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.

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

© 2009 port GmbH
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. Introduction ........................................ 5

2. Common Functionality ............................... 6
   2.1. Long Display Format .............................. 6
   2.2. Short Display Format .............................. 6
   2.3. Additional Outputs .............................. 7
   2.4. Interactive Commands ............................ 7
   2.5. Filter ........................................ 10
   2.6. Trigger ....................................... 11
   2.7. CAN Errors .................................... 11

3. CANserver Horch ................................... 12
   3.1. Commandline Synopsis ........................... 12
   3.2. Description .................................... 12
   3.3. Options ....................................... 13

4. embedded Horch .................................. 16
   4.1. embedded Horch for UConnect-XE164 ........... 16

5. Overview Implementations .......................... 17
   5.1. Overview CANserver Horch ....................... 17
   5.2. Overview embedded Horch ....................... 19
1. Introduction

This manual describes functionality of CANserver Horch and embedded Horch. CANserver Horch is a CAN-to-TCP gateway, while embedded Horch is a CAN-to-RS232 gateway. Both gateways use the same ASCII format on the side of TCP resp. RS232 gateway. Both gateways use the same ASCII format on the side of TCP resp. RS232. For this reason they are both called horch. Due to different architectures and differing requirements does not every implementation support all described features.
2. Common Functionality

2.1. Long Display Format

Received CAN Messages are displayed as ASCII text strings. *LONG DISPLAY FORMAT* is aligned with good readability in a terminal program.

The long format description is:

```
[timestamp] <id-dec>/0x<id-hex> : <type> : 0{<data>}8
```

- **type**: `<frametype> + <datatype>`
- **datatype**: D|R, data or remote frame
- **frametype**: x|s, extended or standard frame format

**example:**

- 991330039.943806 12/0x00c : sD : 80 12 34 0d
- 991330039.944806 12/0x00c : xD : 80 12 34 0d
- 991330039.945806 4660/0x1234 : xR : (length=0)
- 991330039.946806 4660/0x1234 : xD : 01 02 03 04 05 06
- 991330039.947806 4660/0x1234 : xR : (length=4)

The message ID is always displayed in decimal and hexa-decimal. The leading time stamp value is optional and can be activated by an interactive command. The format of the displayed data bytes can be selected by interactive commands from decimal, hexa-decimal or ASCII characters.

2.2. Short Display Format

Received CAN Messages are displayed as ASCII text strings. *SHORT DISPLAY FORMAT* is optimized for short lines and leads to short transfer times. The ASCII text strings make processing in applications that communicate with horch easier.
The short format description is:

```
[<timestamp>]:<idhex>:<type>:<length>:0{<data>}8:<chksum>
```

idhex: Can-Identifier in hex without 0x

type: <frametype> + <datatype>

datatype: D | R , data or remote frame

frametype: x | s , extended or standard frame format

data: databyte in hex without 0x,
   two digits for one Byte,
   no space

chksum: simple one Byte XOR of all chars

hex-values use Capital characters

example:
99.300:620:sD:4:01020304:14
123.976:100:xD:8:01020304050607FF:29
138.959:180:sR:8::3B
156.768:1000056:xR:2::35

The message ID is always displayed in hexa-decimal. The leading time stamp value is optional and can be activated by an interactive command. The format of the displayed data bytes is limited to hexa-decimal.

### 2.3. Additional Outputs

Other messages than received CAN messages start with strings like:

- ERROR: CAN or driver errors
- DEBUG: debug messages received as CAN messages
- INFO: information

All messages are prepended by the time stamp, if enabled.

### 2.4. Interactive Commands

At run-time horch can be controlled through commands from it’s stdin channel (console, RS232 or TCP/IP). Most commands consist of one letter and are used to change formatting of CAN messages. In case stdin comes from the console horch uses the system command stty(1) to switch the console into the cbreak, noecho mode.

? On-line help, show command overview
a  Formatting of data bytes as ASCII characters

b  Change bit rate on-line
   \ \ 1.25 \  \  \  \  \  \  
   Valid bit rate values are specified within the implementation table.

c  print cut-mark

d  Formatting of data bytes as decimal numbers

f  filter setting
   Install a new filter for receive messages. Previous settings are overwritten. The
   format of filter specification is described in chapter Filter

F  Request of current filter settings

h  Formatting of data bytes as hexa-decimal numbers

i  On LINUX Systems an Interpreter program can be started which interprets and
   displays the present content of the logfile. The following command line is e-
   xecuted:
   konvert -L -x std.int -n std.nam logfile | less

l  Toggles the current state of local file logging. Logfiles are not created in server
   mode.

m  acc_code  acc_mask
   Set the content of acceptance and mask register of the CAN controller chip. With
   the help of this command a message filtering based on the CAN chip hardware is
   possible. (see e.g. SJA1000 documentation) \textit{This command is currently only available for can4linux and Janz CAN-USB.}
   \texttt{acc_code} and \texttt{acc_mask} can be a 32 bit value as decimal or hexadecim-
   al number. The interpretation of this values is implementation specific.

N  Switches over to long display format.

o  flag
   Setting of specified options via an option flag. Each bit in the flag has a special
   meaning and influences therefore the behavior of \texttt{horch} or the underlaying CAN
device driver. The following bits are already defined:

Bit 0  self-reception mode, receive self sent CAN frames
Bit 1  Listen Only Mode, don’t acknowledge received CAN frames
Bit 2  Use TimeStamp, return receive time for received messages

If a bit is not set, the behaviour is reset, or the opposite.

Example:
   o1
   o 0x01
   o 0x05

q  quit program (in console mode)
Quit
   quit program (in server mode)

R  Reset the CAN controller, e.g. after an Error CAN Busoff.

s  display statistic information
   No generic information are available. Lines do always start with twice <<. The possible information printed here are depending on the CAN controller type and the layer-2 CAN driver. The first column contains the name of the used CAN controller in plain text, followed by CAN controller specific register values. For the most often used CAN controller SJA1000 a statistic line looks like this:
   :: sja1000 <act baud rate> <status register> <error_warning limit> <rx errors> <tx errors> <error code> <buslast>

Example:
   :: sja1000 125 12 96 0 0 166 0.0

S  Switches over to short display format.

t  activate display of time stamps.

T  deactivate display of time stamps.

x  change trigger settings

   Format: x <idx> <mask> [r] <id> [<data>]
   (see chapter Trigger)
y activates CAN message trigger.
Before using this command valid trigger values should be set (-x).

Y deactivates trigger settings

w and

W sends a CAN message
A CAN message is sent. All of the letters following the command letter are interpreted as arguments. The capital command letter W is used to send in extended frame format (using 29 bits) and the noncapital command letter w is used for the base frame format (using 11 bits). If the letter r is following the command letter as first argument, an RTR message is sent.

w [r] <id> 0{<data>}8

w 222 0xaa 0x55 100 ; standard message with three data bytes
w r 0x100 0 0 0 ; standard rtr message with data length code 3
W 0x100 1 2 ; extended message with two data bytes

H Formatting of data bytes as hexa-decimal numbers
In contrast to h command letter, CAN message data are stored as binary data as canmsg_t structure (see can4linux c-header definition) if local file logging is enabled (-l). All other formats are stored as ASCII character lines.

V Request of the horch version

2.5. Filter

Format: f<filter 1> [.<filter 2> ...]

{id} <id> should pass the filter only
{id1}-<id2> <id1> to <id2>, message range should pass
{id}- beginning from <id> should pass the filter
-<id> from 0 to <id> should pass the filter

{id}, <id1>, <id2> belong to the selected range (inclusive)

Example:

f1,100-200,555,1200-

The current filter settings can be requested by the online command F.
2.6. Trigger

Format: x <idx> <mask> [r] <id> [<data>]

*idx* is a value between 0 and 2 and specifies a trigger buffer. *mask* specifies which bits are 'don’t care bytes' within a CAN message. A bit set means the data byte at this position in the received message and the data byte in the the option data must have the same value. A bit reset (0) means the data byte specified only the length of the trigger message. The value will be ignored by the trigger.

Is the 3rd parameter a r, so the trigger waits for a RTR Message

*id* is the CAN-ID to trigger on,

*data* are optional data bytes of the message that define the length of the message to be triggeres. When no data bytes are given the number is irrelevant.

Example:

```
  x 0 0 0x80,r, y
```

In this example, horch waits for a CAN message with ID 0x80. Number and content of data bytes are not relevant.

2.7. CAN Errors

Errors recognized by the driver are displayed in the console as text messages. The following self-explanatory messages are known:

```
ERROR: OVERRUN       CAN chip overrun
ERROR: PASSIVE       Error passive
ERROR: BUSOFF        Error Busoff (use command R for Bus on)
ERROR: Buffer OVERRUN Software buffer overrun
```
3. CANserver Horch

3.1. Commandline Synopsis

horch [dtSTV] [-D dev][[-a] -b baud][-C -c id][-l file][-p port][-s time]

Command line options

Options  short description
-a       advanced - use option «-b baud» as BTR0/1 value
-b baud  CAN baud rate (in kBit/s, standard 125)
-C       interpret message id given with -c as debug message
-c id    use CAN ID as debug ID
-d       debug mode, useful for program developer
-l file  Logfilename, logging is enabled/disabled with interactive 'l'
-p n     use portnumber n in Servermode, default 7235
-s time  cyclic CAN status display (ms)
-t       show time stamp at start up
-D device CAN device name, (e.g. can1 (LINUX); 0..2 (Level-X))
-S       TCP/IP server mode
-T       use OS time, not drivers time stamp
-V       show program version

additional for LevelX

Options  short description
-B board  board name (see horch -h)
-U number board number (e.g. 1..4 COM port)

3.2. Description

CANserver Horch is a command line application that is capable of receiving and transmitting CAN messages and displaying them in a console window. The output can also be received over a TCP/IP connection in server mode. Because the receiving of CAN messages depends on a special hardware, the application is divided in the two parts, hardware dependent part for CAN receipt and console specific part.

The hardware layer can be a direct interface routine to a CAN controller or an operating system device driver.

The console application part is used to configure the application and shows the received CAN messages in different formats. Display formats are selectable on-line via control characters.

In the Server Mode the CANserver Horch acts as a TCP/IP server waiting for a client that connects to it. It can be seen that stdin and stdout are served by the TCP/IP connection.
in the further command description.

The EtherCAN version of CANserver Horch shows an established client connection by switching Status 1 LED on.

Different implementations of CANserver Horch are available. The single client version CANserver Horch can serve only one TCP/IP client. An extended multi client version of the CANserver Horch is available. It can serve up to ten TCP/IP clients. Application specific extensions are possible. The current version is shown with the

```
$ horch -V
```

A special message ID, which can be changed using -c ID is interpreted as debug message when developing embedded CAN applications. This CAN message contains a continuous character stream which is displayed as text message. CAN devices that have no serial output for debug information can use CAN via the function

```
debug_print_can(unsigned char level, char *fmt, ...);
```

for formatted text output. This C-function works with the CANopen Library by `port` and can be requested if necessary. Text is displayed line by line.

It is possible to filter out interesting messages according to the message ID, others are ignored (see: Filter).

Another feature is the definition of trigger messages. Up to three different messages can be defined. Nothing will be received or shown, until one of these messages are recognized on the CAN bus (see: Trigger).

### 3.3. Options

- **-a** If specified, the baud value given with -b baud is used to set the bit-timing registers directly. The value alignment is driver dependent. *(currently only implemented for can4linux)*.

  ```
  example:
  horch -a -b 0x13c
  ```

- **-b <Bitrate>**

  use baud rate **Bitrate** in kbaud. Without this option the driver is opened with the value from file `/proc/sys/Can/Baud` (LINUX can4linux). Otherwise 125 Kbit/s is the default bit rate.

  ```
  example:
  horch -b 250
  ```

- **-d** Switch debug mode on

  Messages about internal states and program flow are printed to **stderr**.
The CAN message ID given as an argument to the -c option gets a special interpretation if the option -C is set. It's content is interpreted and displayed as an ASCII character stream. CAN applications get the opportunity to send text messages, e.g. debugging messages via CAN. The debug messages are collected until a finishing new-line is received. Then the collected ASCII string is printed, proceeded by the string DEBUG:

example:
horch -C -c 385

The formatted display output can be saved in a local log file. The default name is logfile. With the following option it is possible to change the log-file name. Logging is activated sending the interactive command 'l' to the horch. Logfiles are not created in Server mode.

The internet protocol uses port number <port> to address a specific service on an server host. This is port number 7235 for CANserver Horch. When starting the port number can be set with this option.

Display CAN controller status information every <time> ms.

By default displaying of the time stamp is disabled at start up. It can be enabled interactively. With this option given, it is enabled at start up.

Select the CAN channel to be used (LINUX and Windows drivers). <dev> is the device name. Under Linux this is interpreted as /dev/.dev. Precondition is an installed LINUX CAN device driver. (can4linux, cpc). For CPC interfaces under Windows <dev> is interpreted according the setting in <windir>/cpconf.ini The default entry is CHAN00.
For Level-X Interfaces under Windows the following additional parameters are necessary.

-B <board>
-U <unit>

Example: PCI-IntelliCAN under WinXP:

```
horch_lxbc.exe -BLXN4pi2j -U0 -D0
```

The correct board name depends on the driver type and version used and depends on the operating system it runs on. Possible values can be requested using option -h.

Unit 0 is the first board.
Device 0 is CAN0 on the two channel PCI-IntelliCAN board.

-S  Using CANserver Horch in TCP/IP server mode.
This server is reachable within the local host as localhost, or within a TCP/IP networks with the name of the hosted computer or the IP address and the port number 7235. All interactive commands to CANserver Horch can be given over socket streams.
For the command mode the server can also be reached by the common telnet application.

```
telnet host 7235
```

-T  use operating system time as time stamp.
By default CANserver Horch uses the time stamp provided by the driver at receive time. If the driver does not support time stamps, the operating system time can be used instead. Usually this time is not the receive time, rather the display time.

-V  prints the version number to stdout
4. embedded Horch
Implementing of embedded Horch depends strongly on architecture and development environment and is therefore not part of this manual.

4.1. embedded Horch for UConnect-XE164
*embedded Horch for UConnect-XE164* is a *horch* implementation for the USB UConnect stick by Infineon. Communication with the PC works via a virtual COM interface.
5. Overview Implementations

5.1. Overview CANserver Horch

<table>
<thead>
<tr>
<th>function</th>
<th>can4linux</th>
<th>CPC-Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Long Display Format</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Short Display Format</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CAN Bus Error</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>debug message</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>info message</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>interactive commands</td>
<td></td>
<td></td>
</tr>
<tr>
<td>? (help)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>a (ASCII)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>b (bitrate,kbit/s)</td>
<td>10, 20, 50, 100, 125, 250, 500, 800, 1000</td>
<td>10, 20, 50, 100, 125, 250, 500, 800, 1000</td>
</tr>
<tr>
<td>c (line)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>d (Decimal)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>f (filter)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>F (filter)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>h (hex)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>i (interpreter)</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>l (log)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>m (mask)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>N (long format)</td>
<td>default</td>
<td>default</td>
</tr>
<tr>
<td>o (optional)</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>q (quit)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>R (CAN Reset)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>s (statistic)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>S (short format)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>t (timestamp)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>T (timestamp)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>x (trigger)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>y (trigger)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Y (trigger)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>w (send base frame)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>W (end ext. frame)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>H (hex)</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>function</td>
<td>can4linux</td>
<td>CPC-Windows</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>V (version)</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

command line options

<table>
<thead>
<tr>
<th>option</th>
<th>can4linux</th>
<th>CPC-Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>-a (special bitrates)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>-C (CAN debug)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>-c (CAN Debug)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>-s (status)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>-T (time stamp)</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

x...supported
-.not yet supported
### 5.2. Overview embedded Horch

<table>
<thead>
<tr>
<th>function</th>
<th>UConnect-XE164</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display</td>
<td></td>
</tr>
<tr>
<td>Long Display Format</td>
<td>x</td>
</tr>
<tr>
<td>Short Display Format</td>
<td>x</td>
</tr>
<tr>
<td>CAN Bus Error</td>
<td>x</td>
</tr>
<tr>
<td>debug message</td>
<td>-</td>
</tr>
<tr>
<td>info message</td>
<td>x</td>
</tr>
<tr>
<td>interactive commands</td>
<td></td>
</tr>
<tr>
<td>? (help)</td>
<td>-</td>
</tr>
<tr>
<td>a (ASCII)</td>
<td>-</td>
</tr>
<tr>
<td>b (bitrate,kbit/s)</td>
<td>50, 125, 250, 500, 1000</td>
</tr>
<tr>
<td>c (line)</td>
<td>x</td>
</tr>
<tr>
<td>d (Decimal)</td>
<td>-</td>
</tr>
<tr>
<td>f (filter)</td>
<td>-</td>
</tr>
<tr>
<td>F (filter)</td>
<td>-</td>
</tr>
<tr>
<td>h (hex)</td>
<td>x</td>
</tr>
<tr>
<td>i (interpreter)</td>
<td>not applicable</td>
</tr>
<tr>
<td>l (log)</td>
<td>not applicable</td>
</tr>
<tr>
<td>m (mask)</td>
<td>-</td>
</tr>
<tr>
<td>N (long format)</td>
<td>x</td>
</tr>
<tr>
<td>o (optional)</td>
<td>-</td>
</tr>
<tr>
<td>q (quit)</td>
<td>not applicable</td>
</tr>
<tr>
<td>R (CAN Reset)</td>
<td>x</td>
</tr>
<tr>
<td>s (statistic)</td>
<td>bitrate</td>
</tr>
<tr>
<td>S (short format)</td>
<td>x</td>
</tr>
<tr>
<td>t (timestamp)</td>
<td>x</td>
</tr>
<tr>
<td>T (timestamp)</td>
<td>x</td>
</tr>
<tr>
<td>x (trigger)</td>
<td>-</td>
</tr>
<tr>
<td>y (trigger)</td>
<td>-</td>
</tr>
<tr>
<td>Y (trigger)</td>
<td>-</td>
</tr>
<tr>
<td>w (send base frame)</td>
<td>x</td>
</tr>
<tr>
<td>W (end ext. frame)</td>
<td>x</td>
</tr>
<tr>
<td>H (hex)</td>
<td>not applicable</td>
</tr>
<tr>
<td>V (version)</td>
<td>-</td>
</tr>
</tbody>
</table>

x..supported  
-..not yet supported