CiA 309

Access from other networks

Part 3: Mapping

Version: 1.1.0

12 December 2006

HISTORY

Date	Changes
2004-09-15	*Publication of version 1.0 as draft standard proposal*
2006-12-12	*Publication of version 1.1 as draft standard*
	Minor editorial corrections and clarifications.

General information on licensing and patents

CAN in AUTOMATION (CiA) calls attention to the possibility that some of the elements of this CiA specification may be subject of patent rights. CiA shall not be responsible for identifying any or all such patent rights.

Because this specification is licensed free of charge, there is no warranty for this specification, to the extent permitted by applicable law. Except when otherwise stated in writing the copyright holder and/or other parties provide this specification “as is” without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the correctness and completeness of the specification is with you. Should this specification prove failures, you assume the cost of all necessary servicing, repair or correction.

Trademarks

CANopen® and CiA® are registered community trademarks of CAN in Automation. The use is restricted for CiA members or owners of CANopen vendor ID. More detailed terms for the use are available from CiA.

CAN in Automation e. V.
Kontumazgarten 3
DE - 90429 Nuremberg, Germany
Tel.: +49-911-928819-0
Fax: +49-911-928819-79
Url: www.can-cia.org
Email: headquarters@can-cia.org

Contents

1 Scope

This specification specifies the services and protocols to interface CANopen networks to a TCP/IP-based network.

This set of specifications is organized as follows:

- Part 1: General principles and services
- Part 2: Modbus/TCP mapping
- Part 3: ASCII mapping

This part of the specification defines the ASCII-based communication syntax for CANopen gateway devices. The aim is to provide a lightweight counterpart to solutions with CORBA or OPC.

2 References

The references given in part 1 shall apply to this part, too.

7CiA301/	CiA 301, CANopen application layer and communication profile
/CiA309-1/	CiA 309:2006, Interfacing CANopen with TCP/IP – Part 1: General principles and services (V1.1)
/ISO/IEC 646/	ISO/IEC 646, 1991 Information technology – ISO 7-bit coded character set for information interchange
/ISO/IEC 9899/	ISO/IEC 9899, 1999 Programming languages – C
/RFC 2045/	RFC 2045 Multipurpose internet mail extensions

3 Abbreviations and definitions

3.1 Abbreviations

The abbreviations given in part 1 shall apply to this part, too.

ASCII	American Standard Code for Information Interchange
BNF	Backus Naur form
CPU	Central Processing Unit
CR	Carriage Return
CRLF	Carriage Return and Line Feed
LF	Line Feed

3.2 Definitions

3.2.1 General

The definitions given in part 1 shall apply to this part, too.

Command

controls the gateway 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 “[“, “]”.

Note: In the given examples it is assumed that network address and node address are preset.

Data type syntax

The mandatory data types shall be supported.

Table 1: Syntax and CANopen data types

Syntax	CANopen Type	Category
b	Boolean	Mandatory
u8	Unsigned8	Mandatory
u16	Unsigned16	Mandatory

Syntax	CANopen Type	Category
u24	Unsigned24	Optional
u32	Unsigned32	Mandatory
u40	Unsigned40	Optional
u48	Unsigned48	Optional
u56	Unsigned56	Optional
u64	Unsigned64	Optional
i8	Integer8	Mandatory
i16	Integer16	Mandatory
i24	Integer24	Optional
i32	Integer32	Mandatory
i40	Integer40	Optional
i48	Integer48	Optional
i56	Integer56	Optional
i64	Integer64	Optional
r32	Real32	Optional
r64	Real64	Optional
t	Time of day (with two arguments: day ms)	Optional
td	Time difference	Optional
vs	Visible string	Optional
os	Octet string	Optional
us	Unicode string	Optional
d	Domain	Optional

The value of the data type *domain*, *octet string*, and *unicode string* shall be encoded in mime-base64 as specified in /RFC 2045/. All wrapping CRLF shall be stripped from the encoded data to have one long string.

whitespace

As specified in /ISO/IEC 9899/ except of CR and LF.

Note: A visible string with whitespace is enclosed with double quotes to denote it as single argument of the command. If a double quote is used within the string, the quotes are escaped by a second quotes, e.g. “Hello “”World””, CANopen is great”.

3.2.2 Command structure

The principle communication is based on non-case-sensitive ASCII strings /ISO/IEC 646/ instead of architecture and CPU/compiler depending binary structures. Due to this, no application handles with things like endianness, data size and byte alignment. In all cases where numbers are used, the typical representation like specified in /ISO/IEC 9899/.

100 - decimal, starting with a number

0x64 - hexadecimal, starting with the string 0x

1.22 - float

.22e10 - float

22e3 - float

3.2.2.1 Request

The CANopen gateway is controlled by commands. A command is composed of tokens, which are separated by any number of whitespaces and is closed with a CRLF.

All commands are confirmed. Commands start with a sequence number which is enclosed by square brackets [ ]. The sequence number is a 4-byte value. It is not used for event-triggered messages. According to the addressing principle, a network number and a node number follow the sequence number. Network number and node number are optional, when the CANopen gateway only provides one Ethernet/CAN interface or when a client presets them. Commands that affect only the server not a remote node but a net and node are given net and node are ignored. 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> | <compound-parameter>
<compound-parameter>  ::= <value> <parameter>
<map-object>          ::= <datatype> | <multiplexor>
<datatype>            ::= 'b' | 'u8' | 'u16' | 'u32' | 'u40' | 'u48' |
                          'u56' | 'u64' | 'i8' | 'i16' | 'i24' | 'i32' |
                          'i40' | 'i48' | 'i56' | 'i64' | 'r32' | 'r64' |
                          't' | 'td' | 'vs' | 'os' | 'us' | 'd'
<multiplexor>         ::= <index> <subindex>
<index>               ::= UNSIGNED16
<subindex>            ::= UNSIGNED8
```

Net numbers are starting with 1. Node numbers are starting with 1. The value 0 for net or node is used to address all networks or all nodes.

The token <value> designates a value of the possible CANopen data types.

Within the description of the commands the sequence number is omitted for reasons of readability.

3.2.2.2 Response

The CANopen gateway shall respond with the same sequence number at the first position as given by the request. This number shall be given in decimal format. There shall be only one response to a request.

```
<command-response>    ::= "["<sequence>"]" <response>
<response>                  ::= <value> | <error-string> | <emcy-list>
                | "OK"
<error-string>        ::= "Error:" <error code>
<error-code>          ::= <internal-error-code> | <sdo-abort-code>
<internal-error-code>::= see table 2
<sdo-abort-code>      ::= UNSIGNED32
<emcy-list>           ::= [<emcy1> " "..<emcy254>]
```

The sdo-abort-codes (SAC) are defined in /CiA301/. Allowed internal-error-codes (IEC) are listed in Table 2:

Table 2 - Internal error code

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

103	Time-out (where applicable)
200	Lost guarding message
201	Lost connection
202	Heartbeat started
203	Heartbeat lost
205	Boot-up
300	Error passive
301	Bus off
303	CAN buffer overflow
304	CAN init
305	CAN active (at init or start-up)
400	PDO already used
401	PDO length exceeded

Note: After bus-off, the command init should be invoked to reset the CAN controller.

3.2.2.3 Event triggered messages

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

```
<event-trigged-message> ::= [[net] node] <event-specifier> <parameter>
<event-specifier>       ::= "EMCY" | "ERROR" | "PDO" | "SYNC" | "USER"
```

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

4 Network access command specification

4.1 SDO access commands

4.1.1 General

The following command definitions shall be used to implement the SDO access services as defined in part 1.

SDO access services are addressing a specific object at an SDO server via index and sub-index and a transfer data type.

4.1.2 Upload SDO command

Indication syntax:

```
[[net] node] r[ead] <multiplexor> <datatype>
```

Examples:

```
[21] r 0x1000 0 u32
[4096] read 0x1008 0 vs
```

Response syntax:

See chapter 3.2.2.2

4.1.3 Download SDO command

Indication syntax:

```
[[net] node] w[rite] <multiplexor> <datatype> <value>
```

Examples:

```
[20] 1 23 w 0x1016 0 u16 100
```



```
[23] write 0x1016 0 u16 0x64
```

Response syntax:

See chapter 3.2.2.2

4.1.4 Configure SDO timeout command

The timeout delay time for abort error code 'SDO protocol timed out', used by the gateway's SDO client, may be set.

Indication syntax:

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

Response syntax:

See chapter 3.2.2.2

4.2 PDO access commands

4.2.1 General

The following command definitions shall be used to implement the PDO access services as defined in part 1. Normally a PDO is first configured before transmission and reception is possible.

A PDO is seen from the view of the gateway. An RPDO therefore receives data from the CANopen network and a TPDO sends the data into the CANopen network.

Note: In order to disable/delete a gateway PDO, the bit 31 in the COB-ID is used. For details see /CiA301/.

4.2.2 Configure RPDO command

Indication syntax:

```
[[net] node] set rpdo <nr> <COB> <tx-type> <nr-of-data> <map-obj1>
[..<map-obj64>]
<tx-type> ::= "rtr" | "event" | "sync<0..240>"
```

Note: In case a `<map-obj>` is given in form of index and subindex, i.e. `<multiplexor>`, they are counted as 1 in the `<nr-of-data>`. The `<nr>` is an offset; it starts with 1.

Example:

```
[12] set rpdo 1 0x180 event 3 u8 u8 u16
[24] 2 set rpdo 1 0x180 event 3 u8 u8 i16
```

Response syntax:

See chapter 3.2.2.2

4.2.3 Configure TPDO command

Indication syntax:

```
[[net] node] set tpdo <nr> <COB> <tx-type> <nr-of-data> <map-obj1>
[..<map-obj64>]
```

Note: In case a `<map-obj>` is given in form of index and subindex, i.e. `<multiplexor>`, they are counted as 1 in the `<nr-of-data>`. The `<nr>` is an offset; it starts with 1.

Example:

```
[13] set tpdo 1 0x201 rtr 4 u8 u16 u16 u8
```

Response syntax:

See chapter 3.2.2.2

Note: It is not recommended to support RTR. Therefore it is recommended to set bit 30 of the COB-ID accordingly. For details see /CiA301/.

4.2.4 Read PDO data command

Indication syntax:

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

If the transmission type is different than RTR, the gateway answers with the values of the mapped objects.

Response syntax:

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

4.2.5 Write PDO data command

Indication syntax:

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

Response syntax:

See chapter 3.2.2.2

4.2.6 RPDO received command

Indication syntax:

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

Examples:

```
1 pdo 1 2 123 4       ;# gateway with more than one network
                      ;# received RPDO1 at net 1 with two objects
                      ;# mapped
pdo 2 1 1234          ;# RPDO2 with one object mapped
pdo 2 3 100 2 4       ;# three objects mapped
```

4.3 CANopen NMT commands

4.3.1 General

The following command definitions shall be used to implement the CANopen NMT services as defined in part 1. The supported services depend on the gateway class.

4.3.2 Start node command

Indication syntax:

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

Response syntax:

See chapter 3.2.2.2

4.3.3 Stop node command

Indication syntax:

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

Response syntax:

See chapter 3.2.2.2

4.3.4 Set node to pre-operational command

Indication syntax

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

Response syntax:

See chapter 3.2.2.2

4.3.5 Reset node command

Indication syntax:

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

Response syntax:

See chapter 3.2.2.2

4.3.6 Reset communication command

Indication syntax:

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

Response syntax:

See chapter 3.2.2.2

4.3.7 Enable node guarding command

Activating node-guarding functionality enables another event-triggered response message to the clients of the gateway. Only in the case that one of the monitored CANopen nodes violates the guarding protocol, an event message shall be sent to the clients.

Indication syntax:

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

Response syntax:

See chapter 3.2.2.2

4.3.8 Disable node guarding command

Indication syntax:

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

Response syntax:

See chapter 3.2.2.2

4.3.9 Start heartbeat consumer command

Activating the heartbeat consumer at the gateway enables another event-triggered response message to the gateway's clients. Only in the case that one of the monitored CANopen nodes violates the guarding protocol, an event message shall be sent to the clients.

Indication syntax:

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

Response syntax:

See chapter 3.2.2.2

4.3.10 Disable heartbeat consumer command

Indication syntax:

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

Response syntax:

See chapter 3.2.2.2

4.3.11 Error control event received command

Response syntax:

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


4.4 Device failure management commands

4.4.1 General

The following command definitions shall be used to implement the device failure management services as defined in part 1.

4.4.2 Read device error command

Indication syntax:

```
[[net] node] r[ead] error
```

Response syntax:

See chapter 3.2.2.2

4.4.3 Emergency event received command

Response syntax:

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

The m(anufacturer)-error-code shall be returned as five decimal values corresponding to the manufacturer-specific error code in the EMCY message.

4.5 CANopen interface configuration commands

4.5.1 General

The following command definitions shall be used to implement the CANopen interface configuration services as defined in part 1. Settings are valid for the default network, if no network address is given. The node address shall be omitted, in case it is given.

4.5.2 Initialize gateway command

Indication syntax:

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

The bit-rate shall be given as table index of the standard CANopen bit-rate table specified in /CiA 305/.

Response syntax:

See chapter 3.2.2.2

4.5.3 Store configuration command

All settings may be stored. Storage of settings may be selective to a special service. If no argument is given all settings shall be stored.

Indication syntax

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

Table 3 - Storage specifier

Storage specifier	Description
CFG	id, bitrate, default node, default net
PDO	PDO-number, transmission type, number of mapping, mapping
SDO	sdo timeout
NMT	- Node guarding: node, guarding time, life time factor
	- Heartbeat: node, heartbeat time
	- Heartbeat time of the server

Response syntax:

See chapter 3.2.2.2

4.5.4 Restore configuration command

Indication syntax:

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

Response syntax:

See chapter 3.2.2.2

4.5.5 Set heartbeat producer command

Indication syntax

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

Response syntax:

See chapter 3.2.2.2

4.5.6 Set node-ID command

Indication syntax

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

Response syntax:

See chapter 3.2.2.2

4.5.7 Start emergency consumer command

This command is not specified.

4.5.8 Stop emergency consumer command

This command is not specified.

4.6 Gateway management commands

4.6.1 General

The following command definitions shall be used to implement the gateway management services as defined in part 1.

4.6.2 Set default network command

Indication syntax

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

Response syntax:

See chapter 3.2.2.2

4.6.3 Set default node-ID command

Indication syntax

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

Response syntax:

See chapter 3.2.2.2

4.6.4 Get version command

Indication syntax:

```
info version
```

The command responses the current version information of the gateway as a whitespace separated list. The first list elements are values normally contained in the gateway's object dictionary at object 1018_h. The following list elements containing the gateway class, as a combination of possible classes and the version number of the implemented protocol corresponded to part 1.

Response syntax:


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

Result example:

```
[1234] 52 100 1.01 1234567 128 0.85 0.10
```

4.7 Controller management commands

4.7.1 General

The following command definitions shall be used to implement the controller management services as defined in part 1.

4.7.2 Reset controller command

This command is not specified.

4.7.3 Start controller command

This command is not specified.

4.7.4 Stop controller command

This command is not specified.

4.8 Manufacturer-specific commands

4.8.1 General

The following command definitions shall be used to implement the manufacturer-specific services. Gateway manufacturer may add commands to their gateway to provide further features. In order to avoid syntax errors due to an unknown command all user-specific commands shall be pre-pended with an underscore “_”.

If there is the need for event-triggered messages not specified in this part of the specification the event specifier “USER” shall be used.

