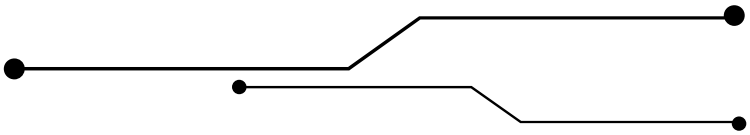
Preliminary User’s Manual

v0.1

hexLIN



USB LIN-BUS Adapter

hexDEV GmbH
Sautterweg 20
70565 Stuttgart
Germany

Tel: +49 711 39082985
E-Mail: info@hexdev.de

Table of Contents

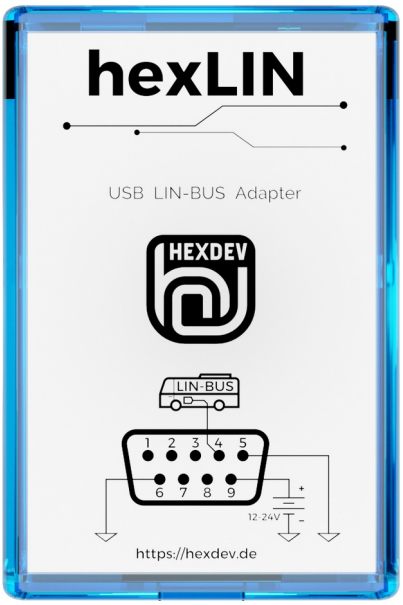
1 Introduction

The hexLIN USB LIN-BUS adapter hardware in its current form is a prototype and work is in progress. The target audience for this documentation (and the adapter hardware itself) is research and development, mainly to test, adapt and discuss a LIN API for mainline Linux kernel.

2 Outreach

hexDEV GmbH is looking out for interesting LIN projects to fund a LIN API for mainline Linux kernel.

3 Exterior View

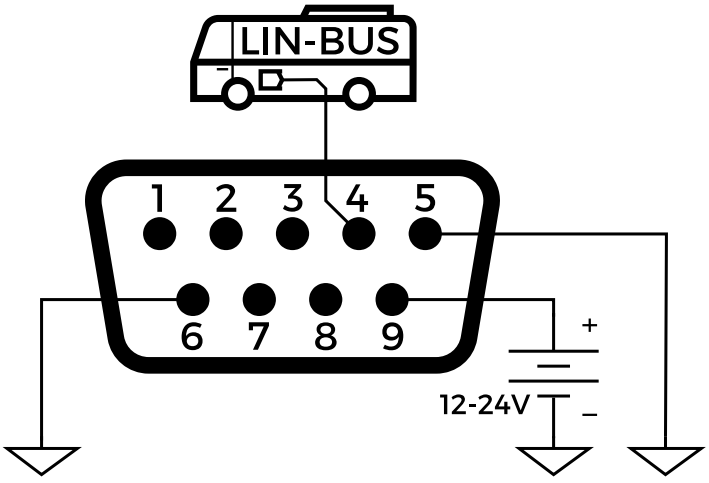


4 Promotion Video

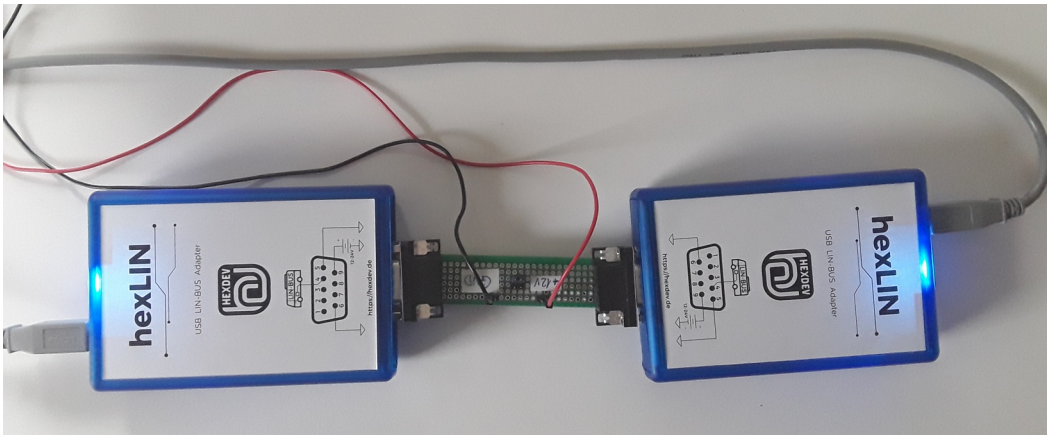
https://hexdev.de/hexlin/hexlin.mp4

5 Hardware LIN Interface

This is how to connect LIN-Bus and Supply (Battery) to the DB9 connector of the USB LIN-BUS Adapter.

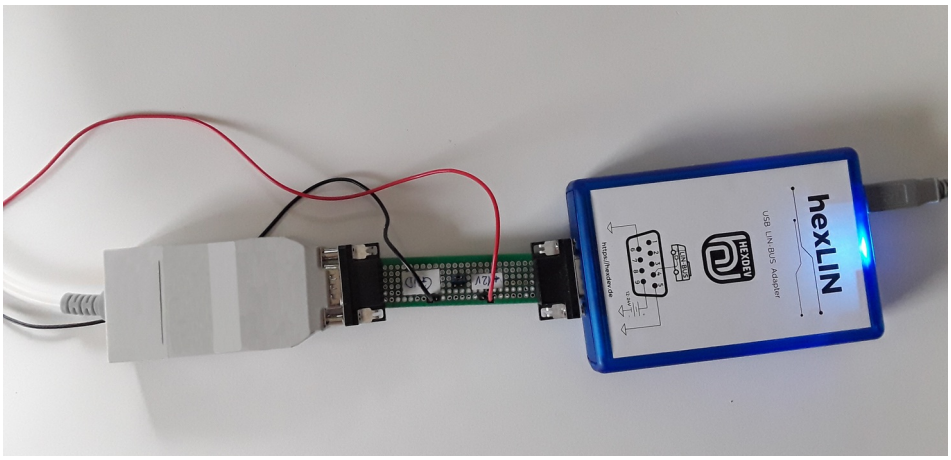


6 Test hexLIN ↔ hexLIN




```
$ hexlin_tests.sh t-bpa

         breakpidanswer: 1       OK
1        breakpidanswer: 1       OK
1        breakpidanswer: 1       OK
1        breakpidanswer: 1       OK
1        breakpidanswer: 1       OK
1        breakpidanswer: 1       OK

$ hexlin_tests.sh t-uncond

         test_two_uncond:
                 tx_device: 0003:16D0:0648.0001
         test_unconditional: 4,0         OK
         rx_device: 0003:16D0:0648.0004
         tx_device: 0003:16D0:0648.0001
         test_unconditional: 16,0        OK
         rx_device: 0003:16D0:0648.0004
         tx_device: 0003:16D0:0648.0001
         test_unconditional: 32,0        OK
         rx_device: 0003:16D0:0648.0004
         tx_device: 0003:16D0:0648.0001
         test_unconditional: 64,0        OK
         rx_device: 0003:16D0:0648.0004
         OK

$ hexlin_tests.sh t-bpe

         test_two_breakpidempty:
                 tx_device: 0003:16D0:0648.0001
         test_breakpidempty: 1   OK
         rx_device: 0003:16D0:0648.0004
                         1
         tx_device: 0003:16D0:0648.0001
         test_breakpidempty: 1   OK
         rx_device: 0003:16D0:0648.0004
                         1
```

7 Test hexLIN ↔ “other LIN hardware”(OLH)



7.1 hexLIN as master, OLH as slave

```
$ olh start slave 9600 /dev/plin0
$ olh set id-filter 0xff:0xff:0xff:0xff:0xff:0xff:0xff:0xff /dev/plin0
$ olh-read /dev/plin0

$ hexlin_tests.sh uncond

  21   2230817510 S 00 f0
  22   2230825510 S 01 f0 f1
  23   2230835518 S 02 f0 f1 f2
  24   2230846516 S 03 f0 f1 f2 f3
  25   2230858526 S 04 f0 f1 f2 f3 f4
  26   2230871526 S 05 f0 f1 f2 f3 f4 f5
  27   2230885531 S 06 f0 f1 f2 f3 f4 f5 f6
  28   2230900530 S 07 f0 f1 f2 f3 f4 f5 f6 f7
  29   2230916535 S 08 f0 f1 f2 f3 f4 f5 f6
  30   2230931536 S 09 f0 f1 f2 f3 f4 f5
  31   2230945540 S 0a f0 f1 f2 f3 f4
```

7.2 hexLIN as slave, OLH as master

```
$ olh start master 9600 /dev/plin0
$ for ((i=0; i<8; i++)); do \
      olh-write -i=0x22 -b="$i 2 1 2" -c=C -d=P /dev/plin0; done

$ hexlin_tests.sh fifo

      no  id  n  data                  cksm flgs
       8  22  4  00 02 01 02 00 00 00 00  fa 4007
       7  22  4  01 02 01 02 00 00 00 00  f9 4007
       6  22  4  02 02 01 02 00 00 00 00  f8 4007
       5  22  4  03 02 01 02 00 00 00 00  f7 4007
       4  22  4  04 02 01 02 00 00 00 00  f6 4007
       3  22  4  05 02 01 02 00 00 00 00  f5 4007
       2  22  4  06 02 01 02 00 00 00 00  f4 4007
       1  22  4  07 02 01 02 00 00 00 00  f3 4007
```

8 Hardware Test Tool

```
/usr/local/bin/hexlin_tests.sh [-v] [test]

   generic tests:
all                 - run all tests below
led                 - blink LEDs
relais              - set master mode and back to slave
dummyrx             - hw sends dummy data by USB to host
slavetable          - set hw slavetable and readback
mdchk               - test setting of mode checksum
baudrate            - set/get baudrate value test

   following tests need a connected LIN-Bus:
uncond              - send and recv uncond. frames by LIN-Bus
bpe                 - send break+id and let empty (timeout) answer
bpa                 - send break+id and let slavetable answer

   following tests need two connected devices on LIN-Bus:
t-uncond            - send and recv uncond. frames by LIN-Bus
t-bpe               - send and recv empty breakpids
t-bpa               - send bp from one device and let other answ

   div functions:
slavetable_active   - active all slaveIDs with some data
slavetable_off      - deactivate all slaveIDs
slave               - show slavetable
fifo                - dump fifo
wakeup_transceiver – trigger wakeup explicitly
```

9 Userland Interface to LIN

LIN uses a new mode of SocketCAN (which is subject to change) from within Linux userland to rx, tx and set baudrate. So e.g. canutils and wireshark can already be used. This chapter shows some example usage scenarios.

9.1 Set bitrate by network interface

```
$ ip link set can0 up type can bitrate 2400
$ ip link set can0 up type can bitrate 4800
$ ip link set can0 up type can bitrate 9600
$ ip link set can0 up type can bitrate 19200
```

9.2 Send data by cansend

From LIN-id 0x01 send 4 bytes: 0xaa 0xbb 0xcc 0xdd:

```
$ cansend can0 101##1aabbccdd
```

From LIN-id 0x02 send 5 bytes: 0xaa 0xbb 0xcc 0xdd 0xee:

```
$ cansend can0 102##1aabbccddee
```

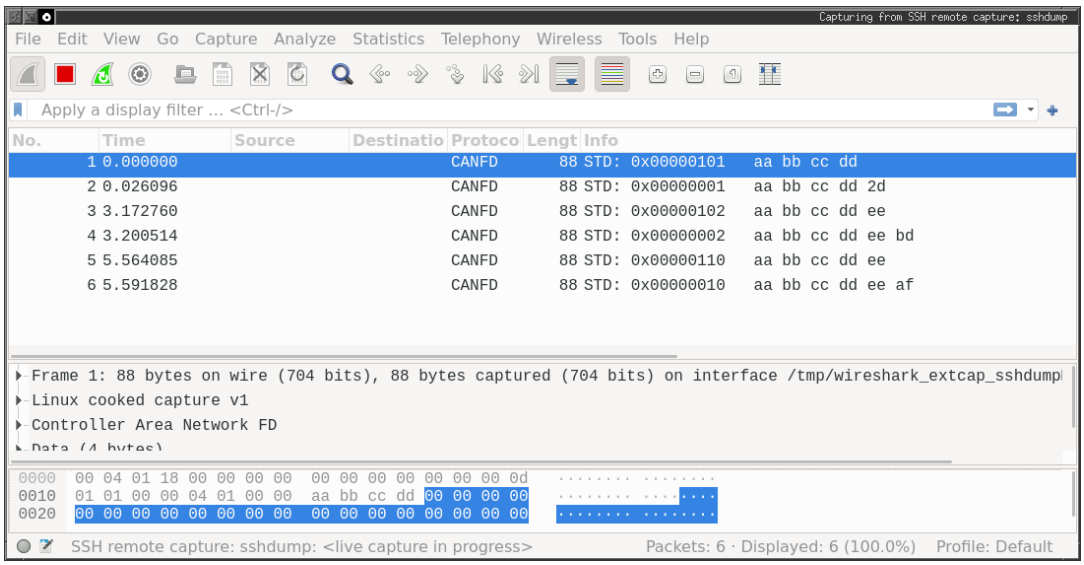
From LIN-id 0x10 send 5 bytes: 0xaa 0xbb 0xcc 0xdd 0xee:

```
$ cansend can0 110##1aabbccddee
```

9.3 Receive LIN frames by candump

```
$ candump can0
 can0  001  [05]  AA BB CC DD 2D
 can0  002  [06]  AA BB CC DD EE BD
 can0  010  [06]  AA BB CC DD EE AF
```

9.4 Trace LIN data by sniffer tool Wireshark



10 LIN Slavetable configuration

Currently the preferred way to configure LIN slavetable is device specific and handled by sysfs interface. This is subject to change and not yet final.

To set LIN answer ID 0x00 to active (“1”) and one databyte to “1” and checksum to “1e”, do this:

```
$ echo "00 1 1 1e" > tx_slaveanswer_pid
```

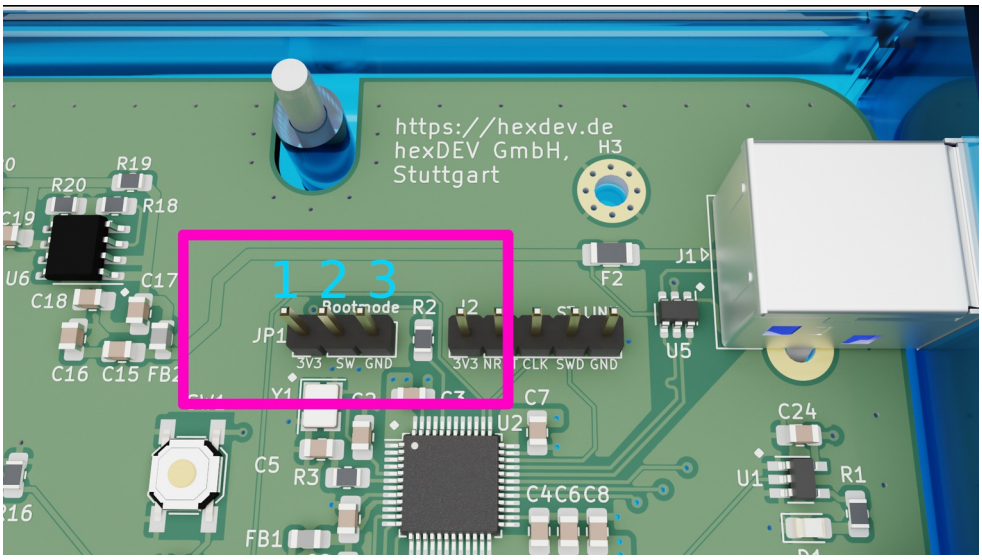
To look at the currently configured slavetable:

```
$ cat  cat tx_slavetable_dump

state   id  n  data                     cksm
active  00  1  1e 00 00 00 00 00 00 00  e1
off     01  0  00 00 00 00 00 00 00 00  00
off     02  0  00 00 00 00 00 00 00 00  00
off     03  0  00 00 00 00 00 00 00 00  00
off     04  0  00 00 00 00 00 00 00 00  00
<snip>
off     3a  0  00 00 00 00 00 00 00 00  00
off     3b  0  00 00 00 00 00 00 00 00  00
off     3c  0  00 00 00 00 00 00 00 00  00
off     3d  0  00 00 00 00 00 00 00 00  00
off     3e  0  00 00 00 00 00 00 00 00  00
off     3f  0  00 00 00 00 00 00 00 00  00
```

See hexlin_tests.sh slavetable for more info.

11 Firmware Update

1. Open case of hexLIN USB LIN-BUS Adapter.
2. Locate jumper JP1 and change default position 2-3 to 1-2.



3. Connect USB cable to host Linux Computer.
4. Get dfu-util by git or from your distribution like so:

```
git clone git://git.code.sf.net/p/dfu-util/dfu-util
./configure --prefix /opt/dfu-util
make -j8
make install
```

4. List connected devices:

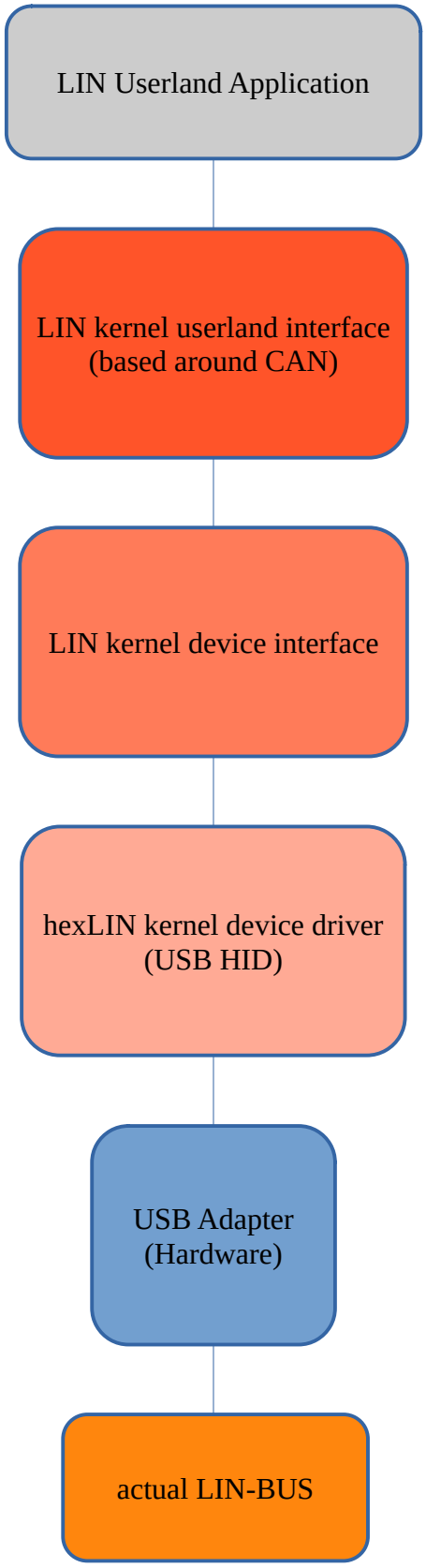
```
./dfu-util -l
```

5. Flash binary hexLIN.bin to device:

```
dfu-util -d 0483:df11 -c 1 -i 0 -a 0 -s 0x08000000 -D \
 ./hexLIN.bin
```

6. reseat jumper jp1 back to position 2-3.

12 Software Layers (current model)



12.1 LIN Userland Application

SocketCAN and sysfs for device specific settings.

12.2 LIN kernel userland interface

special mode SocketCAN

12.3 LIN kernel device interface

mainly:

```
struct lin_device_ops {
     int (*ldo_tx)(struct device *dev, u8 id, u8 n, u8 *data);
     int (*update_bitrate)(struct device *dev, u16 bitrate);
};

int lin_rx(struct lin_device *dev, u8 id, u8 n, u8 *bytes, u8
checksum);
```

12.4 hexLIN kernel device driver

This device specific driver talks to the actual hardware by USB.

This driver has a big sysfs interface to test the hardware independently.

This driver has functions and a kfifio to set and get data from hardware.

12.5 USB-Adapter

It’s a piece of hardware with a USB-Interface connected to a µController and a LIN-Bus Transceiver.

12.6 LIN-BUS

The actual LIN-Bus with multiple slaves and one master node.