Specification

EtherCAN CI
communication protocol

EMS Dr. Thomas Wünsche
Sonnenhang 3
D-85304 Ilmmünster
Tel. +49-8441-490260
Fax +49-8441-81860

EMS
Thomas Wünsche

1. Introduction

The communication protocol for the EtherCAN CI device is based on ASCII coded packages formated as shown in the following diagramm:

"S" (1)	Type (1)	Length (1)	Handle (1)	ts_sec (4)	ts_nsec (4)	Data (1 - FFh)	"T" (1)

- Each packet starts with an upper case "S" (53h) and ends with an upper case "T" (54h).
- The numbers in brackets give the amount of bytes in the field "Data".
- The field "Type" defines how the "Data" field has to be interpreted. A packet sent to the EtherCAN is defined as a "command", a packet sent by the EtherCAN is defined as a "message"
- The field "Length" gives the byte count of the field "Data"
- The field "Handle" is unused at the moment
- The fields "ts_sec" and "ts_nsec" form a timestamp when the packet was generated.
- The field "Data" contains the payload

"Type", "Length", "Handle", "ts_sec", "ts_nsec" and "Data" conform to the parts of a CPC_MSG used by the driver as information and data exchange structure.

2. Field Definitions

2.1. Type

The following table holds the different message command types (send):

Command types	Explanation
1	Send CAN data frame
3	Execute interface/driver control command
6	Initialize controller with CAN parameters
8	Clears input queue (see 28)
11	Inquire actual CAN parameters
12	Set filter parameters of controller
13	Send CAN remote frame
14	Send CAN state message
15	Send XCAN data frame
16	Send XCAN remote frame
17	Reset CAN-Controller
18	Inquire Information
19	Open a channel
20	Close a channel
21	Inquire count of messages in message queue
200	Exit the CAN Bus
25	Request the CAN controllers error counter
28	Clear CPC_CMD queue

The following table holds the different message types (receive):

Message types	Explanation
1	Receive CAN data frame
2	Receive busload message
8	Receive CAN remote frame
9	Send acknowledge (TX acknowledge)
10	Receive power-up message
12	Receive actual CAN parameters

13	Command aborted message
14	Receive CAN state message
15	Used to reset the CAN-Controller
16	Receive XCAN data frame
17	Receive XCAN remote frame
18	Receive information string
19	Receive interface/driver control message
20	Response type for confirmed requests
21	Response type for overrun conditions
22	Response type for keep alive conditions
23	Response type for bus error conditions
24	Response type for a disconnected interface
25	RX/TX error counter of CAN controller

2.2. Length

The length of the variable data field with values from 1 Byte to 255 Byte

2.3. Handle

The handle is a one byte proprietary field and is always set to 0x00 at the moment.

2.4. Timestamp

The timestamp is divided into two 4 Byte fields. The first holds the seconds and the second the nanoseconds in unix-time (counting starts on 1.1.1970)

2.5. Variable Data

2.5.1. Standard CAN data message

This message is a data frame with 11-bit identifier (00000000 up to 000007FF) and with a length up to 8 data bytes.

Identifier (4)	Length (1)	Data field (based on length and message type) (1 - 8)

Example:

000007FF	08	01 02 03 04 05 06 07 08

2.5.2. Extended CAN data message

This message is a data frame with 29-bit identifier (00000000 up to 1FFFFFFF) and with a length up to 8 data bytes.

Identifier (4)	Length (1)	Data field (based on length and message type) (1 - 8)

Example:

1A568B74	08	01 02 03 04 05 06 07 08

2.5.3. Standard CAN remote message

This message is a data frame with 11-bit identifier (00000000 up to 000007FF) and with a length up to 8, but without data bytes.

Identifier (4)	Length (1)	Data field is empty

Example:

000007FF	08	

2.5.4. Extended CAN remote message

This message is a data frame with 29-bit identifier (00000000 up to 1FFFFFFF) and with a length up to 8, but without data bytes.

Identifier (4)	Length (1)	Data field is empty

Example:

1A568B74	08	

2.5.5. CAN params message SJA1000 CAN controller

Controller type (1)	acc_code register 0 to 3 (4)	acc_mask register 0 to 3 (4)	Btr0 (1)	btr1 (1)	outp_contr (1)

Example (for btr0 and btr1 see baudrate table on the end of this document):

02	55555555	ffffffff	01	1C	DA

2.5.6. CAN params message PCA82C200 CAN controller

This controller is replaced by SJA1000 CAN controller.

Controller type (1)	acc_code register (1)	acc_mask register (1)	Btr0 (1)	btr1 (1)	outp_contr (1)

Example (for btr0 and btr1 see baudrate table on the end of this document):

01	00	FF	01	1C	DA

2.5.7. Info message

Source of information (1)	Type of info about source (1)	String with request information (max 62 byte)

Source: 01 for interface, 02 for driver, 03 for library
Type: 01 for version, 02 for serial

Example:

01	01	00...

2.5.8. Overrun message

Event (1)	Count of message overruns (1)

Event: 01 for CAN, 02 for CANstate, 03 for buserror

Example:

01	15

2.5.9. Error message

Error code (errframe) (1)	CAN controller (1)	Error capture code register (1)	RX error counter register (1)	TX error count register (1)

Example:

01	02	00	00	00

2.5.10. Error counter message

Error counter for RX (1)	Error counter for TX (1)

Example:

00	00

2.5.11. Busload message

busload (1)

Example:

00

2.5.12. CANstate message

CANstate (1)

Example:

00

3. Log of an example cansrv <–> client session:

The log assumes that a TCP connection to the EtherCAN device has been established before. Therefor the IP address and the port number on which the cansrv program is listening has to be known.

Mandatory for a working connection is to initialize the CAN controller.

All values are hex values.

Initialising the CAN controller
S 06 0d 00 00000000 00000000 02 00 ffffffff ffffffff 04 1c da T

Inquire CAN initialisation parameters of the interface
S 0b 01 00 00000000 00000000 02 T

Response on inquire CAN initialisation parameters of the interface
S 0c 18 00 00000000 00000000 02 00 ffffffff ffffffff 04 1c da T

Execute CPC_Control command
S 03 02 00 00000000 00000000 05 00 T
enables transmission of CAN messages from EtherCAN to client

Execute CPC_Control command
S 03 02 00 00000000 00000000 0d 00 T
enables transmission of CAN status messages from EtherCAN to client

Inquire the version string of the interface
S 12 02 00 00000000 00000000 01 01 T

Inquire the interface serial number
S 12 02 00 00000000 00000000 01 02 T

Response on inquire version string
„EtherCAN Server V1.1.0“
S 12 18 00 00000000 00000000 01 01 457468657243414e20536572766572205631 2e312e30 T

Response on inquire interface serial number
„0000001“
S 12 09 00 00000000 00000000 01 02 30303030303031 T

Send CAN data frame with 11 bit identifier
ID: 000, L:02 Data: 02 00
S 01 07 00 00000000 00000000 00000000 02 0200 T

Send CAN remote frame with 11 bit identifier
ID: 000, Len: 02
S 0d 05 00 00000000 00000000 00000000 02 T

Send CAN data frame with 29 bit identifier
ID: 12345678, Len: 08 Data: 04 01 00 00 00 00 00 00
S 0f 0d 00 00000000 00000000 78563412 08 04 01 00 00 00 00 00 00 T

Send CAN remote frame with 29 bit identifier
ID: 12345678, Len: 08
S 10 05 00 00000000 00000000 78563412 08 T

Receive CAN data frame with 11 bit identifier
ID: 000, L:02 Data: 02 00
S 01 07 00 04680000 c083c401 00000000 02 0200 T

Receive CAN remote frame with 11 bit identifier
ID: 000, Len: 02
S 08 05 00 04680000 30c84d1e 00000000 02 T

Receive CAN data frame with 29 bit identifier
ID: 12345678, Len: 08 Data: 04 01 00 00 00 00 00 00
S 10 0d 00 04680000 90b5de3a 78563412 08 0401000000000000 T

Receive CAN remote frame with 29 bit identifier
ID: 12345678, Len: 08
S 11 05 00 05680000 90a6561d 78563412 08 T

Clear command queue of the interface
S 1c 00 00 00000000 00000000 T

Appendix A: CAN Baudrates

The CAN controller is clocked with 16MHz. We recommend the following settings to achieve the standard CAN baudrates. For other baudrates please refer to CAN controller data sheet [1].

Baudrate [kBaud]	btr0	btr1
1000	00h	14h
800	00h	16h
500	00h	1Ch
250	01h	1Ch
125	03h	1Ch
100	04h	1Ch
50	09h	1Ch
25	13h	1Ch
20	18h	1Ch
10	31h	1Ch

Appendix B: Documents

[1] SJA1000 data sheet, www.semiconductors.philips.com/acrobat/datasheets/SJA1000_3.pdf