ANTAL ELECTRONIC

Field Bus and Communication Technology

Manual

PDP2CAN

Version 3.08

Important Notice

Version 3.08
April 2000

Contents

1 Operational Safety

! Only qualified personell are allowed to install and operate this equipment. Qualified personell are those trained and certified to install, and operate electronic equipment with respect to the valid electronic safety standards to date.

! The module must not be installed and wired while in a powered condition.

! To guarantee functionality of the module proper means of transportation, storage and installation must be observed.

! Use power supplies certified according to the national electronic safety standards.

! Proper connection of all power supply and data line connections to be observed

! When transferring the module from a cold environment to a warm one condensation may occur. The module has to be perfectly dry before installation and use. Do not install the module near open water or where high humidity is present.

! No user servicable parts inside. Warranty void if module is opened and/or tampered with.

2 Introduction

The PDP 2 CAN bus interface allows you to convert data from a PRODIBUS DP ring for a secondary CAN ring and vice versa.
The PROFIBUS side is designed as a DP slave. The interfaces correspond to EN 50170. They are galvanically insulated via DC/DC converters and opto couplers. A C515C micro controller, supported by a SPC3 ASIC (Siemens) is in charge of protocol handling. The DP slave supports the entire DP protocol according to EN50170.
Transfer rates from 9.6 baud to 12 Mbaud are detected automatically.
The amount of input and output information is 320 Byte maximum (this corresponds to 10 CAN modules with 2 PDOs each for sending and receiving each of which is 8 bytes long.)

The CANopen side is designed as a an independent master which can be controlled via PROFIBUS. Up to 15 CANopen slave modules according to CiA standard DS 301 Version 3.0 can be operated within a CANopen network.
All parameterised modules are automatically detected, started, and monitored by the CANopen master. For data exchange, up to 5 sending and receiving PDOs and one emergency message for error messages to the master are used. For parameterisation of CAN modules, any SDOs can be sent and received.
The CAN bus interface corresponds to ISO/DIS 11898 and is galvanically insulated at 1 kV DC. It was designed with the CAN bus driver module 82C250 and the C1515C micro controller's integrated basic CAN controller. The serial interface (RS232, RS422, RS485 or TTY) is also galvanically insulated.

Chapter
3

3 Hardware

3.1 Display Elements and Connections

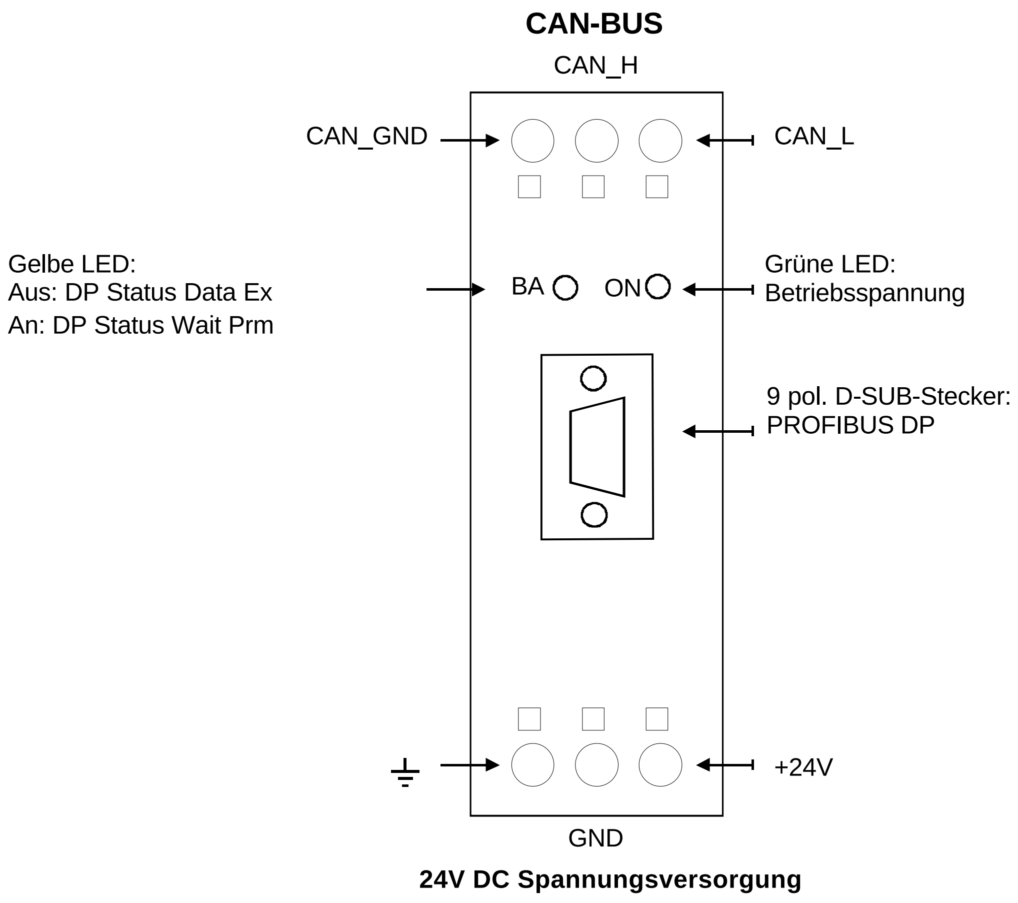


Figure 1 Display and Operation Elements of the PDP 2 CAN

The PDP 2 CAN uses two LEDs as display elements. The green LED indicates correct power supply, and the yellow LED shows the current status of the PROFIBUS.

The D SUB plug is the connection for the PROFIBUS_DP. The three-pin screw-on plugs are used for supplying power and for connecting the CAN bus.

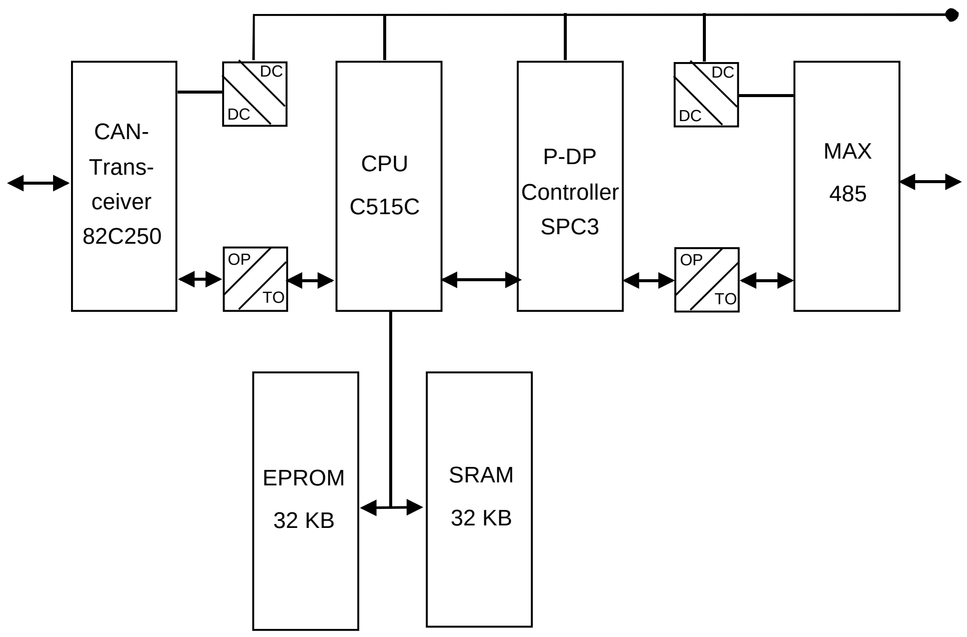


Figure 2: Block diagram PDP2CAN

The CAN bus interface corresponds to ISO/DIS 11898 and is galvanically insulated at 1 kV DC. It was designed with the CAN bus driver module 82C250 and the C515C micro controller's integrated basic CAN controller. A SPC3 by Siemens is responsible for the PROFIBUS protocol handling. It supports the complete PROFIBUS DP protocol according to EN 50170. The interface is galvanically separated via DC/DC converters and opto couplers. Figure 2 shows the block diagram of the PDP2CAN.

3.2 PROFIBUS Bus Interface

Insulation:	1 kV DC via opto coupler and DC/DC converter
Transfer rate:	9,6kBit/s ... 12MBit/s

Allocation of the 9 pin D SUB plug

Pin No.	Allocation
1	n.c.
2	n.c.
3	Data channel B (RxD/TXD P)
4	n.c.
5	DGND (data transfer potential, mass at 5V)
6	VP (voltage of the terminator resistors)
7	n.c.
8	Data line A (RxD/TxD N)
9	n.c.

3.3 DIP switches

The DIP switch allows you to set the PROFIBUS module address and the CAN baud rate. It can be found on the bottom of the module.

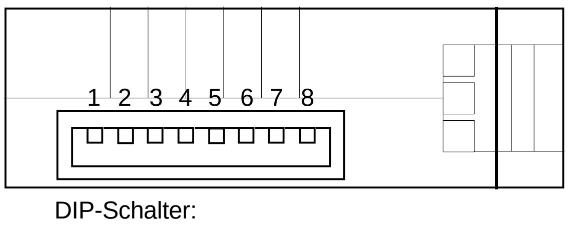


Figure 3 Position of DIP switches

Chapter 4

4 Module configuration

4.1 Switch SW1

The DIP switch allows you to configure the PROFIBUS module address (node address) and the CAN protocol. Switches 1 to 1 set the node address within the range of 1 to 126. Switch 8 configures the CAN protocol (CANopen and CAN layer 2 protocol). They are allocated as follows:

Node Address:

Module ID:	1	2	4	8	16	32	64
Switch No.:	1	2	3	4	5	6	7

The settings “0” and “126” are not valid. If they are set, the default setting 1 will be used.

Switch ON means Bit = logical 1
Switch OFF means Bit = logical 0

CAN Protocol:

Switch S8 sets the relevant CAN protocol. Currently, the CAN layer 2 protocol and the more widely used CANopen protocol is implemented. Further customer specific protocols can be implemented.

Protocol	**S8**
CAN Layer 2	off
CANopen	on

Switch ON means Bit = logical 1
Switch OFF means Bit = logical 0

Chapter
5

5 How it works

The secondary PDP 2_CAN bus interface allows you to convert data from a PROFIBUS DP ring into for a secondary CAN ring and vice versa. Before it is possible to transfer data between the PROFIBUS and CAN, the number of CANopen modules in the CAN ring, their module ID and the length of their respective PDOs must be specified during the parameterisation phase and during the configuration phase. After successful parameterisation and configuration, the CAN modules are started up and monitored. The diagnostic area of the process image shows the status of the CAN modules. Additionally, all emergency messages from the CAN bus are displayed there. In a SDO window in the process image, CANopen modules can be parameterised by means of SDO messages. Input and output data are updated per projected guarding cycle.

5.1 Parameterisation

Master and slave are identified by means of the parameterisation message. They are galvanically insulated via DC/DC converters and opto couplers. A C515C micro controller, supported by a SPC3 ASIC (Siemens) is in charge of protocol handling. The DP slave supports the entire DP protocol according to EN50170.

The parameterisation message identifies master and slave and determines the working mode for the slave. User specific parameters configure the following CAN parameters (shown in an example):

- CAN baud rate
- Synchronous transfer
- Number of CAN nodes in a CAN ring, their module ID and PDO length for sending and receiving

The sequence of parameters in detail:

0x00: The first parameter byte must always be 0x00; it is used internally by the PROFIBUS

CAN Baudrate
The first byte in the parameter message configures the CAN baud rate. The following baud rates are supported:

Value (hex)	Baud rate
00	1000 KBit/s
01	500 KBit/s
02	250 KBit/s
03	125 KBit/s
04	100 KBit/s
05	50 KBit/s
06	20 KBit/s
07	10 KBit/s

SYNC_TIME
Here, the synch time is set in units of 1 ms (10 ms to 64 ms are possible) range (0A..40$_{\text{(hex)}}$). Sync time = 0 means that synchronous mode is not supported.

Number of CAN modules
The number of CAN nodes in the ring is set here (range: 1 .. 15)

Parameter CAN module 1 (Please see chapter 5.1.1 Composition of the parameter byte)

Module ID (Please see chapter 5.1.1 Composition of the parameter byte)
The module ID of the CAN node is entered here. (range: 1..127; see parameter byte).
Length PDO 1 rx (Please see chapter 5.1.1 Composition of the parameter byte)
This sets the length of receiving PDO 1 (see parameter byte).
Length PDO 2 rx (Please see chapter 5.1.1 Composition of the parameter byte)
This sets the length of receiving PDO 2 (see parameter byte).
Length PDO 3 rx (Please see chapter 5.1.1 Composition of the parameter byte)
This sets the length of receiving PDO 3 (see parameter byte).
Length PDO 4 rx (Please see chapter 5.1.1 Composition of the parameter byte)
This sets the length of receiving PDO 4 (see parameter byte).
Length PDO 5 rx (Please see chapter 5.1.1 Composition of the parameter byte)
This sets the length of receiving PDO 5 (see parameter byte).
Length PDO 1 tx (Please see chapter 5.1.1 Composition of the parameter byte)
This sets the length of sending PDO 1 (see parameter byte).
Length PDO 2 tx (Please see chapter 5.1.1 Composition of the parameter byte)
This sets the length of sending PDO 2 (see parameter byte).
Length PDO 3 tx (Please see chapter 5.1.1 Composition of the parameter byte)

This sets the length of sending PDO 3 (see parameter byte).
 Length PDO 4 tx (Please see chapter 5.1.1 Composition of the parameter byte)
This sets the length of sending PDO 4 (see parameter byte).
 Length PDO 5 tx (Please see chapter 5.1.1 Composition of the parameter byte)
This sets the length of sending PDO 5 (see parameter byte).

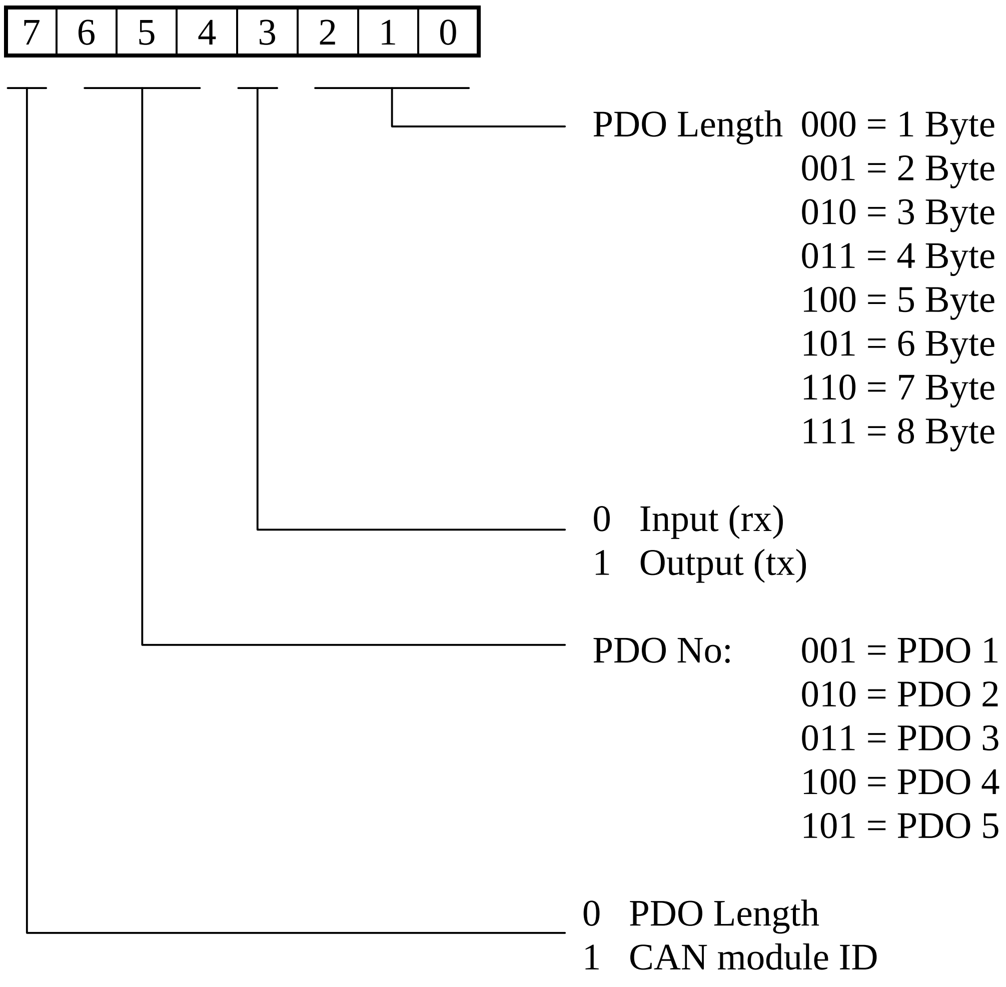
Only the PDO lengths supported by the CAN module are to be entered here.
.
.
Parameters CAN module n (Please see chapter 5.1.1 Composition of the parameter byte)
 Module ID (Please see chapter 5.1.1 Composition of the parameter byte)
The module ID of the CAN node is entered here. (range: 1..127; see parameter byte).
 Length PDO 1 rx (Please see chapter 5.1.1 Composition of the parameter byte)
This sets the length of receiving PDO 1 (see parameter byte).
 Length PDO 2 rx (Please see chapter 5.1.1 Composition of the parameter byte)
This sets the length of receiving PDO 2 (see parameter byte).
 Length PDO 3 rx (Please see chapter 5.1.1 Composition of the parameter byte)
This sets the length of receiving PDO 3 (see parameter byte).
 Length PDO 4 rx (Please see chapter 5.1.1 Composition of the parameter byte)
This sets the length of receiving PDO 4 (see parameter byte).
 Length PDO 5 rx (Please see chapter 5.1.1 Composition of the parameter byte)
This sets the length of receiving PDO 5 (see parameter byte).
 Length PDO 1 tx (Please see chapter 5.1.1 Composition of the parameter byte)
This sets the length of sending PDO 1 (see parameter byte).
 Length PDO 2 tx (Please see chapter 5.1.1 Composition of the parameter byte)
This sets the length of sending PDO 2 (see parameter byte).
 Length PDO 3 tx (Please see chapter 5.1.1 Composition of the parameter byte)
This sets the length of sending PDO 3 (see parameter byte).
 Length PDO 4 tx (Please see chapter 5.1.1 Composition of the parameter byte)
This sets the length of sending PDO 4 (see parameter byte).
 Length PDO 5 tx (Please see chapter 5.1.1 Composition of the parameter byte)
This sets the length of sending PDO 5 (see parameter byte).

Only the PDO lengths supported by the CAN module are to be entered here.

5.1.1 Composition of the parameter byte

In order to allow efficient and memory saving parameterisation, the lengths of the PDOs supported are defined by the following convention:
Bit 7 marks CAN module ID or PDO length, the next 3 bits mark the PDO number, bit 3 marks input or output PDO respectively, and the 3 lower bits mark the length of the PDO.

Parameter byte:



e.g.:	Module ID	1	>	1 000 0 001 = 81 (hex)
	Module ID	32	>	1 010 0 000 = A0 (hex)
	PDO 1 rx, Length 3Byte		>	0 001 0 010 = 12 (hex)
	PDO 4 tx, Length 4Byte		>	0 100 1 011 = 4B(hex)

5.2 Configuration

After parameterisation, the master has to send a configuration message to the respective slave. The slave receives information about the length of input and output data.

The user composes the configuration message in the project tool, where it is also possible to enter the address range in which effective data are stored (see example)

If the slave detects during the check up that input and output lengths do not correspond to the parameterisation, it will report a configuration error to the master during a later diagnostic query.
In this case, it will not be ready for transfer of effective data.

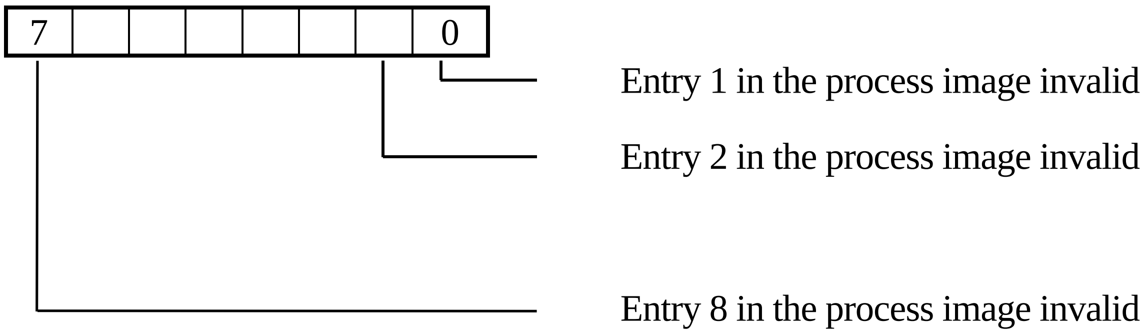
5.3 Data exchange

After the master has detected error free parameterisation and configuration at the end of the start sequence, it will send data exchange messages. For this purpose, the PROFIBUS master cyclically sends all data of parameterised sending identifiers to the PDP 2 CAN. Every change in data (both on the CANopen and the PROFIBUS side) is transferred instantly during the next cycle. In addition, the input data of CANopen modules is updated per guarding cycle (default 1 s).

5.4 Diagnostic area

The diagnostic area consists of 8 bytes. It informs the user about any invalid area in the process image and about any CAN module which is not operational.

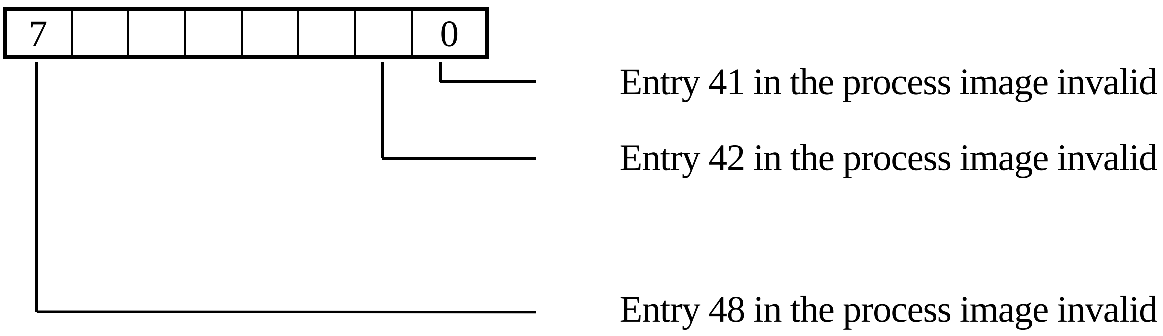
Byte 0


7
0
Entry 1 in the process image invalid
Entry 2 in the process image invalid
Entry 8 in the process image invalid

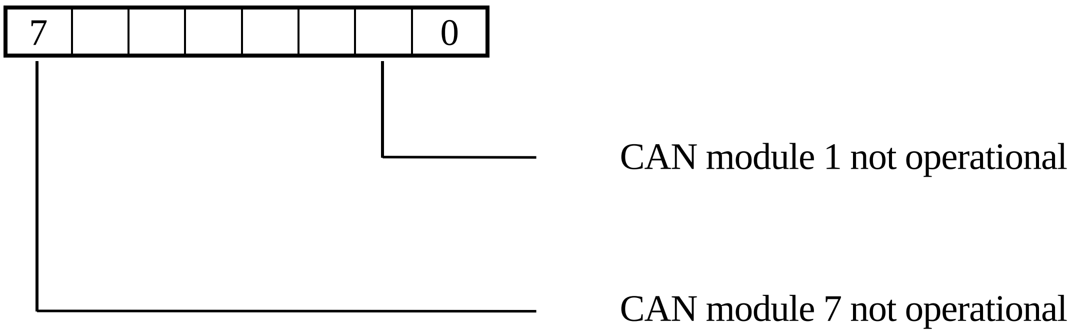
:

:

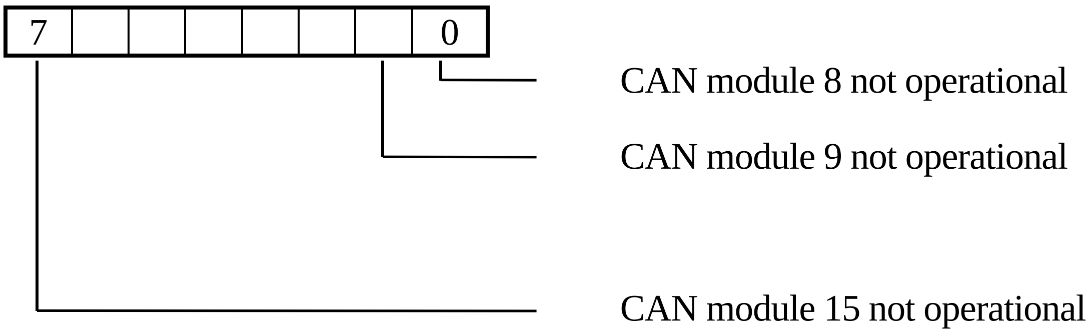
Byte 5


7
0
Entry 41 in the process image invalid
Entry 42 in the process image invalid
Entry 48 in the process image invalid

Byte 6



Byte 7


7
0
CAN module 8 not operational
CAN module 9 not operational
CAN module 15 not operational

5.5 Emergency area

Emergency messages coming from the CAN bus are copied into the emergency area. The first emergency message sent over the CAN bus is copied directly into the emergency area, and all others are stored in a ring buffer on the PDP 2 CAN. They are written into the emergency area as soon as the user has read the previous emergency message. Because of this, a handshake has been programmed. The emergency message is composed in the following way:

EMERGENCY:
Message length (total): 11
Byte 0: CANopen module ID (1 to 15)
Byte 1: 0 to 8 (data length)
Byte 2 to 9: (data bytes)

Handshaking for EMERGENCY messages:

The handshaking bits are located in the high byte of byte 0 in the SDO message.

If set, bit 4 (in Byte 0 of receiving SDO) indicates a new emergency message in the emergency area.
If set, bit 5 (in Byte 0 of receiving SDO) indicates that data in the emergency area are valid.
if set, Bit 6 (in Byte 0 of receiving SDO) indicates that the data has been read by the user.

Sequence:

PDP2CAN:	If Bit 4 and Bit 6 = 0 and EMERGENCY messages in Ring buffer, write first EMERGENCY message into emergency area, set bit 4 and bit 5 wait until bit 6 is set
	if Bit 6 = 1, delete emergency area, delete Bit 4 and Bit 5
User:	wait for bit 4 and bit 5, read EMERGENCY message, set bit 6
	Wait until bits 4 and bit 5 are deleted, delete bit 6

5.6 Parameterisation of CAN modules (SDO Transfer)

The SDO communication (SDO = **S**ervice **D**ata **O**bject) is a confirmed or acknowledged service of the CANopen protocol. This means that during error free operation, the addressed module sends a SDO response to a SDO request. SDOs are normally required for parameter configuration (write request PDO) or parameter query (read request SDO) on a module (CANopen slave). They are not suitable for fast process data transfer, which is handled by PDOs.
The PDP 2 CAN handles all SDO communication over a 11 byte window in the process image.

The PDP-2-CAN is capable to send both Expedited and Segmented transfers. An expedited transfer is always four (4) bytes long while the segmented transfer may have any length.

WARNING

It is absolutely necessary to wait for the answer of the addressed module. This means that only one SDO transfer at a time can be processed!

The SDO messages are composed in the following way:
5.6.1 Expedited-Transfers **(until 4 Bytes data)**

CAL_WRITE_REQ (send SDO variable Write):

Message length (total): 11
Byte 0: Handshake Byte
Byte 1: 36 (message type)
Byte 2: CANopen module ID (1 to 15)
Byte 3: Index (High Byte)
Byte 4: Index (Low Byte)
Byte 5: Sub index
Byte 6: 1 to 4 (number of valid data bytes)
Byte 7 to 10: (data bytes)

CAL_READ_REQ (send SDO variable Read):

Message length (total): 6
Byte 0: Handshake Byte
Byte 1: 39 (message type)
Byte 2: CANopen module ID (1 to 15)
Byte 3: Index (High Byte)
Byte 4: Index (Low Byte)
Byte 5: Sub index

CAL_READ_CNF_P (receive SDO Var Read Resp pos.):

Message length (total): 11
Byte 0: Handshake Byte
Byte 1: 40 (message type)
Byte 2: CANopen module ID (1 to 127)
Byte 3: 4 to 7 (Data length)
Byte 4: Index (High Byte)
Byte 5: Index (Low Byte)
Byte 6: Sub index
Byte 7 to 10: (Data bytes)

CAL_READ_CNF_N (receive SDO Var Read Resp neg.):

Message length (total): 11
Byte 0: Handshake Byte
Byte 1: 40 (message type)
Byte 2: CANopen module ID (1 to 127)
Byte 3: 4 to 7 (Data length)
Byte 4: Index (High Byte)
Byte 5: Index (Low Byte)
Byte 6: Sub index
Byte 7: Error_Class
Byte 8: Error_Code
Byte 9: Additional_Code (High Byte)
Byte 10: Additional_Code (Low Byte)

CAL_WRITE_CNF_P (receive SDO Var Write Resp pos.):

Message length (total): 7
Byte 0: Handshake Byte
Byte 1: 37 (message type)
Byte 2: CANopen module ID (1 to 127)
Byte 3: 3 (data length)
Byte 4: Index (High Byte)
Byte 5: Index (Low Byte)
Byte 6: Sub index

CAL_WRITE_CNF_N (receive SDO Var Write Resp neg.):

Message length (total): 11
Byte 0: Handshake Byte
Byte 1: 38 (message type)
Byte 2: CANopen module ID (1 to 15)
Byte 3: 7 (data length)
Byte 4: Index (High Byte)
Byte 5: Index (Low Byte)
Byte 6: Sub index
Byte 7: Error_Class
Byte 8: Error_Code
Byte 9: Additional_Code (High Byte)
Byte 10: Additional_Code (Low Byte)

5.6.2 Segmented-Transfers **(> 4 Bytes data)**

CAL_SEGMENTED_REQ (send Segmented SDO-Transfer)**:**

Message length (total): 11
Byte 0: Handshake-Byte
Byte 1: 60 (Telegrammtyp)
Byte 2: CANopen Modul-ID (1 until 15)
Byte 3: SDO-Telegramm-Specifier (CCS,t,X,n,e,s,c) (Byte 0 CANopen SDO-telegramm)
Byte 4: (Byte 1 CANopen SDO-telegramms) z. B. Index (Low Byte)
Byte 5: (Byte 2 CANopen SDO-telegramms) z. B. Index (High Byte)
Byte 6: (Byte 3 CANopen SDO-telegramms) z. B. Subindex
Byte 7 bis 10: (Bytes 4-7 CANopen SDO-telegramms) z. B. Datenbytes

CAL_SEGMENTED_RSP (receive Segmented SDO-Transfer)**:**

Message length (total): 11
Byte 0: Handshake-Byte
Byte 1: 61 (Telegrammtyp)
Byte 2: CANopen Modul-ID (1 until 15)
Byte 3: SDO-Telegramm-Specifier (SCS,t,X,n,e,s,c) (Byte 0 CANopen SDO-Telegramms)
Byte 4: (Byte 1 CANopen SDO-Telegramms) z. B. Index (Low Byte)
Byte 5: (Byte 2 CANopen SDO-Telegramms) z. B. Index (High Byte)
Byte 6: (Byte 3 CANopen SDO-Telegramms) z. B. Subindex
Byte 7 bis 10: (Bytes 4-7 des CANopen SDO-Telegramms) z. B. Datenbytes

SDO Timeout:

Message length (total): 11
Byte 0: Handshake Byte
Byte 1: 240 (message type)
Byte 2: CANopen module ID (1 to 15)
Byte 3 to 10: 0

This telegram appears in the receive area (SDOrx) if there is no answer received from the slave after a time period of 200 ms.

Handshaking (Send SDO)

If set, Bit 0 indicates data is valid, send SDO message
If deleted, Bit 0 indicates data not valid, wait for next SDO message. This bit must be set to 0 between two SDO messages.

Handshaking (receiving SDO)

Bit 0 set, SDO Transfer is active
If set, Bit 1 set indicates valid data in reception buffer (Data Valid)
If set, Bit 2 indicates that the SDO message has been sent (Data Send)

5.7 Startup process CAN Network (CANopen Master)

Detection and startup of slave modules connected to the CAN network are carried out as follows:

1. Reading the Guarding COB ID (Object 0x100E, 0x00) of all modules (1 to 127)
2. Reading Emergency COB ID (Object 0x1014, 0x00), active modules only
3. Reading PDO1M2S COB ID (Object 0x1400, 0x01), active modules only
4. Reading PDO2M2S COB ID (Object 0x1401, 0x01), active modules only
5. Reading PDO1S2M COB ID (Object 0x1800, 0x01), active modules only
6. Reading PDO2S2M COB ID (Object 0x1801, 0x01), active modules only
7. Reading PDO1M2S transmission type (Object 0x1400, 0x02), only if PDO valid
8. Reading PDO2M2S transmission type (Object 0x1401, 0x02), only if PDO valid
9. Writing guard time (Object 0x100C, 0x00) to active modules (Default: 1 s)
10. Writing lifetime factors (Object 0x100D, 0x00) to active modules (Default: 2)
11. Wait 100 ms
12. NMT Broadcast Start Node
13. Module guarding starts
14. In every guarding cycle, the current state of input channels (if available) is queried, and output channels (if available) are updated.
15. If an active module fails to respond to a guarding request, a new startup attempt will be made, and processes 1 to 12 will begin again as soon as the module has been identified as a bus device.

5.8 Example

In order to save time for troubleshooting and for putting the installation into operation, it is recommended to draw up a table of I/O modules the ring, their CAN identifiers and byte lengths. CAN identifiers are usually set by means of DIP switches on the node modules.

Example A

Three CAN bus nodes are configured as shown below:

1. Node:

1. Digital input module with one 8 bit input channel
2. Digital output modules with one 8 bit input channel
3. Node address is set to 11

2. Node:

1. Digital input module with one 8 bit input channel
2. Digital output modules with one 8 bit input channel
3. Node address is set to 32

3. Node:

1. Digital input module with 8 x 8 bit input channel
2. Digital output module with 4 64 bit output channels
3. Node address is set to 2

This results in the following allocation table:

	Mod-ule ID	PDO Length in Byte	User Parameters (hex)	Config data GSD file
Always 00			00	
CAN Baud rate (500kBaud)			01	
Synchronous mode (off)			00	
Number of CAN modules			03	

EA-area for SDO				SDOrx
EA-area for SDO				SDOtx
EA-area for Diagn.				DIAG
EA-area for Teleg.				EMGY

Module ID (1 xxx x xxx)	11		8B	
PDO 1 rx (0 001 0 xxx)		1 (000)	10	PDOrx 1 Byte
PDO 2 rx (0 010 0 xxx)				
PDO 3 rx (0 011 0 xxx)				
PDO 4 rx (0 100 0 xxx)				
PDO 5 rx (0 101 0 xxx)				
PDO 1 tx (0 001 1 xxx)		1 (000)	18	PDOtx 1 Byte
PDO 2 tx (0 010 1 xxx)				
PDO 3 tx (0 011 1 xxx)				
PDO 4 tx (0 100 1 xxx)				
PDO 5 tx (0 101 1 xxx)				

Module ID (1 xxx x xxx)	20		A0	
PDO 1 rx (0 001 0 xxx)		1 (111)	17	PDOrx 8 Byte
PDO 2 rx (0 010 0 xxx)				
PDO 3 rx (0 011 0 xxx)				
PDO 4 rx (0 100 0 xxx)				
PDO 5 rx (0 101 0 xxx)				
PDO 1 tx (0 001 1 xxx)		1 (111)	1F	PDOtx 8 Byte

PDO 2 tx (0 010 1 xxx)				
PDO 3 tx (0 011 1 xxx)				
PDO 4 tx (0 100 1 xxx)				
PDO 5 tx (0 101 1 xxx)				

Module ID (1 xxx x xxx)	2		82	
PDO 1 rx (0 001 0 xxx)				
PDO 2 rx (0 010 0 xxx)				
PDO 3 rx (0 011 0 xxx)				
PDO 4 rx (0 100 0 xxx)				
PDO 5 rx (0 101 0 xxx)				
PDO 1 tx (0 001 1 xxx)		1 (111)	1F	PDOtx 8 Byte
PDO 2 tx (0 010 1 xxx)		1 (111)	2F	PDOtx 8 Byte
PDO 3 tx (0 011 1 xxx)		1 (111)	3F	PDOtx 8 Byte
PDO 4 tx (0 100 1 xxx)		1 (111)	4F	PDOtx 8 Byte
PDO 5 tx (0 101 1 xxx)				

This environment requires the following parameterisation and configuration data (in hex, separated by comma)

Hex-Parameter (PDP2CAN-specific):
00,01,00,03,8B,10,18,A0,17,1F,82,17,1F,2F,3F,4F,00,00,00,00,00....

Configuration data:

SDOrx
SDOtx
DIAG
EMGY
PDOrx 1 Byte
PDOtx 1 Byte
PDOrx 8 Byte
PDOtx 8 Byte
PDOrx 8 Byte
PDOtx 8 Byte
PDOtx 8 Byte
PDOtx 8 Byte
PDOtx 8 Byte

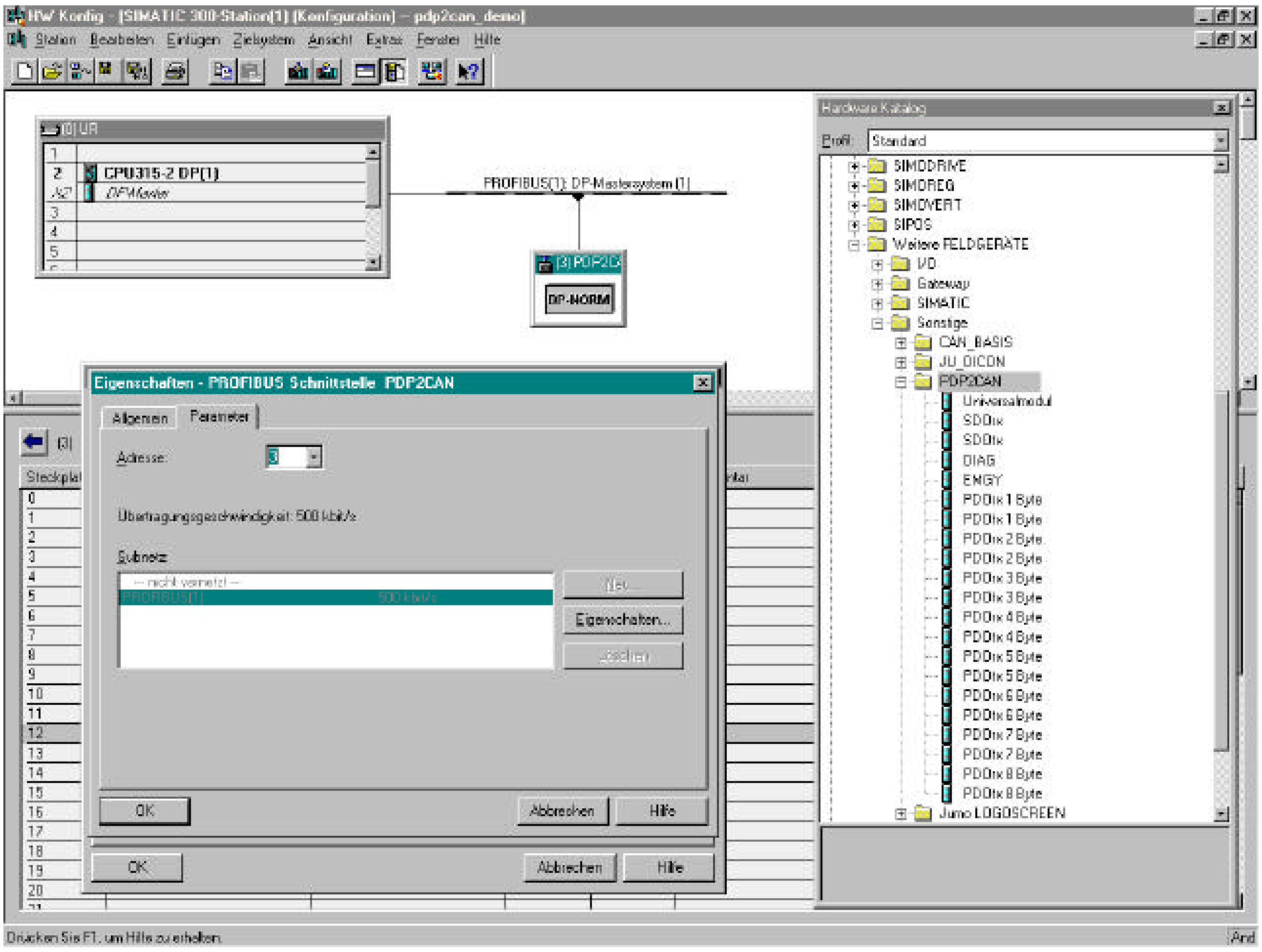
With these configuration data, the objects SDOrx, SDOtx, DIAG, EMGY have to be configured in the sequence shown above. If they are not, PDP-2-CAN will report a parameterisation error.

Chapter 6

6 Projects with STEP 7

Perform the following steps:

Copy GSD file PDP2CAN.GSD into path ..\S7DATA\GSD.
Use command update GSD files in order to update the hardware catalogue.
Activate slave PDP2CAN from the path “other field bus devices”, “others”.
All possible configuration Ids will be shown similar to figure 4: slave modules
The PDP2CAN slave can be placed into the Profibus(1) net by drag and drop.
Assign the desired address to the slave.


HW Konfig - [SIMATIC 300-Station(1) (Konfiguration) -- pdp2can_demo]

Eigenschaften - PROFIBUS Schnittstelle PDP2CAN

Adresse: 3

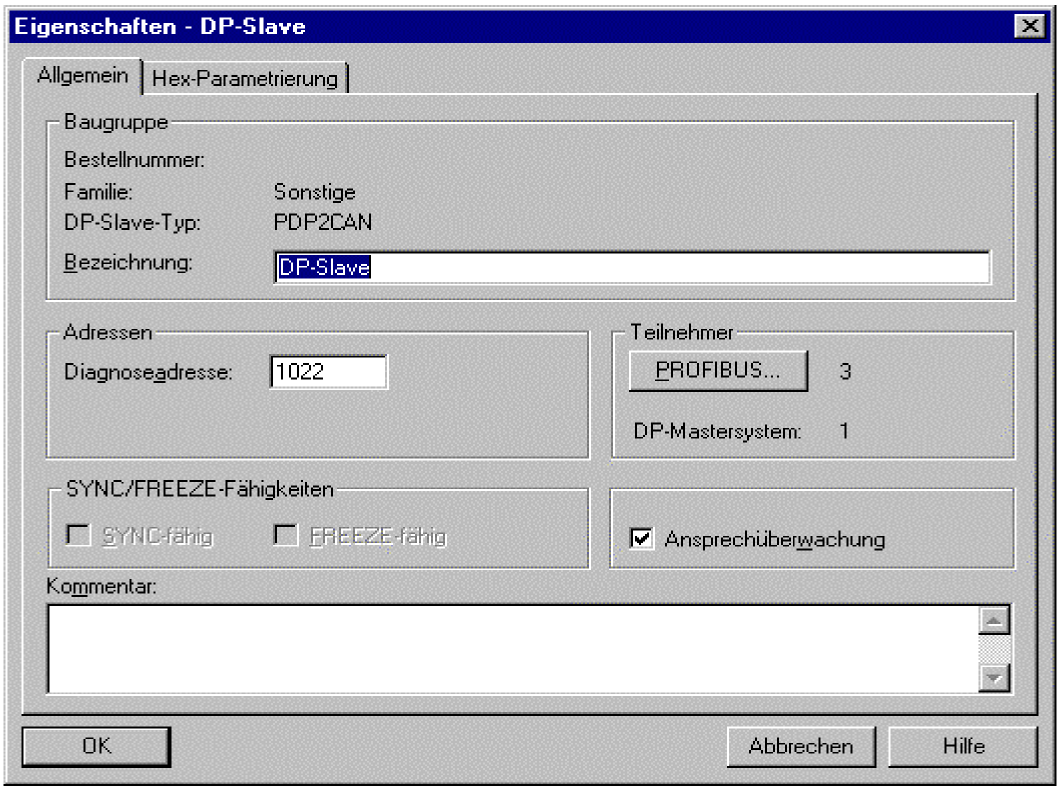
Übertragungsgeschwindigkeit: 500 kbit/s

Subnetz:
-- nicht vernetzt --
PROFIBUS(1) 500 kbit/s

Drücken Sie F1, um Hilfe zu erhalten.

Figure 4: Slave modules

When this is done, take the desired PDP2CAN module from the hardware catalogue and place it (per drag and drop) into the slave's composition list according to the configuration list.
When selecting "DP slave properties" (DP slave Eigenschaften) in the parameterisation section, general and user-specific parameterisation can be carried out (see figures 5 and 6). Figure 6 also shows a sample parameterisation.


Eigenschaften - DP-Slave

Allgemein | Hex-Parametrierung

Baugruppe

Bestellnummer:	
Familie:	Sonstige
DP-Slave-Typ:	PDP2CAN
Bezeichnung:	DP-Slave

Adressen

Diagnoseadresse: 1022

Teilnehmer

PROFIBUS... 3

DP-Mastersystem: 1

SYNC/FREEZE-Fähigkeiten

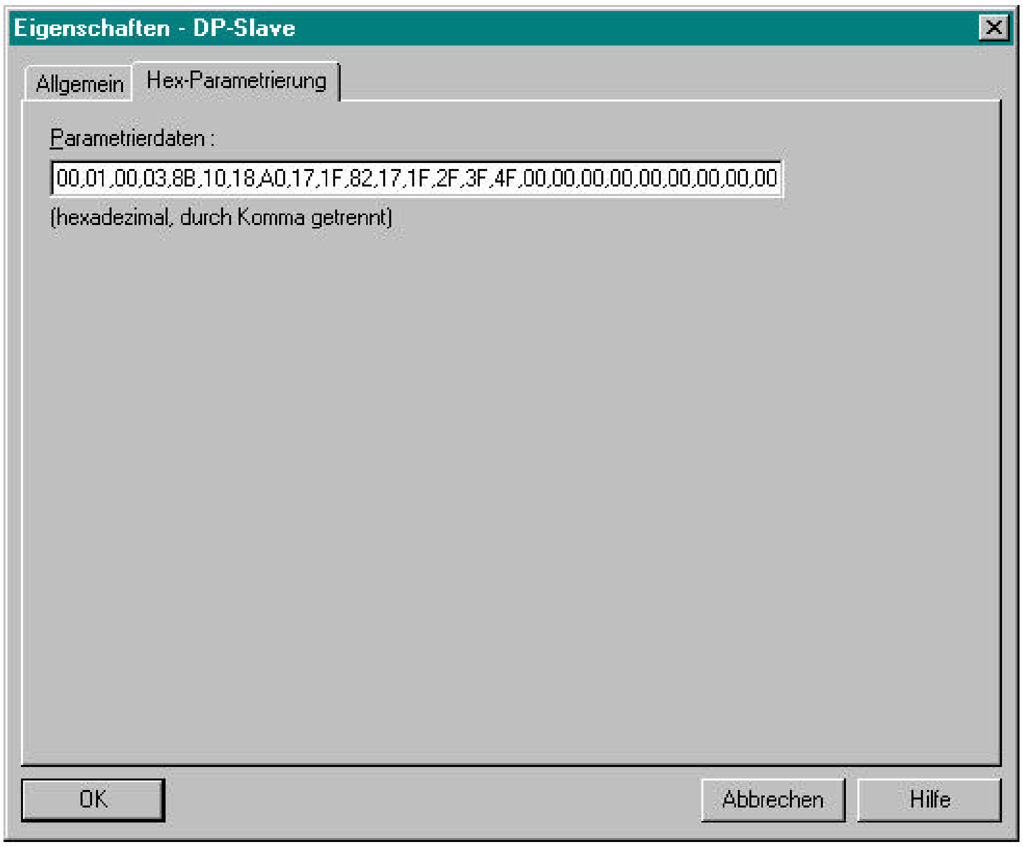
☐ SYNC-fähig ☐ FREEZE-fähig

☑ Ansprechüberwachung

Kommentar:

OK | Abbrechen | Hilfe

Figure 5: general properties DP Slave

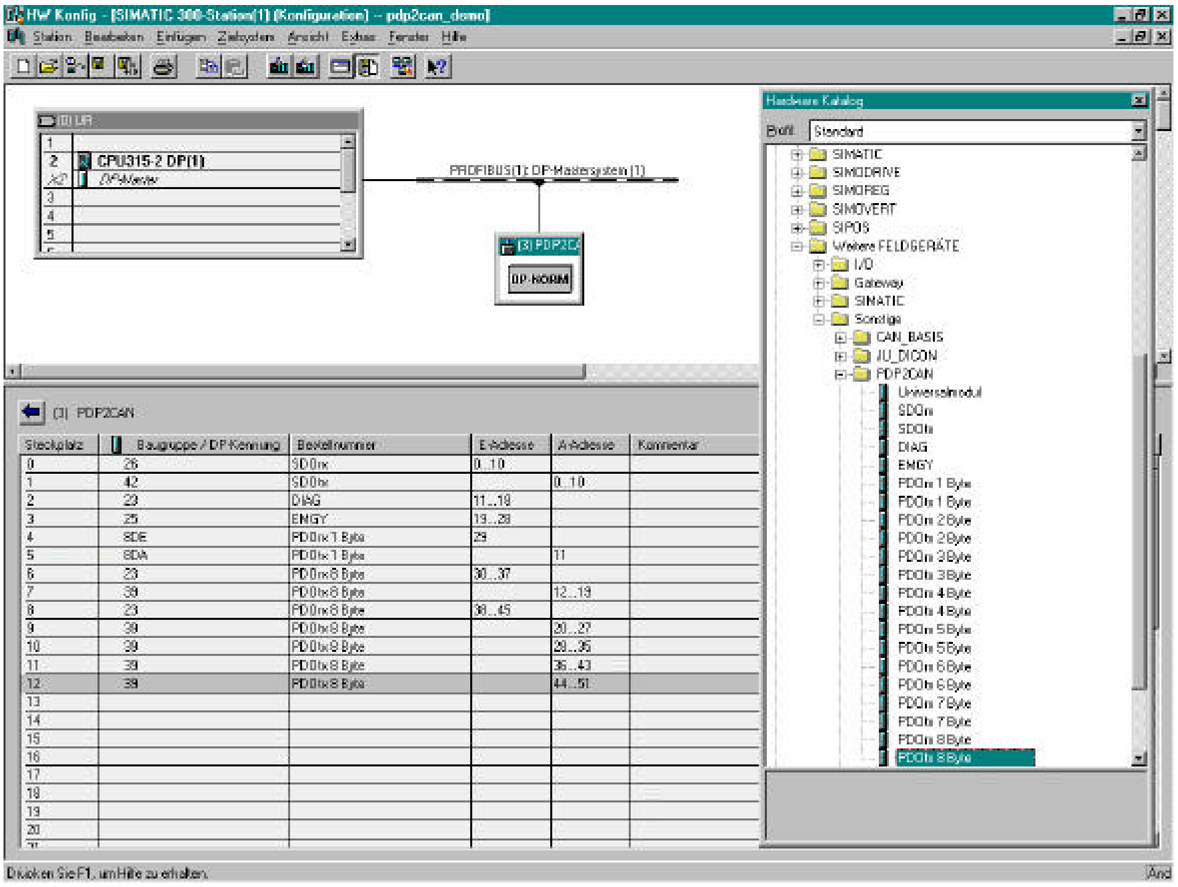


Parametrierdaten :

00,01,00,03,8B,10,18,A0,17,1F,82,17,1F,2F,3F,4F,00,00,00,00,00,00,00,00,00

(hexadezimal, durch Komma getrennt)

Figure 6: user specific parameterisation DP Slave



(3) PDP2CAN

Steckplatz	Baugruppe / DP-Kennung	Bestellnummer	E-Adresse	A-Adresse	Kommentar
0	26	SDOrx	0...10		
1	42	SDOtx		0...10	
2	23	DIAG	11...18		
3	25	ENGY	19...28		
4	8DE	PDOrx 1 Byte	29		
5	8DA	PDOtx 1 Byte		11	
6	23	PDOrx 8 Byte	30...37		
7	39	PDOtx 8 Byte		12...19	
8	23	PDOrx 8 Byte	38...45		
9	39	PDOtx 8 Byte		20...27	
10	39	PDOtx 8 Byte		28...35	
11	39	PDOtx 8 Byte		36...43	
12	39	PDOtx 8 Byte		44...51	
13					
14					
15					
16					
17					
18					
19					
20					

Figure 7: process image DP Slave

Chapter 7

7 CANopen

7.1 General

The CAN bus (control area network) is an international open field bus standard for buildings, automation of manufacturing and processes. Originally, it was developed for car technology. Because of its comprehensive error recognition, the CAN bus is considered the safest bus system with a remaining error probability of less than 4.7 x 10^{11}. Faulty messages are signalled and sent again automatically.
In contrast to Profibus and Interbus S, different layer 7 user profiles are defined according to CAL layer 7 protocol. (CAL=CAN application layer). One is these user profiles is CANopen. The CiA (CAN in Automation) is in charge of its standardisation.

7.1.1 CANopen

CANopen is the user profile in the area of industrial real-time systems. It is currently being implemented by many manufacturers. CANopen was published as profile DS 301 by the CAN user organisation CiA. The communication profile DS 301 was designed for standardisation of devices.
Products from different manufacturers become thus interchangeable.
In addition to that, device-specific data and process data are standardised in the device profile DS 401 in order to ensure that devices are interchangeable. DS 401 is standardises digital and analogue input/output modules. The CANopen communication profile is based on an object directory similar to the one used by Profibus. The communication profile DS 301 defines two types of objects as well as some special objects:

- Process data objects (PDO)
  PDOs are used for transferring real-time data
- Service data objects (SDO)
  SDOs allow read and write access to the object directory

It consists of a communication profile specifying the objects which are to be used for transfer of specific data, and the device profiles which define the data to be transferred together with the objects.

7.1.2 Transfer medium

CAN is based on a bus topology. You have the option of building up a network structure by means of router nodes. The number of participants in the net is only limited by the performance of the bus driver module used.
The maximum network extension is limited speed of signals. At 1 Mbaud, the maximum total length is 40 meters, and at 80 Kbauds, 1000 meters are possible. CAN bus uses a shielded three-wire cable as transfer medium (five-wire cable optional).
The CAN bus works with differences in voltage. Thus, it is less susceptible against interference than a voltage or electric current interface. The net should be configured as a bus with a 120 Ω terminator resistor at the end.
All devices on the net communicate at the same baud rate. The bus topology enables you to integrate and remove components without consequences and allows you to put the installation into operation in a step-by-step process. Later extensions have no effect on devices which are already operational. The system detects automatically if a device malfunctions or if new devices are present on the net.

7.1.3 Bus access

In general, one can differentiate between controlled (deterministic) or uncontrolled (random) bus access.
CAN works according the system Carrier Sense Multiple Access (CSMA), which means that every device has the same privileges concerning bus access and is able to access the bus as soon as it is free (random bus access)
The exchange of messages is related to the messages, not the devices. Every message is unambiguously marked with a priority identifier. Only one device at a time can use the bus for its message. During simultaneous multiple access
The bus access control within CAN is done by means of non-destructive, bit-wise arbitration.
Non-destructive means that the winner determined by the arbitration process does not have to re-send its message. In a situation where multiple a simultaneous access takes place, the device with the highest priority is selected. When a device which is ready to send detects that the bus is busy, it will delay its request to send until the current transfer has finished.

7.1.4 Can Baud rate

CAN Baud rate	gar. max. bus length
1 Mbaud	25 m
500 kBaud	100 m
250 kBaud	250 m
125 kBaud	500 m
100 kBaud	600 m
50 kBaud	1000 m
20 kBaud	2500 m
10 kBaud	5000 m

7.1.5 Cables for the CAN Bus

The CAN Bus uses a shielded three-wire cable.

Figure 7 1: CAN Bus: cables

7.1.6 Line terminator

In systems with more than two devices, all devices are wired parallel. The bus cable must be run through the device without interruption

A terminator resistor must be used at both cable ends in order to avoid signal reflection and transfer problems!

7.1.7 Message structure:

All CANopen messages use the following structure:

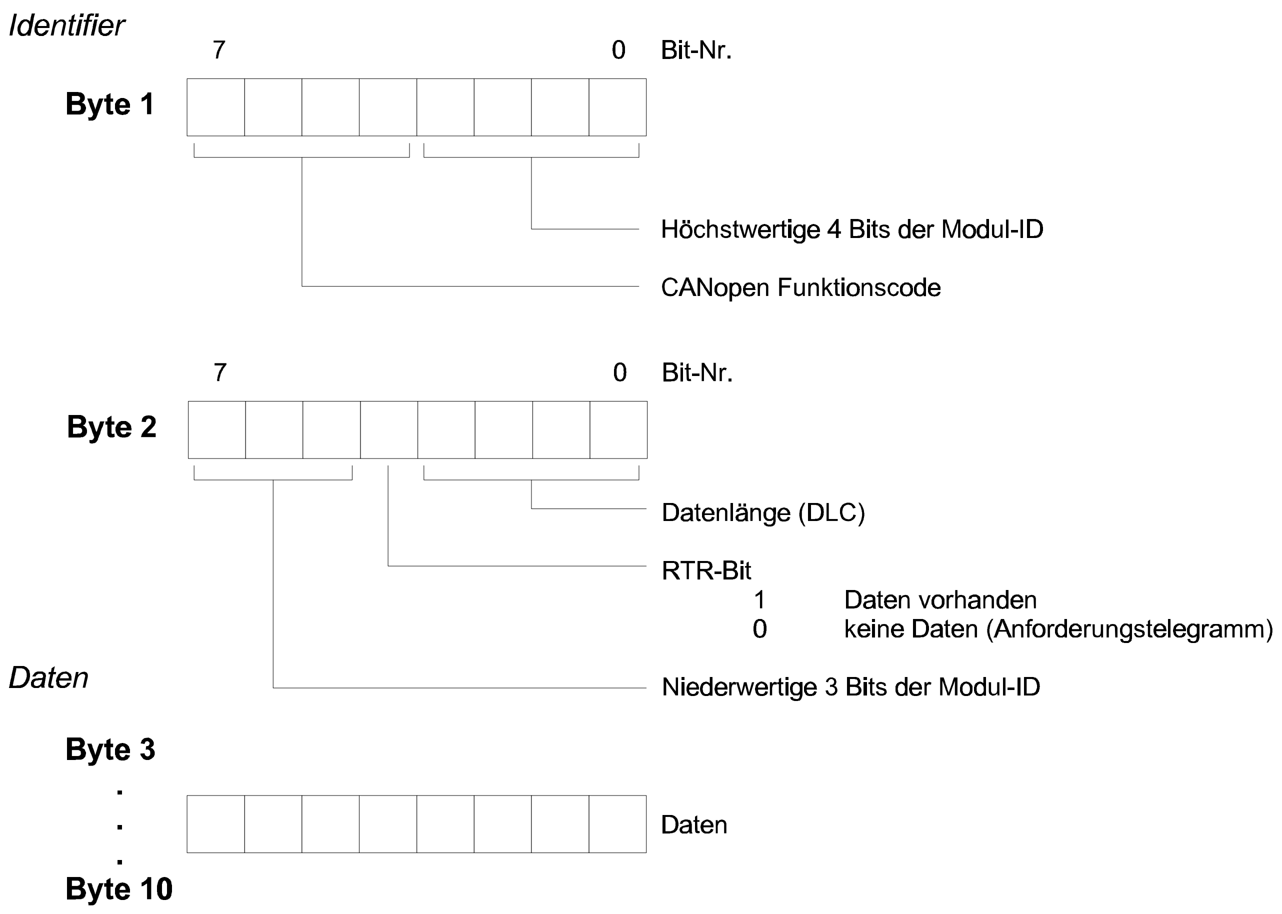


Fig. 7 2: CANopen: message structure

In contrast to a layer 2 message, there is an additional separation of the 2 Byte identifier into a function part and a module ID. The function part specifies the type of the message (object), addresses the recipient by means of the module ID. The data exchange between CANopen devices is carried out by means of objects. The CANopen communication profile defines two object types and several special objects. Every object is assigned a function code (please see following table)

7.1.7.1 CANOPEN FUNKTIONSCODES

This table lists the objects defined under CANopen with their function codes.

Object	**Function Code (4 Bits)**	**Recipient**	**Definition**	**Function**
NMT	0000	Broadcast	CiA DS 301	Network management
EMERGENCY	0001	Master	CiA DS 301	Error message
PDO1S2M	0011	Master, Slave (RTR)	CiA DS 301	Digital input data 1
PDO1M2S	0100	Slave	CiA DS 301	Digital output data 1

PDO2S2M	0101	Master, Slave (RTR)	CiA DS 301	Analogue input data 1
PDO2M2S	0110	Slave	CiA DS 301	Analogue output data 1
PDO3S2M	0111	Master, Slave (RTR)	Application spec.	D o. A input data 2
PDO3M2S	1000	Slave	Application spec.	D o. A output data 2
PDO4S2M	1001	Master, Slave (RTR)	Application spec.	D o. A input data 3
PDO4M2S	1010	Slave	Application spec.	D o. A output data 3
PDO5S2M	1101	Master, Slave (RTR)	Application spec.	D o. A input data 4
PDO5M2S	1111	Slave	Application spec.	D o. A output data 4
SDO1S2M	1011	Master	CiA DS 301	Configuration data
SDO1M2S	1100	Slave	CiA DS 301	Configuration data
Node Guarding	1110	Master, Slave (RTR)	CiA DS 301	Module monitoring

The exact structures and contents of all objects are described in detail in "CiA Communication Profile DS 301 Version 3.0" and "CiA Device Profile for I/O Modules DPS 401 Version 1.4".

7.1.7.2 CANOPEN OBJECT S

PDO

For the exchange of process data, Process Data Objects (PDOs) are available. Every PDO consists of 8 data bytes. Transmit PDOs are used for input data, and Receive PDOs for output data. PDOs are transferred without confirmation, because the CAN protocol ensures that the data is transmitted correctly.

SDO

The Service Data Object (SDO) is used for access to the object directory. The SDO allows read or write access to the object directory. The specifications for the Multiplexed Domain Transfer Protocol used by the SDO can be found in the CAN layer 7 protocol. If necessary, messages are split into several CAN messages with identical identifiers (segmentation). In the SDO`s first CAN message, 4 or 8 bytes are used for protocol information. For access to the object directory with a maximum length of up to 4 bytes, a single CAN message is sufficient. When the data is longer than 4 bytes, a segmented transfer takes place. The following SDO objects contain up to 7 bytes effective data. The last bit carries and end marker. Transfer of SDOs is confirmed, i.e. the reception of every message is acknowledged. A detailed description of the exact structure and contents of all objects can be found in "CiA Communication Profile DS 301 Version 3.0" and in "CiA Device Profile for I/O Modules DS 401 Version 1.4". All necessary error messages for SDOs according to DS 301 have been implemented.

Emergency Object

Emergency objects are broadcast so that other devices on the CAN bus can be informed about internal device errors. After boot up, the COB identifier configured in the variable 1014h for the emergency message is 080h + module ID (hex notation).

CANopen EMERGENCY message contents:

Byte No.	Contents
0	Emergency Error Code (DS 301) low Byte
1	Emergency Error Code (DS 301) high Byte
2	Emergency Error Register (DS 301)
3	Application specific Error Code
4	Additional Error Information 1
5	Additional Error Information 2
6	Additional Error Information 3
7	Additional Error Information 4

Node Guarding

CAN open defines Node Guarding in order to allow monitoring bus devices. The guarding operation of the module starts with the first guarding request message (RTR) received by the master. The respective COB identifier is firmly set to 700h + module ID in the object directory. If, during guarding operation, no guarding request message (RTR) is received within the parameter specified in “Guard Time” (Object 100Ch), the module will assume that the master is not working correctly any more. After the time set by the product of “Guard Time” (100Ch) and “Life Time Factor” (100Dh), the module automatically switches to “Pre-Operational” mode.

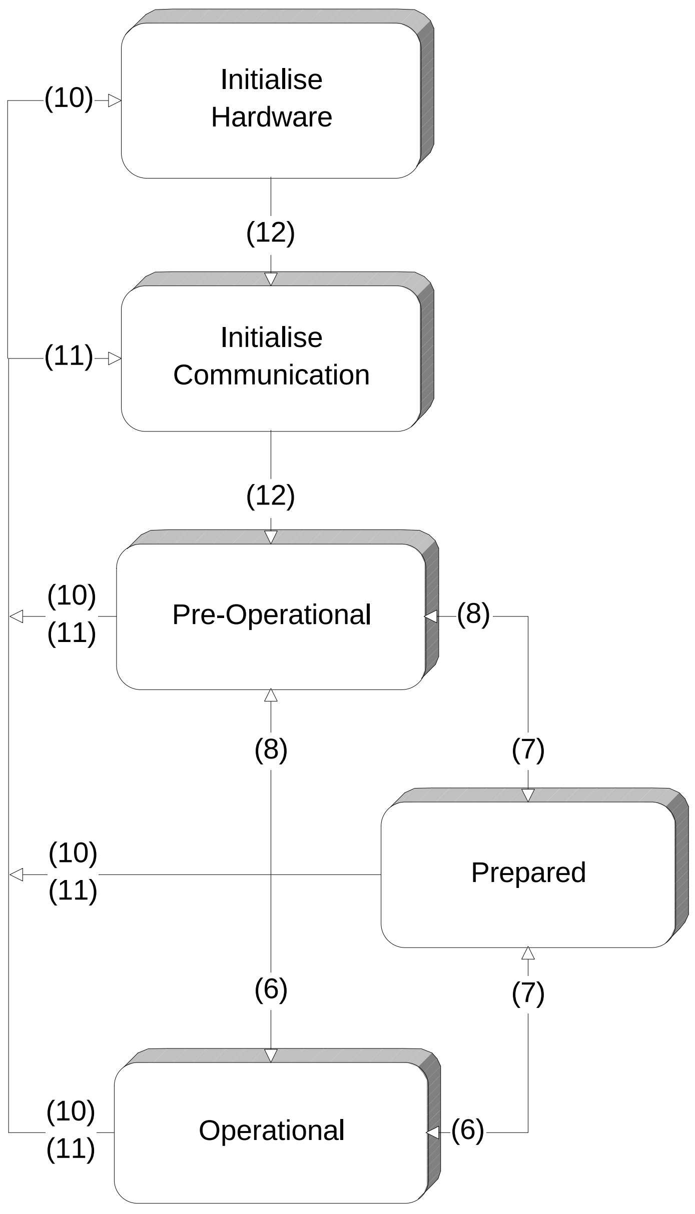
If either “Guard Time” (100Ch) or “Life Time Factor” (100Dh) is set to zero by means of SDO download from the master, no check for the expiration of guarding time is carried out, and the module will remain in its current mode.

NMT

The network management (NMT) specifies global services for network monitoring and management. This includes, apart from login on and out individual devices, monitoring all devices during normal operation and handling exceptions.

NMT service messages have the COB identifier 0000h. An additional module ID is not required. Length is always 2 data bytes. The first data byte contains the NMT Command Specifier:

NMT Services (DS 301):



(6): "Start_Remote_Node"
NMT Command Specifier: 01h

(7): "Stop_Remote_Node"
NMT Command Specifier: 02h

(8): "Enter_Pre operational_State"
NMT Command Specifier: 80h

(10): "Reset_Node"
NMT Command Spe cifier: 81h

(11): "Reset_Communication"
NMT Command Specifier: 82h

(12): Initialisation finished
enter Preoperational automatically

The second data byte contains the module ID (00h for a Broadcast Command).

8 Specifications

Voltage:	24V DC
Power consumption:	ca. 150mA
Galvanic insulation:	1 kV DC
Bus speed CAN:	max. 1 MBit
Bus speed Profibus:	max. 12 MBit
Protection:	IP 20
Dimensions:	height: 80mm
	width: 23mm
	depth: 90mm
Mounting:	Hat-Rail mounting

General Warning!

In order to conform to EMV regulations, all data lines must be shielded. This shield must be connected to the earth potential. All earth clamp of our modules must be connected to the earth potential, too.
Antal Electronic will not guarantee compliance to EMV protection measures if these measures are not taken.