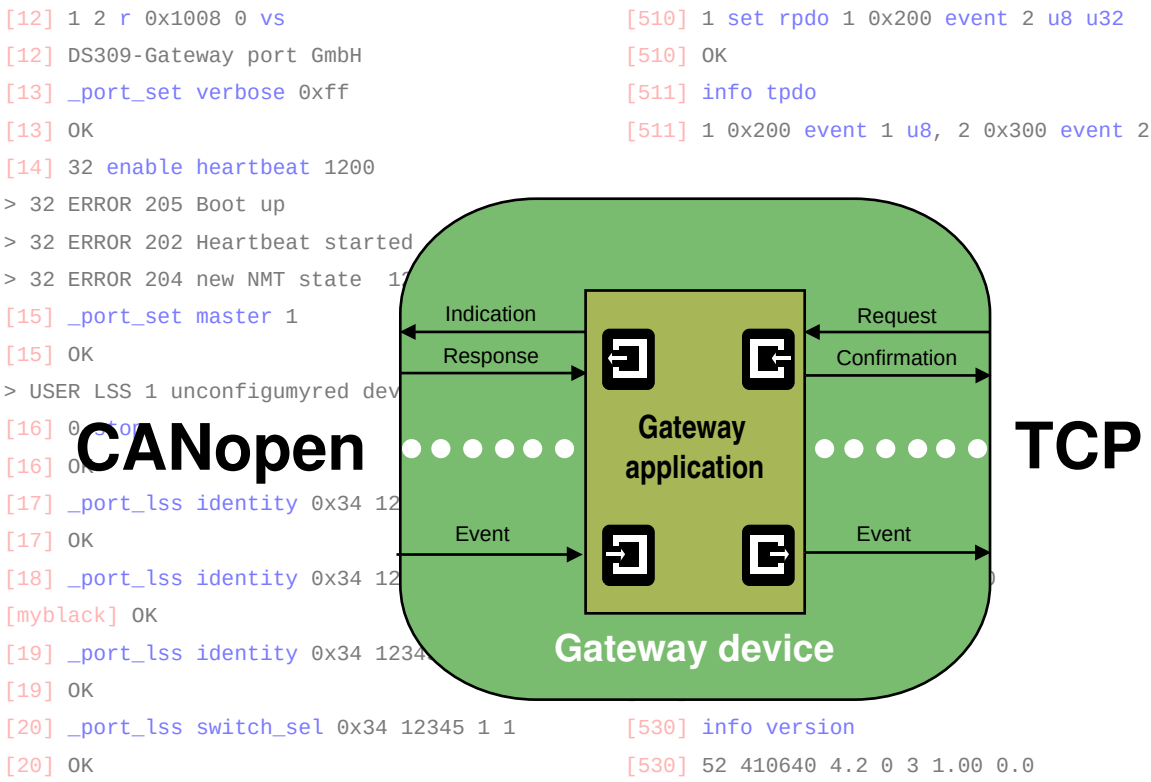
CANopen

Gateway Server DS309-3
User Manual



Disclaimer
All rights reserved

The programs, boards and documentations supplied by *port* GmbH are created with due diligence, checked carefully and tested on several applications.

Nevertheless, *port* GmbH can not take over no guarantee and no assume del credere liability that the program, the hardware board and the documentation are error-free respective are suitable to serve the special purpose.

In particular performance characteristics and technical data given in this document may not be constituted to be guaranteed product features in any legal sense.

For consequential damages, which are emerged on the strength of use the program and the hardware boards therefore, every legal responsibility or liability is excluded.

port has the right to modify the products described or their documentation at any time without prior warning, as long as these changes are made for reasons of reliability or technical improvement.

All rights of this documentation lie with *port.* The transfer of rights to third parties or duplication of this document in any form, whole or in part, is subject to written approval by *port.* Copies of this document may however be made exclusively for the use of the user and his engineers. The user is thereby responsible that third parties do not obtain access to these copies.

The soft- and hardware designations used are mostly registered and are subject to copyright.

CANopen®
is registered trademark, licensed by CiA - CAN in Automation e.V., Germany.

EtherCAT®
is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

We are thankful for hints of possible errors and may ask around for an information.

We will go all the way to verify such hints fastest

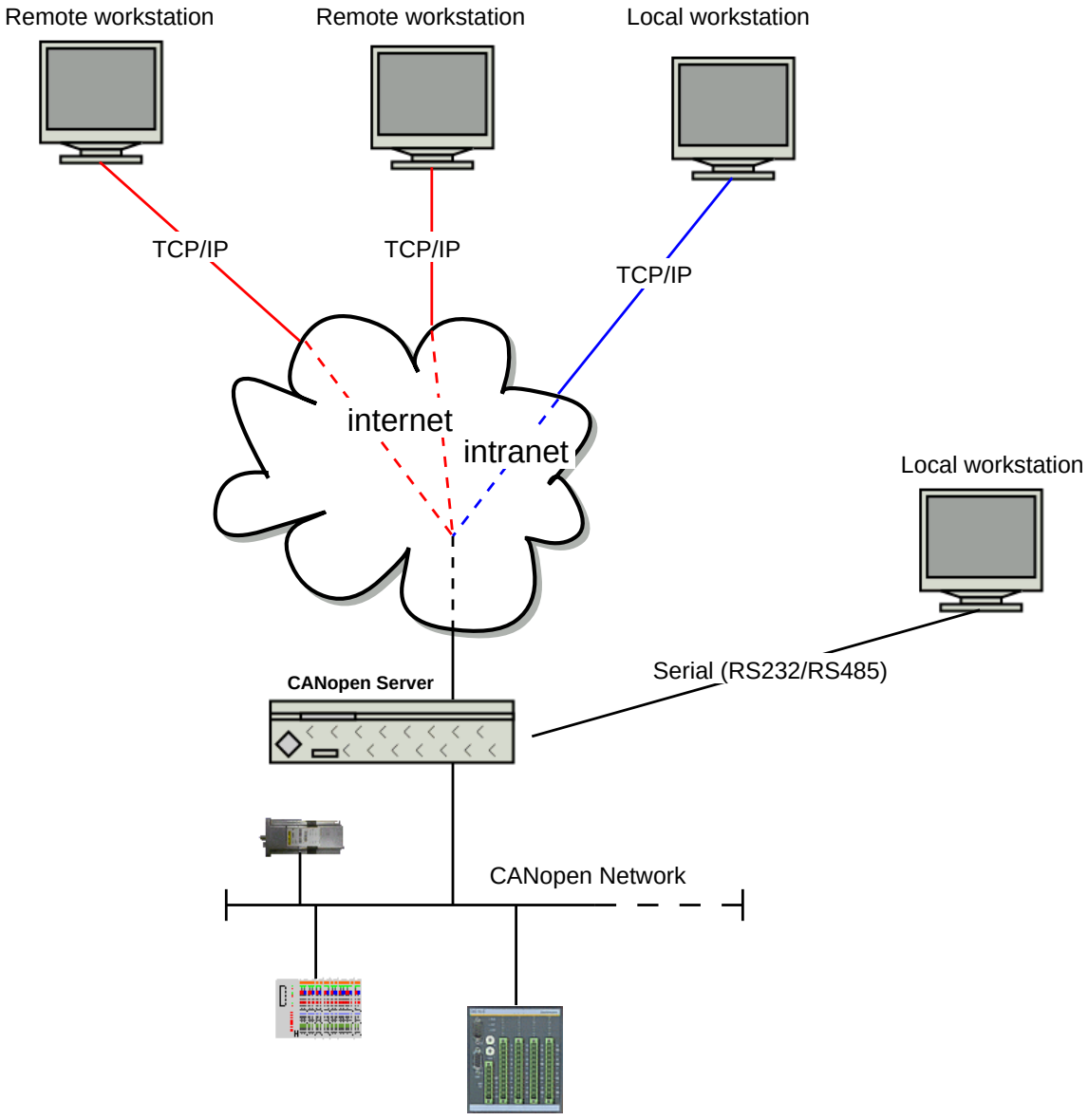
Copyright

Regensburger Straße 7
D-06132 Halle
Tel. +49 345 - 777 55 0
Fax. +49 345 - 777 55 20
E-Mail service@port.de
Internet http://www.port.de

Table of Contents

1. Overview

The CANopen Gateway Server implements the protocol DS309-3. It can be accessed from a remote workstation or locally.



1.1. Features

Service	Instances	
	Full version	Starterkit
SDO-Client	127	2(*)
SDO-Server	1	1
PDO-Producer	512	1,2
PDO-Consumer	512	1,2
EMCY-Consumer	127	127
Heartbeat-Consumer	127	2 (*)
Time-Producer	✗	-
Time-Consumer	✗	-
NMT-Master	✗	✗
LSS-Master	✗	-
Domain Size	15 MByte (**)	1 MByte (**)
TCP/Clients	10	2
Other	-	automatic exit after 1 hour

(*) Node id 32 or 64 can be accessed.

(**) EtherCAN: 128 kByte, one domain transfer at a time

See appendix 1 for the complete object directory.

Note: The performance of the CANopen Gateway Server depends on the used CAN interface hardware. Especially at high bus load and high baud rates some CAN messages may be lost.

1.2. Installation and Start

1.2.1. CAN driver

The CANopen Gateway Server accesses the CAN interface card by means of a layer 2 driver. This driver has to be installed before using the CANopen Server. For installation of the driver please refer to the delivered driver manual.

EMS Wünsche ⟨`http://www.ems-wuensche.com`⟩
I+ME Actia ⟨`http://www.ime-actia.de`⟩
Kvaser ⟨`http://www.kvaser.com`⟩
Sontheim ⟨`http://www.sontheim-industrie-elektronik.de`⟩
Janz ⟨`http://www.janztec.com`⟩

The most current list of supported hardware can be requested at service@port.de.

1.2.2. Command line options

1.2.2.1. Common options

Option	Argument	Description	Default value
-p	port	- TCP/IP port	7234
-b	baud	- CAN Baudrate (*)	125
-n	id	- Node-ID of CANopen Gateway Server	6
-H	time	- start producing HB frames at start-up	0
-R		- Don't send NMT PREOP ALL at exit	
-D	device	- CAN device/channel, e.g. can1	
-V		- Version	
-S		- Server mode	
-L	filename	- Path of licence file	

(*) values for baudrate: 20, 50, 125, 250, 500, 1000

When the CANopen Gateway Server is started without option **-S** it runs in command line mode. Commands can be entered interactively from command line, then, and sequence numbers can be omitted. In command line mode the CANopen Gateway Server processes the initialization file `m4d_ini.cmd`. This ASCII file is contains DS309-3 commands. Lines starting with "#" are comments and are not executed. It can be tailored to match the application needs.

1.2.2.2. Options for Linux, EtherCAN, IGW900

Option	Description
-s	Timer resolution in ms

1.2.2.3. Options for EMS Wünsche CPC

Option -D specifies the channel given in the configuration file C:\Windows\cpcconf.ini which is located in the Windows system directory.

1.2.2.4. Options for Peak

Option	Description
-i	Interface 1 - USB 2 - Parallel Dongle
-I	Interrupt for parallel dongle
-B	I/O base address for parallel dongle

1.2.2.5. Options for Kvaser

Option -D specifies the card channel. The card channel can be obtained with the "Kvaser Hardware Configuration" tool.

2. Definitions

Command

controls the CANopen Gateway Server and interacts with CANopen devices. It may have a long form and a short form. The short form is a one or two letter abbreviation of the long form. The long form is obtained by concatenating the short form and the string enclosed in brackets ”[”, ”]”.

Datatypes

Syntax	CANopen Type	Supported
b	Bool	✓
u8	Unsigned8	✓
u16	Unsigned16	✓
u24	Unsigned24	✓
u32	Unsigned32	✓
u24	Unsigned40	✓
u48	Unsigned48	✓
u56	Unsigned56	✓
u64	Unsigned64	✓
i8	Integer8	✓
i16	Integer16	✓
i24	Integer24	✓
i32	Integer32	✓
i24	Integer40	✓
i48	Integer48	✓
i56	Integer56	✓
i64	Integer64	✓
r32	Real32	✓
r64	Real64	✓
t	Time of day (two arguments: day ms)	✓
td	Time difference	-
vs	Visible string	✓
os	Octet string	✓
us	Unicode string	✓
d	Domain	✓

Visible strings can have a maximum length of 127 bytes. If the string contains "white-spaces" it has to be enclosed in double quotes.
Octet and unicode strings and domain data are base64 encoded. The encoded octet and unicode string can have a maximum length of 127 bytes. Domain data up to 1 mega byte is supported.

2.1. Commands

Commands are communicated as case insensitive ASCII strings. Numbers are represented in manner of the C programming language:

100	- decimal, starting with a number
0x64	- hexadecimal, starting with the string 0x
1.22	- float
.22e10	- float
22e3	- float

2.1.1. Command Request

A command is composed of tokens separated by whitespaces and closed with a CRLF.
All commands are confirmed. Each command starts with a 4 byte sequence number which is enclosed by square brackets. The sequence number is followed by net and node of the CANopen device. Both numbers are optional (see Set default network/node).
In BNF notation a command defines as follows:

```
<command-request>    ::= "["<sequence>"]" [[<net>] <node>] <command>
<sequence>           ::= UNSIGNED32
<net>                ::= UNSIGNED8
<node>               ::= UNSIGNED8
<command>            ::= <command-specifier> | <compound-command>
<compound-command>   ::= <command-specifier> <parameter>
<parameter>          ::= <value> | <command-parameter>
<compound-parameter> ::= <value> <parameter>
```

Net and node numbers are starting with 1. The value 0 is used to address all nodes and all networks.

2.1.2. Command Response

The CANopen Gateway Server responds with the same sequence number at the first position as given by the request. This number shall be given in decimal format.

```
<command-response> ::= "["<sequence>"]" <response>
<response>         ::= <value> | <error-string> | <emcy-list> | "OK"
<error-string>     ::= "Error:" <error code>
<error-code>       ::= <internal-error-code> | <sdo-abort-code>
<emcy-list>        ::= [<emcy1> " "..<emcy254>]
<emcyN>            ::= UNSIGNED32
```

Allowed internal-error-codes (IEC) are listed below:

IEC	Message text
100	request not supported
101	syntax error
102	Request not processed due to internal state

IEC	Message text
200	lost Guarding Msg
201	lost connection
202	Heartbeat started
203	Heartbeat lost
204	new NMT state
205	Boot up
300	CAN passive
301	CAN bus off
302	CAN overflow
303	CAN buffer overflow
304	CAN init
305	CAN active
400	PDO already used
401	PDO length exceeded

2.1.3. Event triggered messages

Messages due to errors in the CANopen network or the occurrence of communication objects using the producer-consumer principle (PDO, EMCY) shall not use a sequence number.

```
<event-trigged-message> ::= [[net] node] <event-specifier> <parameter>
<event-specifier>       ::= "EMCY" | "ERROR" | "SYNC" | <pdo-event> |
                            "USER" <user-event>
<pdo-event>             ::= [net] "PDO" <parameter>
<user-event>            ::= <lss-event> <block-event>
<lss-event>             ::= "LSS" <parameter>
<block-event>           ::= "BLOCK" <parameter>
```

The content of event-triggered messages is described within the command description that enables the specific service.

3. Network access command specification

3.1. SDO access commands

3.1.1. Upload SDO

Command syntax

```
[[net] node] r[ead] <index> <subindex> <datatype>
```

Accesses the remote device via SDO read messages.

Domain data have to be passed bas64 encoded. See also "SDO Block event" and Appendix: Examples.

3.1.2. Download SDO

Command syntax

```
[[net] node] w[rite] <index> <subindex> <datatype> <value>
```

Accesses the remote device via SDO write messages.

3.1.3. Configure SDO timeout

Command syntax

```
[[net] node] set sdo_timeout <ms>
```

The SDO timeout time is used to abort a SDO transfer when there is no device with the given node id. SDO timeout times are node specific. In order to set one SDO timeout for all nodes node id 0 can be used.

A timeout is measured between each SDO request and confirmation.

3.1.4. SDO Block Event

Command syntax

```
[net] [node] USER BLOCK <current> <maximum>
```

With SDO domain down/upload the server indicates progress of the current transmission with the SDO Block Event. On an SDO Upload it is required that the SDO server provides correct size information. The current argument is the number of blocks that have been transmitted/received. The maximum argument is the number of total blocks that have to be transmitted or received.

3.2. PDO access commands

3.2.1. Configure RPDO command

Command syntax

```
[net] set RPDO <nr> <COB-ID> <tx-type> <nr-of-data> <datatype1>[..<datatype64>]
```

```
<tx-type> ::= "rtr" | "event" | "sync<0..240>"
```

Setup a PDO for receiving data. The datatype maps to an object in the object directory. On reception of an RPDO the RPDO Event is triggered. The receive PDO is seen from the side of the CANopen Gateway Server.

A PDO can be deactivated by writing using value 0x80000000 for the cob.

TCP/IP clients do not have separate namespaces for PDO, i.e. if client 1 has setup PDO 1 and 2 and a client 2 connects and also sets up PDO 1 and 2 it alters the PDO definition of client 1.

It is allowed to set all possible COB-Id. No extra checking is carried out. This means that if the same COB-ID is used for RPDO and TPDO it may happen that no PDO is received.

3.2.2. Configure TPDO command

Command syntax

```
[net] set tpdo <nr> <COB> <tx-type> <nr-of-data> <datatype1>[..<datatype64>]
```

Setup a PDO for sending data. The datatype maps to an object in the object directory. The transmit PDO is seen from the side of the CANopen Gateway Server.

A PDO can be deactivated by writing using value 0x80000000 for the cob.

TCP/IP clients do not have separate namespaces for PDO, i.e. if client 1 has setup PDO 1 and 2 and a client 2 connects and also sets up PDO 1 and 2 it alters the PDO definition of client 1.

It is allowed to set all possible COB-Id. No extra checking is carried out. This means that if the same COB-ID is used for RPDO and TPDO it may happen that no PDO is received.

3.2.3. Read PDO data command

Command syntax

```
[net] r[ead] p[do] <nr>
```

Requests PDO via an RTR CAN message. A Receive RTR-PDO has to be setup first.

3.2.4. Write PDO data command

Command syntax

```
[net] w[rite] p[do] <nr> <nr-of-data> <value1>[..<value64>]
```

Sends given data with a PDO. The PDO has to be setup before.

3.2.5. RPDO Event

Command syntax

```
[net] PDO <nr> <nr-of-data> <value1>[..<value64>]
```

A PDO was received. The mapped data is given as arguments formatted as decimal values The PDO has to be setup, previously.

3.3. CANopen NMT commands

The CANopen Gateway Server has linke any othe CANopen device a node id itself (see command line parameters). Node id 0 and 0x80 have special meanings. A NMT command to node id 0 adresses all CANopen nodes in the network including the server. A NMT command to node id 0x80 adresses all CANopen nodes in the network excluding the server. The server remains in the NMT state that was issued before.

3.3.1. Start Node

Command syntax

```
[[net] node] start
```

Sends the OPERATIONAL command.

3.3.2. Stop node

Command syntax

```
[[net] node] stop
```

Sends the STOPPED command.

3.3.3. Set node Preoperational

Command syntax

```
[[net] node] preop[erational]
```

Sends the PREOPEATIONAL command.

3.3.4. Reset node

Command syntax

```
[[net] node] reset node
```

Sends the RESET NODE command.

In case a SDO transfer is currently in progress that was started by another TCP client then "Reset node" is not carried out. If the SDO transfer was started by the TCP client that also sent the "Reset node" command the SDO transfer is aborted and the reset is carried out.

Note: With node number 0 the gateway is reset, too, i.e. the PDO configuration is lost.

3.3.5. Reset communication

Command syntax

```
[[net] node] reset comm[unication]
```

Sends the RESET COMMUNICATION command.

See also Reset Node Siehe auch Knoten zurücksetzen

3.3.6. Enable node guarding

Command syntax

```
[[net] node] enable guarding <guardingtime> <lifetimefactor>
```

Activates the node guarding error control mechanism. If the master doesn’t receive a response of the slave in the given time it sends the event triggered message ERROR 200 or ERROR 201.

Accuracy of the Guarding depends on the timer resolution. On Linux platforms it can be adjusted with the command line parameter ’-s’. On Windows it is fixed.

3.3.7. Disable node guarding

Command syntax

```
[[net] node] disable guarding
```

Deactivates the node guarding error control mechanism.

3.3.8. Enable heartbeat

Command syntax

```
[[net] node] enable heartbeat <heartbeattime>
```

Start heartbeat dectection on the CANopen Gateway Server. It is checked that heartbeat messages are received in the given time. If a heartbeat message is not received an event triggered ERROR message is sent. The time is given in milliseconds.

The heartbeat time of the producer has to be set with a separate SDO Download Command.

Accuracy of the Guarding depends on the timer resolution. On Linux platforms it can be adjusted with the command line parameter '-s'. On Windows it is fixed.

3.3.9. Disable heartbeat

Command syntax

```
[[net] node] disable heartbeat
```

Stop heartbeat dectection on the CANopen Gateway Server.

3.3.10. Error control Event

Command syntax

```
[[net] node] ERROR <internal-error-code>
```

On detection of NMT errors or CAN errors this message is sent to a connected client.

3.4. Device Failure management

3.4.1. Read device error command

NOT implemented

3.4.2. Emergency Event

Command syntax

```
[[net] node] EMCY <emcy-code> <error-register> <m-error-code>
```

```
<m-error-code> ::= <UNSIGNED8> <UNSIGNED8> <UNSIGNED8> <UNSIGNED8> <UNSIGNED8>
```

On reception of a CANopen EMCY message this message is sent.

Note: Reception has to be enabled. See command extension.

3.5. CANopen interface configuration commands

3.5.1. Initialize gateway

Command syntax

```
[net] init <bitrate>
```

Sets the bitrate the CANopen Gateway Server is working with. The bitrate can be given as value or table index of the default CANopen table.

Bitrate	Table index
1000	0
800	1
500	2
250	3
125	4
reserved	5
50	6
20	7
10	8
Autobaud	9

Autobaud is not supported.

3.5.2. Store configuration

Command syntax

```
[net] store <specifier>
```

NOT implemented.

3.5.3. Restore configuration

Command syntax

```
[net] restore <specifier>
```

NOT implemented.

3.5.4. Set heartbeat producer

Command syntax

```
[net] set heartbeat <ms>
```

Start sending hearbeat messages of the CANopen Gateway Server. The time is given in milliseconds.

3.5.5. Set node id

Command syntax

```
[net] set id <value>
```

NOT supported. Can only be set at start. See commandline options

3.5.6. Start emergency consumer

See port extension.

3.5.7. Stop emergency consumer

See command extension.

3.6. Gateway management commands

3.6.1. Set default newtork

Command syntax

```
[net] set network <value>
```

Set default network. When set the network parameter can be omitted at following commands.

3.6.2. Set default node id

Command syntax

```
[net] set node <value>
```

Set default remote node id. When set the node parameter can be omitted at following commands.

3.6.3. Get version

Command syntax

```
info version
```

Request version of the CANopen Gateway Server. The version is given in the format:

```
<version-string> ::= <vendor-id> <product-code>
                     <version-high>.<version-low> <serial-number>
                     <network-class> <protocol-version>
                     <implementation-class>
```

4. Command Extensions

The standard DS309-3 allows user specific commands and event triggered messages. All extensions provided by the port CANopen Gateway Server are prepended with the prefix: _port_. Exception are the info commands since it is already available.

The CANopen Gateway Server allows multiple TCP/IP connections. This can be problematic if two clients don't know from each other and send NMT commands like start and preop. To handle this situation one client can set a master flag and thus restrict others that way that they can not send NMT or LSS commands. However, to be compliant with the standard this only works with port specific commands. The "normal" DS309-3 commands do not know of the master flag.

Especially for multiclient environments the CANopen Gateway Server provides `register` commands to hook up on a running server and register for already configured PDO, heartbeat, nodeguarding, EMCY and TIME services.

4.1. Status and Configuration

4.1.1. Set master

Command syntax

```
_port_set master <0|1>
```

Activate master mode for TCP/IP client. The master flag can only be set from one client.

4.1.2. Set verbose response

Command syntax

```
_port_set verbose 0xff
```

Show verbose error messages.

4.1.3. Get default network

Command syntax

```
info net[work]
```

Shows default network used for commands.

4.1.4. Get default node

Command syntax

```
info node
```

Shows default node used for commands.

4.1.5. Get Build information

Command syntax

```
info build
```

Returns a string that contains the build date, version and additional information.

Example

```
5.3.10, Nov 13 2012, CPC-PCI can4linux,
```

4.1.6. Get CANopen node Id of server

Command syntax

```
info id
```

Returns CANopen node Id of the server. See also command line parameters.

4.1.7. Get CAN bitrate

Command syntax

```
info bitrate
```

Returns CAN bitrate of the server.

4.1.8. Get SDO timeout

Command syntax

```
[node] info sdo_timeout
```

Shows the node specific SDO timeout. A value of 0 for node is not allowed.

4.1.9. Get TPDO setup

Command syntax

```
info tpdo
```

Shows the TPDO configuration in form of a comma separated list. `<pdoNr> <cobId> <nrOfElements> <datatype1> .. <datatypeN>[,<pdoNr> <cobId> <nrOfElements> <datatype1> .. <datatypN>]`

4.1.10. Get RPDO setup

Command syntax

```
info rpdo
```

Shows the RPDO configuration in form of a comma separated list. `<pdoNr> <cobId> <nrOfElements> <datatype1> .. <datatypeN>[,<pdoNr> <cobId> <nrOfElements> <datatype1> .. <datatypN>]`

4.2. SDO

4.2.1. Register SDO Server Write Indikation

Command syntax

```
_port_reg sdoserverind
```

Register for receiving of SDO Server Write Indikation
The indication is called for all objects at manufacturer area.

Numeric values are indicated as:
SDOSERV <index>:<subindex> size: <len> data: <value>

Non-Numeric values are indicated as:
SDOSERV <index>:<subindex> b64size: <len> data: <data>
<data> are codes as b64.

4.2.2. Unregister SDO Server Write Indikation

Command syntax

```
_port_unreg sdoserverind
```

Unregister for receiving of SDO Server Write Indikation

4.3. PDO

4.3.1. Register PDO

Command syntax

```
_port_reg rpdo <pdoNr>
```

Register for receiving an RPDO.

4.3.2. Unregister PDO

Command syntax

```
_port_unreg rpdo <pdoNr>
```

Unregister an RPDO.

4.4. NMT-Master

4.4.1. Register Nodeguarding

Command syntax

```
[net] [node] _port_reg guarding
```

Register for receiving Nodeguarding events. Events are only sent when a protocol violation has occured.

4.4.2. Unregister Nodeguarding

Command syntax

```
[net] [node] _port_unreg guarding
```

Stop receiving nodeguarding events.

4.4.3. Register heartbeat

Command syntax

```
[net] [node] _port_reg heartbeat
```

Register for receiving heartbeat events. Events are only sent when a protocol violation has occured or a new heartbeat start was detected.

4.4.4. Unregister heartbeat

Command syntax

```
[net] [node] _port_unreg heartbeat
```

Stop receiving heartbeat events.

4.4.5. Register emergency

Command syntax

```
[net] [node] _port_reg emcy
```

Register for receiving an emergency events.

4.4.6. Unregister emergency

Command syntax

```
[net] [node] _port_unreg emcy
```

Stop receiving emergency events.

4.4.7. Enable Sync Producer

Command syntax

```
_port_enable sync <cobid> <sync-cycle_in_µs> [<sync_counter>]
```

Start sending SYNC messages. The master flag has to be set for this command. SYNC cycle time is specified in micro seconds.

4.4.8. Disable Sync Producer

Command syntax

```
_port_disable sync
```

Stop sending SYNC messages. The master flag has to be set for this command.

4.5. LSS Master

The CANopen Gateway Server can configure LSS slaves with the LSS commands. For all commands the master flag has to be set. The CANopen Gateway Server own identity object can be read with the `info version` command.

It is recommended that LSS services should be carried out in STOPPED state.

4.5.1. Switch Selective

Command syntax

```
[net] _port_lss switch_sel <vendorId> <product> <revision> <serialNo>
```

Set single LSS slave in CONFIGURATION state.

4.5.2. Switch global

Command syntax

```
[net] _port_lss switch_glob <0|1>
```

Set complete network in CONFIGURATION or OPERATION state.

4.5.3. Configuration of node id

Command syntax

```
[net] _port_lss set_node <nodeId>
```

Set the node id of an LSS slave.

4.5.4. Request node

Command syntax

```
[net] _port_lss get_node
```

Get the node id of an LSS slave.

4.5.5. Identify LSS slaves

Command syntax

```
[net] _port_lss identity <vendorId> <product> <rev lo> <rev hi>
                         <serial low> <serial hi>
```

Scans the network for nodes that are in the given address range.

4.5.6. Bitrate Configuration

Command syntax

```
[net] _port_lss set_bitrate <sl_table_sel> <sl_table_idx>
                            [<gw_table_sel> <gw_table_idx>]
```

Set the new bitrate of an LSS slave. The LSS slave has to be in state CONFIGURATION.

The first two parameter define the bitrate of the LSS slave. The last two parameter define the bitrate of the CANopen Gateway Server. They are used when autobaud is to be set at the LSS slaves.

Bitrate	Table index
1000	0
800	1
500	2
250	3
125	4
reserved	5
50	6
20	7

10	8
Autobaud	9

Only table 0, the standard CANopen table, is supported by the CANopen Gateway Server.

4.5.7. Bitrate activation

Command syntax

```
[net] _port_lss activate_bitrate <time>
```

Activates the bitrate. The CANopen Gateway Server responds after 2 * time is elapsed. The time is given in milli seconds.

4.5.8. Store Configuration

Command syntax

```
[net] _port_lss store
```

On reception of this command the LSS slave saves the bitrate and node id. The LSS slave has to be in CONFIGUATION state.

4.5.9. Identify unconfigured LSS slaves

Command syntax

```
[net] _port_lss identity_non_cfg
```

Invokes all unconfigured LSS slaves to send a message. On reception of this message the CANopen Gateway Server will send the LSS event.

4.5.10. LSS Event

Command syntax

```
USER LSS <no>
```

On detection of an unconfigured LSS slave the CANopen Gateway Server sends the LSS Event.

no	Description
1	unconfigured device detected

4.6. Miscellaneous

Manchmal ergibt sich die Notwendigkeit ohne Bezug auf CANopen, ohne ein PDO zu konfigurieren, einen beliebigen CAN Frame zu senden. Dazu kann das Kommando

_port_wr benutzt werden. *Command syntax*

```
_port_wr <cob-id> <type> <length> <data0> ... <data7>
```

length	number of bytes	
type	Frame specifier, 2 characters	
	first character, frame format	
	[sS]	standard or base frame format
	[xX]	extended base frame format
	second character, frame type	
	[dD]	data frame
	[rR]	RTR frame

5. Appendix

5.1. Object Directory Overview

Index (hex)	Object	Name	Type	Access
1000	VAR	Device Type	UNSIGNED32	const
1001	VAR	Error Register	UNSIGNED8	ro
1018	VAR	Identity Object	IDENTITY	const
1002	VAR	Manufacturer Status Register	UNSIGNED32	ro
1003	ARRAY	Pre-defined Error Field	UNSIGNED32	ro
1005	VAR	COB-ID SYNC	UNSIGNED32	rw
1006	VAR	Communication Cycle Period	UNSIGNED32	rw
1007	VAR	Synchronous Window Length	UNSIGNED32	rw
1008	VAR	Manufacturer Device Name	VISIBLE_STRING	const
1009	VAR	Manufacturer Hardware Version	VISIBLE_STRING	const
100A	VAR	Manufacturer Software Version	VISIBLE_STRING	const
100C	VAR	Guard Time	UNSIGNED16	rw
100D	VAR	Life Time Factor	UNSIGNED8	rw
1010	ARRAY	Store Parameter Field	UNSIGNED32	rw
1011	ARRAY	Restore Default Parameters	UNSIGNED32	rw
1012	VAR	COB-ID Time Stamp	UNSIGNED32	rw
1013	VAR	High Resolution Time Stamp	UNSIGNED32	rw
1016	ARRAY	Heartbeat Consumer Entries	UNSIGNED32	rw
1017	VAR	Producer Heartbeat Time	UNSIGNED16	rw
1019	VAR	Synchronous counter overflow value	UNSIGNED8	rw
1028	ARRAY	Emergency Consumer	UNSIGNED32	rw
1200	ARRAY	Server SDO Parameter	UNSIGNED32	rw

Index (hex)	Object	Name	Type	Access
1201	ARRAY	Server SDO Parameter	UNSIGNED32	rw
1280 - 12FF	ARRAY	Client SDO Parameter	SDO_PARAMETER	rw
1400 - 143F	ARRAY	Receive PDO Communication Parameter 1	PDO_COMM_PARA	rw
1600 - 163F	ARRAY	Receive PDO Mapping Parameter 1	PDO_MAPPING	rw
1800 - 183F	ARRAY	Transmit PDO Communication Parameter 1	PDO_COMM_PARA	rw
1A00 - 1A3F	ARRAY	Transmit PDO Mapping Parameter 1	PDO_MAPPING	rw
1F50	ARRAY	Download Program Data	DOMAIN	rw
1F51	ARRAY	Program Control	UNSIGNED8	rw
1F80	VAR	NMT Startup	UNSIGNED32	rw
1F81	ARRAY	Slave Assignment	UNSIGNED32	rw
1F82	ARRAY	Request NMT	UNSIGNED8	rw
1F83	ARRAY	Request Guarding	UNSIGNED8	rw
1F84	ARRAY	Device Type Identification	UNSIGNED32	rw
1F85	ARRAY	Vendor Identification	UNSIGNED32	rw
1F86	ARRAY	Product Code	UNSIGNED32	rw
1F87	ARRAY	Revision Number	UNSIGNED32	rw
1F88	ARRAY	Serial Number	UNSIGNED32	rw
1F89	VAR	Boot Time	UNSIGNED32	rw
2000	ARRAY	UNSIGNED8 PDOs	UNSIGNED8	rw
2001	ARRAY	UNSIGNED16 PDOs	UNSIGNED16	rw
2002	ARRAY	UNSIGNED32 PDOs	UNSIGNED32	rw
2003	ARRAY	INTEGER8 PDOs	INTEGER8	rw
2004	ARRAY	INTEGER16 PDOs	INTEGER16	rw
2005	ARRAY	INTEGER32 PDOs	INTEGER32	rw
2006	ARRAY	REAL32 PDOs	REAL32	rw
2007	ARRAY	UNSIGNED24 PDOs	UNSIGNED24	rw

Index (hex)	Object	Name	Type	Access
2008	ARRAY	UNSIGNED40 PDOs	UNSIGNED40	rw
2009	ARRAY	UNSIGNED48 PDOs	UNSIGNED48	rw
200A	ARRAY	UNSIGNED56 PDOs	UNSIGNED56	rw
200B	ARRAY	UNSIGNED64 PDOs	UNSIGNED64	rw
2100	ARRAY	Short Message Service Object	VISIBLE_STRING	rw
2200	VAR	ostring	OCTET_STRING	wo

5.2. Examples

All examples have been carried out in commandline mode.

SDO Access

Access several indices of a CANopen node.

```
COM Shell > 32 r 0x1018 0 u32
COM Shell > [0] 0x4

COM Shell > 32 r 0x3000 0 u8
COM Shell > [0] 0x0

COM Shell > 32 w 0x3000 0 u8 12
COM Shell > [0] OK

COM Shell > 32 r 0x3000 0 u8
COM Shell > [0] 0xc
```

Set default node id. Access numerical values and string values.

```
COM Shell > set node 32
[0] OK

COM Shell > r 0x3000 0 u8
COM Shell > [0] 0xc

COM Shell > w 0x3010 0 i8 0xff
COM Shell > [0] OK

COM Shell > r 0x3010 0 i8
COM Shell > [0] -1

COM Shell > w 0x3010 0 i8 0xf9
COM Shell > [0] OK

COM Shell > r 0x3010 0 i8
COM Shell > [0] -7

COM Shell > r 0x1008 0 vs
COM Shell > [0] S16-LINUX

COM Shell > r 0x1008 0 os
COM Shell > [0] UzE2LUxJTlVYAAAAAAAAAAAAAAA=
```

Read a domain value.

```
COM Shell > w 0x2002 0 "test1.dat"
[0] ERROR 101 Syntax Error - Bad Para Count
```

Write a domain value.
The raw data has to be converted to base64 format. The Linux-Tool base64 is used for this.

```
COM Shell > w 0x2002 0 vs test1.dat
COM Shell > [0] OK
```

```
> echo 'Hello World!' | base64
SGVsbG8gV29ybGQhCg==
```

```
COM Shell > w 0x2000 0 d SGVsbG8gV29ybGQhCg==

[0] OK
COM Shell > r 0x2000 0 d
COM Shell > 32 USER BLOCK 1 1

[0] MTIzMTIzMTIzMTIzMTIzMTIzMTIzMTIzMTIzMTIzMTIzMQ==
```

PDO Konfiguration

Initialize PDO 1 and deinitialize PDO 1. Initialize PDO 2 and start network.

```
COM Shell > set rpdo 1 0x181 event 1 u8
[0] OK

COM Shell > set rpdo 1 0x80000181 event 1 u8
[0] OK

COM Shell > set rpdo 2 0x181 event 1 u8
[0] OK

COM Shell > 0 start
[0] OK
```

Reception of PDO 2. Switch off verbose error message. Initialize PDO 3. Switch on verbose error message. Initialize PDO 3.

```
COM Shell > PDO 2 1 0xaa

COM Shell > _port_set verbose 1
[0] OK

COM Shell > set rpdo 3 0x182 event 2 u8
[0] ERROR 100

COM Shell > _port_set verbose 0xff
[0] OK

COM Shell > set rpdo 3 0x182 event 2 u8
[0] ERROR 100 Syntax Error - Invalid data
```

```
COM Shell > set rpdo 3 0x182 event 2 u8 u16
[0] OK
```

NMT Error control

Error control with Heartbeat. Node is switched off and started again (Boot up). Node has change NMT state.

```
COM Shell > 32 enable heartbeat 1200
[0] OK

COM Shell > 32 ERROR 203 Heartbeat lost

32 ERROR 205 Boot up

32 ERROR 202 Heartbeat started

COM Shell > 32 ERROR 204 new NMT state  127
```

Emergency

Switch on reception of emergency messages.

```
COM Shell > 32 _port_reg emcy
[0] OK

32 EMCY 0xff00 0x81 0xaa 0x0 0x12 0xaf 0x0
```

LSS node configuration

Switch on verbose error messages. Set master flag. LSS Event received. Change to NMT state STOPPED (recommended). Start identifying LSS slave. Switch LSS slave into configuration mode

```
COM Shell > _port_set verbose 0xff
[0] OK

COM Shell > _port_set master 1
[0] OK

COM Shell > USER LSS 1 unconfigured device(s) detected

COM Shell > 0 stop
[0] OK

COM Shell > _port_lss identity 0x34 12345 0 2 0 2
[0] OK

COM Shell > _port_lss identity 0x34 12345 0 1 0 1
[0] OK

COM Shell > _port_lss identity 0x34 12345 1 1 1 1
```

```
[0] OK

COM Shell > _port_lss switch_sel 0x34 12345 1 1
[0] OK
```

Request node id. Set node id and check it. Switch to operation mode. Change to NMT state peroperational. Access node with SDO commands.

```
COM Shell > _port_lss get_node
[0] 255

COM Shell > _port_lss set_node 32
[0] OK

COM Shell > _port_lss get_node
[0] 32

COM Shell > _port_lss switch_glob 0
[0] OK

COM Shell > 0 preop
[0] OK

COM Shell > 32 r 0x1018 1 u32
COM Shell > [0] 0x34

COM Shell > 32 r 0x1018 2 u32
COM Shell > [0] 0x3039

COM Shell > 32 r 0x1018 3 u32
COM Shell > [0] 0x1

COM Shell > 32 r 0x1018 4 u32
COM Shell > [0] 0x1

COM Shell > 32 w 0x2000 0 u8 1
COM Shell > [0] ERROR 0x06010002  Access Error - write a read only object

COM Shell > 32 r 0x2000 0 u8
COM Shell > [0] 0x20
```

Gateway management

Setup PDO and request configuration. Request version information. Request default node. Request SDO timeout from default node. Request SDO timeout from node 32.

```
COM Shell > set rpdo 2 0x181 event 1 u8
[0] OK

COM Shell > info rpdo
```

```
[0] 2 0x181 event 1 u8

COM Shell > set tpdo 1 0x200 event 1 u8
[0] OK

COM Shell > set tpdo 2 0x300 event 2 u32 u32
[0] OK

COM Shell > info tpdo
[0] 1 0x200 event 1 u8, 2 0x300 event 2 u32 u32

COM Shell > info version
[0] 52 410640 4.2 0 3 1.00 0.0

COM Shell > info node
[0] 127

COM Shell > info sdo_timeout
[0] 1000

COM Shell > 32 set sdo_timeout 1500
[0] OK

COM Shell > 32 info sdo_timeout
[0] 1500
```

Transmission of layer-2 CAN frames

The example shows how the command *_port_wr* can be used to send raw CAN frames.

```
COM Shell > _port_wr 0x1111 xd 4 1 2 3 4
[0] OK

COM Shell > _port_wr 0x1111 sr 5
[0] OK

COM Shell > _port_wr 0x111 sd 3 10 16 0xaa
[0] OK
```

6. Index