# USBtin - USB to CAN interface

USBtin is a simple USB to CAN interface. It can monitor CAN busses and transmit CAN messages. USBtin implements the USB CDC class and creates a virtual comport on the host computer.

- Control with simple serial protocol (LAWICEL / SLCAN compatible)
- Presets for common baud rates (10k ... 1M)
- Listen-only and active mode
- USB powered (no isolation between USB and CAN!)
- 120 Ohm termination via jumper
- Bootloader for firmware updates
- Open source GUI and Java Library

### Contents





## Drivers

For **Linux, MacOS and Windows >8 no additional driver** is needed! USBtin is automatically detected as virtual serial port.

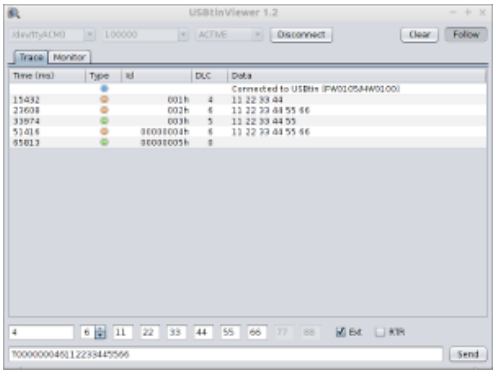
Driver for Windows <8: USBtin_Win_Driver.zip

USBtin is mapped to serial port **"/dev/ACMx"** (Linux, MacOS) or **"COMx"** (Windows). You have to select this corresponding port in the software. On some Linux systems, "modemmanger" blocks the port for about 20s after plug-in (if this disturbs: disable or remove the modemmanager).

## Software

There are several options to control USBtin from the host computer.

### USBtinViewer - Simple multi-platform GUI



USBtinViewer is a **simple GUI for USBtin**. It is open source (GPL), written in Java and runs on several platforms (Windows, Linux, MacOS). Depending on operating system, double-click the downloaded JAR file or run it from command line with "java -jar USBtinViewer_v1.0.jar". Requirement: Java JRE>=1.6

USBtinViewer_v1.2.jar (2015-10-06)

USBtinViewer source code on GitHub

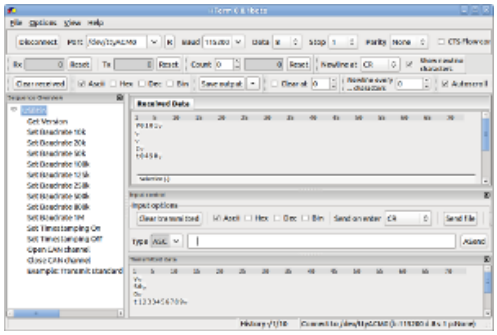
### USBtinLib - Java Library for own developments

```
usbtin.connect("COM3");

usbtin.addMessageListener(new CANMessageListener() {
    @Override
    public void receiveCANMessage(CANMessage canmsg) {
        System.out.println(canmsg);
    }
});

usbtin.openCANChannel(10000, USBtin.OpenMode.ACTIVE);

System.in.read(); // wait for user input

usbtin.send(new CANMessage(0x100, new byte[]{0x11}));

usbtin.closeCANChannel();
usbtin.disconnect();
```

USBtinLib is a **Java Library for accessing USBtin**. It simplifies the use of USBtin in own applications. USBtinLib is open source (LGPL) and runs on several platforms (Windows, Linux, MacOS). The example code (left box) prints out incoming CAN messages and send out one message. Tutorial: Create own GUI with USBtinLib for testing or simulating CAN bus devices.
Requirement: Java JRE>=1.5

- USBtinLib-1.1.0.jar (2015-10-03)
- USBtinLib source code on GitHub
- Demo application source code on GitHub

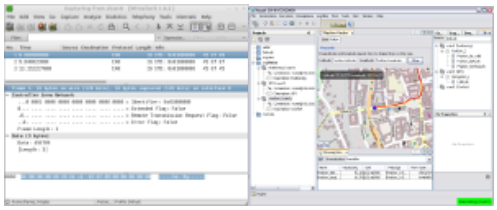
### Serial terminal - Control USBtin with any serial terminal



Commands to USBtin can be sent with **any serial terminal program** such as HTerm which is available as freeware for Windows and Linux. There is a sequence file for HTerm with the main USBtin commands predefined. To load it in HTerm, right click into "Sequence Overview" -> "Load File...".

- USBtin_HTerm_Sequences.hts
- HTerm - Terminal program

### Linux-can (SocketCAN) - Use linux CAN functions



USBtin also works with **linux-can (SocketCAN)**. Here is short tutorial about this:
USBtin and linux-can (SocketCAN)

Tools based on linux-can can be used with USBtin. For example:

- Kayak
- Wireshark

## Firmware

### Source code and HEX file

The firmware of USBtin is available as precompiled HEX file and as C (XC8 compiler) source code:

- usbtin.2014-12-05.tar.gz Version v1.5 - New buffer structure with USB ping-pong and CDC directly to USB ram.
- usbtin.2014-09-20.tar.gz Version v1.4 - Improved performance: Increased bulk transfer packet size (patch by Jürgen Liegner)
- usbtin.2014-04-07.tar.gz Version v1.3 - Based on improvement suggestions by Mikael Gustavsson (MG Digital Solutions)
- usbtin.2013-01-11.tar.gz Version v1.2 - Added filter function 'm' and 'M' and register write 'W'
- usbtin.2012-03-09.tar.gz Version v1.1 - First public version

### Firmware update via bootloader

Set the bootloader jumper (JP1) and plug in USBtin. Now the bootloader starts. Use bootloader application (e.g. MPHidFlash) to load the new firmware into the flash of USBtin:

```
mphidflash -w USBtin_firmware_v1.x.hex
```

### ASCII commands

USBtin registers as a virtual serial port on the host computer. With simple ASCII commands USBtin can be controlled over this serial port. You can send/receive commands from any serial terminal program or from your own program. If you don't like to deal with the plain commands, there is also a simple GUI: USBtinViewer.

#### Example

Set 10 kBaud, open CAN channel, send CAN message (id=001h, dlc=4, data=11 22 33 44), close CAN:

```
Command:               Response:
S0[CR]                 [CR]
O[CR]                  [CR]
t001411223344[CR]      z[CR]
C[CR]                  [CR]
```

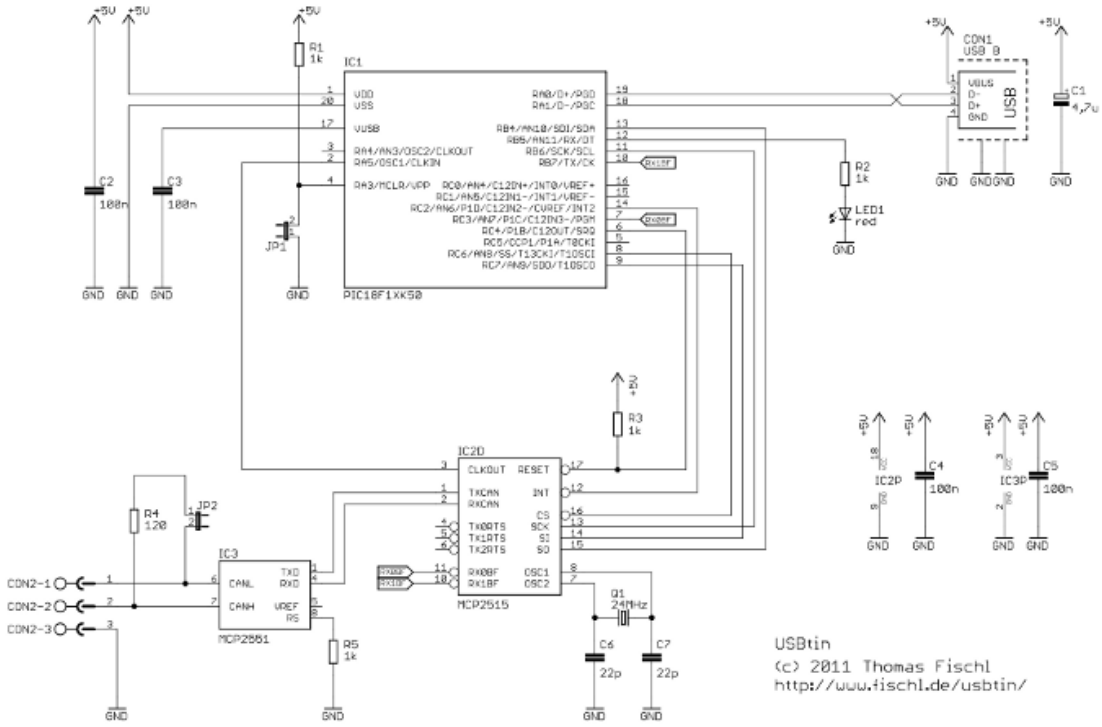
**Command list**

The commands are line based and terminated with newline character CR (0xD). On error the USBtin responses with 0x7.

| | |
|---|---|
| Sx[CR] | Set baudrate<br>x: Bitrate id (0-8)<br>S0 = 10 kBaud<br>S1 = 20 kBaud<br>S2 = 50 kBaud<br>S3 = 100 kBaud<br>S4 = 125 kBaud<br>S5 = 250 kBaud<br>S6 = 500 kBaud<br>S7 = 800 kBaud<br>S8 = 1 MBaud |
| sxxyyzz[CR] | Set can bitrate registers of MCP2515. You can set non-standard baudrates which are not supported by the "Sx" command.<br>xx: CNF1 as hexadecimal value (00-FF)<br>yy: CNF2 as hexadecimal value (00-FF)<br>zz: CNF3 as hexadecimal value |
| Gxx[CR] | Read MCP2515 register. xx: Address of MCP2515 register to read as hexadecimal value (00-FF). |
| Wxxyy[CR] | Write MCP2515 register. xx: Address of MCP2515 register to write. Hexadecimal value (00-FF).<br>yy: Data to write to the register. Hexadecimal value (00-FF). |
| V[CR] | Get hardware version. |
| v[CR] | Get firmware version. |
| N[CR] | Get serial number. Returns always 0xffff. |
| O[CR] | Open CAN channel. |
| l[CR] | Open device in loop back mode. |
| L[CR] | Open CAN channel in listen-only mode. |
| C[CR] | Close CAN channel |
| tiiildd..[CR] | Transmit standard (11 bit) frame.<br>iii: Identifier in hexadecimal format (000-7FF)<br>l: Data length (0-8)<br>dd: Data byte value in hexadecimal format (00-FF) |
| Tiiiiiiiildd..[CR] | Transmit extended (29 bit) frame.<br>iiiiiiii: Identifier in hexadecimal format (0000000-1FFFFFFF)<br>l: Data length (0-8)<br>dd: Data byte value in hexadecimal format (00-FF) |
| riiil[CR] | Transmit standard RTR (11 bit) frame.<br>iii: Identifier in hexadecimal format (000-7FF)<br>l: Data length (0-8) |
| Riiiiiiiil[CR] | Transmit extended RTR (29 bit) frame.<br>iiiiiiii: Identifier in hexadecimal format (0000000-1FFFFFFF)<br>l: Data length (0-8) |
| F[CR] | Read status/error flag of can controller<br>Return: Fxx[CR] with xx as hexadecimal byte with following error flags:<br>Bit 0 - not used<br>Bit 1 - not used<br>Bit 2 - Error warning (Bit EWARN of MCP2515)<br>Bit 3 - Data overrun (Bit RX1OVR or RX0OVR of MCP2515)<br>Bit 4 - not used<br>Bit 5 - Error-Passive (Bit TXEP or RXEP of MCP2515)<br>Bit 6 - not used<br>Bit 7 - Bus error (Bit TXBO of MCP2515) |
| Zx[CR] | Set timestamping on/off.<br>x: 0=off, 1=on |
| mxxxxxxxx[CR] | Set accpetance filter mask. SJA1000 format (AM0..AM3). Only first 11bit are relevant.<br>xxxxxxxx: Acceptance filter mask |
| Mxxxxxxxx[CR] | Set accpetance filter code. SJA1000 format (AC0..AC3). Only first 11bit are relevant.<br>xxxxxxxx: Acceptance filter code |

# Hardware

## Schematic - USBtin's circuit diagram

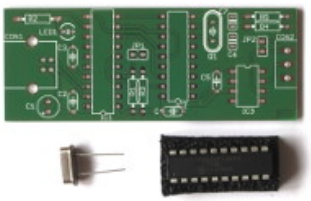



## Partlist - Build your own USBtin!

Here you find a list of all parts needed to build your own USBtin board. The red marked parts are also available as USBtin-Set (PCB + programmed uC + crystal) - see photo in right box.

| Partnumber | Value | Reichelt | Conrad |
|---|---|---|---|
| C1 | 4,7u | RAD 4,7/35 | 445587 - 62 |
| C2, C3, C4, C5 | 100n | X7R-5 100N | 531855 - 62 |
| C6, C7 | 22p | KERKO 22P | 457167 - 62 |
| CON1 | USB-B | USB BW | 738884 - 62 |
| CON2 | 3pol | AKL 182-03 | 729906 - 62 |
| IC1 | PIC18F14K50-I/P | PIC18F14K50-IP (not programmed!) | 651666 - 62 (not programmed!) |
| IC2 | MCP2515-I/P | MCP 2515-I/P | 651447 - 62 |
| IC3 | MCP2551-I/P | MCP 2551-I/P | 651451 - 62 |
| JP1, JP2 | 2pol 2,54mm | SL 1X36G 2,54 | 742235 - 62 |
| LED1 | 3mm LED red | LED 3MM 2MA RT | 184560 - 62 |
| Q1 | 24Mhz !fundamental! | - | - |
| R1, R2, R3, R5 | 1k | 1/4W 1,0K | 405256 - 62 |
| R4 | 120 | 1/4W 120 | 405140 - 62 |
| M1 | 3pol | AKL 169-03 | 729841 - 62 |
| M2 | Jumper | JUMPER 2,54GL SW | 737557 - 62 |
| M3 | Socket 20 | GS 20 | 189545 - 62 |
| M4 | Socket 18 | GS 18 | 189537 - 62 |
| M5 | Socket 8 | GS 8 | 189502 - 62 |
| M6 | Case | TEKO 10007 | 543335 - 62 |
| M7 | PCB | - | - |

**PCB + programmed uC + crystal**



€ 14,90

incl. 19% VAT excl. € 4 shipping

**Order USBtin-Set now!**

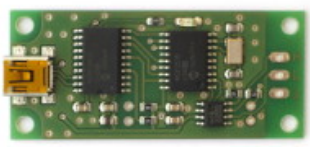
**Buy on Amazon.de!**

(€ 17,90 incl. 19% VAT excl. shipping)

Shipping to all EU countries!

**SMD board, programmed + tested**

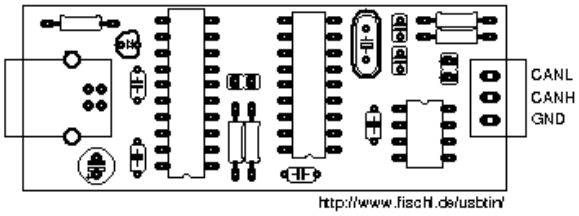
USBtin EB Datasheet



€ 34,90

incl. 19% VAT excl. € 4 shipping

**Order USBtin EB now!**






## Troubleshooting

### Linux/Ubuntu: "Permission denied" when accessing /dev/ACM0

/dev/ttyACMx belongs to dialout group, so add user to this group (re-login or reboot to take effect!):

```
$sudo adduser $USER dialout
```

### Linux/Ubuntu/RaspberryPi: USBtin is blocked or misconfigured after plugging in or bootup

The program "network-manager" probes the serial port (sends strings for modem detection). Add rule so network-manager ignores USBtin:

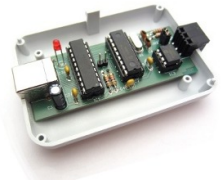
```
sudo gedit /etc/udev/rules.d/10-local.rules
```

Add this line:

```
ATTRS{idVendor}=="04d8", ATTRS{idProduct}=="000a", ENV{ID_MM_DEVICE_IGNORE}="1"
```

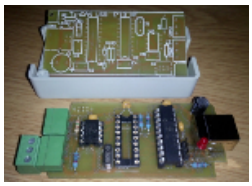
## Gallery

If you have built your own version of USBtin, please let me know!



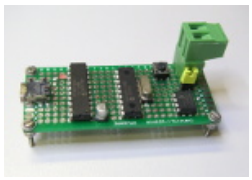
by Thomas Fischl (04/2012)

USBtin-Set with PCB, programmed uC and crystal available:
Order USBtin-Set. Other parts see partlist.



by Nico Ferreira (02/2014)

PCB is homemade with laser.
Plastic box is a Hammond 1551K.



by John (05/2014)

Wired on breadboard.

by Васíлης Σπυρόπουλος (07/2014)

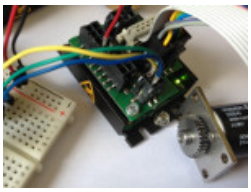
Mounted in case Teko 10014.9.
With OBD-2 adapter cable.

by Александр Сафонов (09/2014)

by John Gerrard (12/2014)

USBtin EB and Wago CANopen Fieldbus Coupler 750-307 test setup
by Georg G. (01/2015)

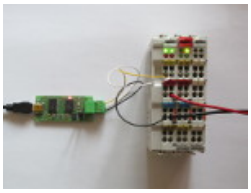
USBtin EB connected to Android device with USB-OTG (Host) function.
USB Serial Terminal Lite is used to communicate with USBtin.
(04/2015)



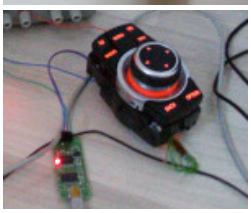
USBtin connected to Maxon EPOS CANopen motor controller. USBtin is used for Sniffing at CANopen.

Video: Use USBtin to run CANopen devices from a Pyton script

by Bas de Bruijn (05/2015)



USBtin EB connected to Beckhoff LC5100 CANopen bus coupler.
For further details see Control CANopen devices with USBtin.
(07/2015)



USBtin EB communicates with BMW iDrive controller.

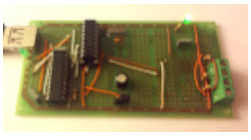
by Manuel R. (08/2015)

USBtin EB is used for parametrization and monitoring of racing car components.

by UPBracing Team e.V. (11/2015)

USBtin EB (installed in blue box) is used to connect SCHUNK FTM115 force torque module with KUKA KRC3 industrial robot.

by Dennis B. (12/2015)



USBtin wired on breadboard. Uses TJA1054 instead of MCP2551. Connected to CAN bus of Jeep Grand Cherokee.

by Mariusz K. (12/2015)

## Links

USBtin and linux-can (SocketCAN)
embedded projects Journal - Article about USBtin in Journal #13

### Other can interface projects

CANUSB adapter by LAWICEL

Tiny-CAN interface by MHS Elektronik
CAN interface with R8C/23

---