry in the list)

object 1A00h, subindex 1: **60000108_h** (i.e. represent 8 bits of object 6000_h, subindex 1 in the PDO)

Because of this programming the process-data telegram will only consist of one byte, which contains the current status of the first eight digital inputs.
If further information is to be contained in the remaining 7 bytes, object $1A00_h$ subindex 0 has to be increased (e.g. to 4) first. Then further list entries can be made in object $1A00_h$ subindices 2, 3, 4.
In I/O-modules 'DIO8' a maximum of eight different objects can be represented in a PDO. For reasons of speed the number of data bits has to be eight or an integral multiple from it.

9.4.2 PDO Transmission Types

9.4.2.1 Synchronous Transmission Types

The synchronous transmission rates can be used, if a participant in the CANopen network can generate the according SYNC telegrams. The I/O-module ‘DIO8’ can only receive those telegrams but cannot generate them.
The synchronous transmission type is defined by the ‘PDO transmission type’ in the communication parameter of the according process-data object.
A transmission type of 5 means that e.g. after five SYNC massages received a process-data object is transmitted. Details on this can be taken from the CiA Draft Standard 301.

9.4.2.2 Event-Controlled Transmission

Each change of a digital input at master or slaves can trigger a message of the digital inputs.
Default-set this is made by all equipped inputs, by changing an event mask this can be changed (see appendix ‘Code Table’ or DS401).
If this event mask is set to 0xFFFFFFFF, all inputs trigger a message. If this mask is set to 0x00000000, no event-controlled transmission is made.
In order to prevent a too high busload caused by event-controlled transmission, two mechanisms have been implemented:

- debouncing the digital inputs (with debounce time which can be set)
- definition of a minimum break which has to pass between the transmission of two event-controlled (!) telegrams.

The default values are:
debounce time 1 ms

- no minimum break between two event-controlled transmissions

9.4.3 Cyclical Asynchronous Transmission

The input information can be transmitted cyclically (for instance every 100 ms) with the process-data object. The transmission cycle time can be changed via the parameter channel or be switched off.
It is an **additional** transmission initiative to the possibilities defined in the CANopen standard and has been especially designed for the use at the servo-motor controller. In CANopen terminology we could refer to this transmission type as ‘cyclical but not dependent on the SYNC object’.

Default value:

- ‘Servo mode’: Cyclical transmission every 100 ms
- ‘DS 401 mode’: Cyclical transmission switched off (0), transmission only depending on ‘PDO transmission type’.

This transmission type can be activated by changing code position 14 (in the manufacturer-specific object area, see appendix).

9.4.4 Node Guarding / Life Guarding

Via the node/Life guarding the control and the connected I/O-modules monitor each other (especially in order to recognize connection failures).
In CANopen terminology the node guarding is the monitoring of the I/O-modules by the NMT-master, the Life guarding, on the other hand, occurs within the I/O-modules and monitors the NMT-master.

Node guarding:

> The NMT-master requests cyclically a special telegram from its NMT-slaves (here: = I/O-modules)by means of a special RTR-message (CAN request). If the response of the slaves does not correspond to the response expected, the NMT-master recognizes an error and reacts accordingly. The response telegram contains the module status and a toggle bit. Identifier and cycle time are given to the NMT-slave during the extended NMT-boot up.

Life guarding:

> The NMT-slaves monitor whether they are being requested by the NMT-master for the purpose of the node guarding. If these requests fail to appear for some time, the slave concerned transmits an emergency message. In the CAN-CBM-DIO8 the nonappearance of the node-guard requests is also shown by the red module-error LED (of course only, if the node guarding has been activated).
> The time until the error is triggered is the product from the node guarding-cycle time and the so-called **lifetime factor**, which is also given to the NMT-slave during the extended-NMT-boot up.

9.4.5 The Emergency Telegram

In case of an internal error the I/O-module transmits an ‘emergency telegram’.
This is structured as follows:

Emergency Telegram

128 + ADR COB-ID	00 error code (low)	23 error code (high)	00 error regis-ter	00 manu. spec. field	00 manu. spec. field	00 manu. spec. field	00 manu. spec. field	00 manu. spec. field

The I/O-module CAN-CBM-DIO8 supports (at least) the following (emergency-) error codes:

Error Code (hex)	Meaning
00xx	error disappeared or no error
10xx	general error
23xx	output current too high
50xx	module-hardware error
60xx	module software error
70xx	error in additional modules
80xx	monitoring
81xx	communication error

In field ‘error register’ of the emergency telegram the current contents of CANopen object 1001_h is inserted.
The ‘manufacturer-specific error field’ is not used at the moment (10/2000).

Via the transmitted emergency telegrams an error history is stored in the I/O-module. For this object 1003_h in the CANopen-object directory is used.

10. Quick Start

10.1 Configuration for the Use in the CANopen Network

The following steps must be followed for a quick start with the most basic configuration:

1. Definition of the outputs of the I/O-module via DIP-switch

- Switch on power supply of the I/O-module.
- Switch 12 ON. Red module-error LED is slowly blinking. (programming mode)
- Switch 11 OFF (future extensions passive)
- Switch 10 ON (I/O-cascading passive)
- Switch 9 ON (‘DS401-mode’)
- Switches 1...8 in accordance with I/O-direction: OFF=input, ON=output
- Switch 12 OFF. The blink pulses of the red module-error LED become shorter.
- Switch off module supply. The definition of the inputs and outputs is completed.

2. Setting CAN address (=’node ID’) and CAN-bit rates at I/O-module
(e.g.: 125 kbit/sec, CAN address 2)

- Switch 1 OFF, switch 2 ON,
- Switches 3 to 6 OFF (CAN address, see also pages22, 23)
- Switch 7 OFF, switch 8 OFF
- Switch 9 OFF, switch 10 ON, switch 11 ON (125 kbit/s, see also page 23)
- Establish CAN connection to control
- Switch on module supply.
  Mode display (green) constantly on, operation display (green) blinks until ‘start node’ had been received.

10.2 Configuration for the Use in CANopen Network

The following steps have to be followed for a quick start with the most basic configuration:

1. Definition of the outputs of the I/O-module via DIP-switch:

- Switch on power supply of the I/O-module.
- Switch 12 ON. Red module-error LED is slowly blinking. (programming mode)
- Switch 11 OFF (future extensions passive)
- Switch 10 OFF (I/O-cascading active)
- Switch 9 OFF (‘servo mode’)
- Switches 1...8 in accordance with I/O-direction: OFF=input, ON=output
- Switch 12 OFF. The blink pulses of the red module-error LED become shorter.
- Switch off power supply of the module. The definition of the inputs/outputs is completed.

2. Setting CAN address, master/slave number and CAN-bit rate in the I/O-module

(e.g. 500 kbit/sec, motor controller at CAN address 1, I/O-master module at CAN address 2) (standard setting for servo-motor controller 93xx)

- Switch 1 OFF, switch 2 ON, switches 3 to 6 OFF (CAN address, see also page 22, 23)
- Switch 7 OFF, switch 8 OFF (module becomes I/O-master)
- Switch 9 OFF, switch 10 OFF, switch 11 ON (500 kbit/s, see also page 23)
- Establish CAN connection to motor controller.
- Switch on power supply of the module. Mode display (green) constantly on, operation display (green) is blinking.

3. Settings at the motor controller

(see ‘Configuration’, ‘Systembus’ in system manual of the servo-motor controller)

- Check whether the bit rate of the motor controller corresponds with the bit rate of all I/O-modules (C0351).
- Make sure that the motor controller transmits or receives with CAN-IN3 and CAN-OUT3 in accordance with the CAN identifiers defined in the appendix.
- Set transmit cycle for CAN-OUT3 telegram to 100 ms to prevent the monitoring in the I/O-module from reporting errors.
- Configure motor controller for CAN master operation (C0352).

10.3 Table of the Most Important Identifiers and Messages for CANopen

Identifiers [HEX]	Designation	Length	Data [HEX]	Description
$0	NMT	2	01 00	start to all (preoperational -> operational)
$0	NMT	2	80 00	operational -> preoperational
$80	NMT	0	-	sync to all
$80 + NodeNo	SDO	0...8 bytes	user data	emergency message from CBM
$600 + NodeNo	SDO	0...8 bytes	user data	to the CAN-CBM-DIO8 (Rx)
$580 + NodeNo	SDO	0...8 bytes	user data	to the CAN-CBM-DIO8 (Tx)
$200 + NodeNo	PDO	0...8 bytes	user data	to the CAN-CBM-DIO8 (Rx)
$180 + NodeNo	PDO	0...8 bytes	user data	to the CAN-CBM-DIO8 (Tx)

Explanation:

NodeNo. ...	1...127 (decimal)
Emergency-Id...	emergency data object
NMT...	network management (master)
PDO...	process data objects
Rx...	receive
SDO...	service data object
Sync...	sync (frame) telegram
Tx...	transmit

11. Appendix

11.1 CAN Identifiers Used

(‘add’ = device address in the area of 1...63 or 1...127 as set by means of the DIP-switch)

‘DS401’ mode with internal I/O-master/slave cascading
(predefined connection set according to DS301 V3.0 chapter 8.4.1)

Function	Identifier	Identifier max	Number
network management (basic, NMT-master ® NMT-slave)	**0**	**0**	**1**
network management (ext., NMT-slave ® NMT-master)	**2025**	**2025**	**1**
emergency message	**128 + Adr**	**191**	**63**
PDO (transmit) (e.g. digital inputs 1...32) from I/O-master module for control	**384 + Adr**	**447**	**63**
PDO (receive) (e.g. digital outputs 1...32) from control to the I/O-modules	**512 + Adr**	**575**	**63**
internal messages from slave 1 to I/O-master	**896 +Adr**	**959**	**63**
internal messages from slave 2 to I/O-master	**960 + Adr**	**1023**	**63**
internal messages from slave 3 to I/O-master	**1024 +Adr**	**1087**	**63**
internal parameter channel from I/O-master module to its slaves 1...3	**1664 + Adr**	**1727**	**63**
SDO (tx) from I/O-master module to the control system	**1408 + Adr**	**1471**	**63**
SDO (rx) from guidance system to the I/O-master module	**1536 + Adr**	**1599**	**63**
node guarding (identifier distribution only by NMT-boot up)	**1792 + xxx**	**2015**	**255**

‘DS401’ mode without internal I/O-master/slave cascading (default setting)
(predefined connection set according to DS301 V3.0 chapter 8.4.1)

Function	**Identifier**	**Identifier max**	**Number**
network management (basic, NMT-master ® NMT-slave)	**0**	**0**	**1**
network management (ext., NMT-slave ® NMT-master)	**2025**	**2025**	**1**
emergency message	**128 + add**	**255**	**127(*)**
PDO (tx) (e.g. digital inputs 1...8)	**384 + add**	**511**	**127**
PDO (rx) (e.g. digitaloutputs 1...8)	**512 + add**	**639**	**127**
SDO (tx) from I/O-master module to the control system	**1408 + add**	**1535**	**127**
SDO (rx) from the control system to the I/O-master module	**1536 + add**	**1663**	**127**
node guarding (identifier distribution only by NMT-boot up)	**1792 + xxx**	**2015**	**255**

(*) 127 node-IDs are only available, if the ‘internal master/slave cascading’ is not used (that means if no modules are linked to form a ‘logical module’).

‘Servo’ Modus

Function	Identifier	Identifier max	Number
network management	**0**	**0**	**1**
I/O - inputs (inputs 1...32) from I/O-master module to motor controller	**767 + add**	**831**	**63**
I/O - outputs (outputs 1...32) from motor controller to I/O-modules	**768 + add**	**830**	**63**
internal messages from slave 1 to I/O-master	**896 +add**	**959**	**63**
internal messages from slave 2 to I/O-master	**960 + add**	**1023**	**63**
internal messages from slave 3 to I/O-master	**1024 +add**	**1087**	**63**
internal parameter channel from I/O-master module to its slaves 1...3	**1664 + add**	**1727**	**63**
SDO (Tx) from I/O-master module to control system	**1408 + add**	**1471**	**63**
SDO (Rx) from control system to I/O-master module	**1536 + add**	**1599**	**63**

11.2 Code Table

Code positions for the fundamental module configuration

Code No. [dec]	**CANopen-Index [hex]**	**Contents**	**Data type [unit]**	**Access r = read w= write**	**Setting / display possibilities**	**Default settings**	**Explanations, references**
5	5FFA	boot up time	16 bit, [ms]	r,w	0...65000 ms, 0xFFFF = no boot message	0	time interval between power-ON and transmitting the boot message
10	5FF5	port direction definitions	32 bit	r,w (*)	bit vector: 0 = input, 1 = output	00000000b	LSB=master MSB=slave3
11	5FF4	debounce time for digital inputs	8 bit, [ms]	r,w	0=off ... 255 ms	1 ms	
12	5FF3	event mask for digital inputs	32 bit	r,w	bit vector: 0 = no IRQ 1 = interrupt	1111...1111 bin	similar to CANopen object 6006_h
13	5FF2	filter constants of the analog inputs	16 bit	r,w	see manual supplement for analog inputs	0 = no filtering	
14	5FF1	transmit cycle of the digital and analog inputs	16 bit, [ms]	r,w	0 = only by event ... 65000 ms	100 ms	independent from the transmission break
15	5FF0	minimum transmission break during event-controlled transmission	8 bit, [ms]	r,w	0 = no break ... 255 ms	0	
16	5FEF	enable/disable internal master/slave compound	8 bit	r,w	0 = no cascading 1 = cascading possible	1	
17	5FEE	activate ‘automatic’ configuration storing	8 bit	r,w	0 = storing only after instruction 1 = storing after each writing	1	see also CANopen object 1010h
18	5FED	hardware model	8 bit	r...			only for manufacturer
19	5FEC	model options	8 bit	r...			only for manufacturer

(*) writing into this code position remains inactive until rebooted.

Code positions for the definition of monitoring and display times of the module without influence on the normal operation

Code No. [dec]	CANopen-Index	Contents	Data type, [unit]	Access r = read w= write	Setting / display possibilities	Default settings	Explanations, references
30	5FE1	PDO-RX monitor (monitoring time when no Life guarding)	16 bit, [ms]	r,w	0= no monitoring ... 25500 ms	0	resolution 100 ms

Code positions for definition of CAN-identifier basis addresses for which the device address set at the DIP-switch (area 1..63) has still to be added.

Code No. [dec]	CANopen-Index [hex]	Contents	Data type [unit]	Access r = read w= write	Setting / display possibilities	Default settings	Explanations, references
50	5FCD	PDO-RX	16 bit	r,w (*)	see CANopen	depending on mode (servo, DS401)	(dig. outputs)
51	5FCC	PDO-TX	16 bit	r,w (*)	see CANopen	dep. on mode	(dig. inputs)
52	5FCB	slave1	16 bit	r	-	896	
53	5FCA	slave2	16 bit	r	-	960	
54	5FC9	slave3	16 bit	r	-	1024	
55	5FC8	master->slaves	16 bit	r	-	1664	

Code positions for purposes of displaying and diagnosing

Code No. [dec]	CANopen-Index [hex]	Contents	Data type [unit]	Access r = read w= write	Setting / display possibilities	Default settings	Explanations, references
93	5FA2	drive ident	16-bit number	r	e.g. DIO8		
99	5F9C	B/W version	string[4]	r	e.g. ‘0.01’		
101	5F9A	internal error and status codes	32 bit	r	(1) EAM_error_flags (2) EAM_slave_errors (3) IOP_status (4) CAL_comm_pha.	-	internal errors and status only for internal use
102	5F99	connected slaves	16 bit	r	bit vector: 0 = recogn. no slave +1 = slave 1 ok, +2 = slave 2 ok, +4 = slave 3 ok	-	the test is only carried out when reading of this code position

Code No. [dec]	CANopen-Index [hex]	Contents	Data type [unit]	Access r = read w= write	Setting / display possibilities	Default settings	Explanations, references
103	5F98	majority loop speed	16 bit	r	provides loops/sec e.g. 3000	-	depending on CPU-demand
120	5F87	set A/D-converter channel	8 bit	w	0 ...7 (CPU-internal multiplexer number)		
121	5F86	read A/D-converter	10 bit	r	0 = 0 V ... 307 = 1.5 V... 511 = 2.5 V... 1023 = 5.0 V	-	ONLY for the internal 'digital' input levels
123	5F84	CAN register access	8 bit	r	reads content of the 256 CAN controller register		ONLY for diagnosis