APPLICATION NOTE

AN80101

Seal USB/CAN Adapter Quick Start

Document revision: Release v1.2
Issue date: Feb 27, 2018

Contents

1 Introduction

The *Seal* USB/CAN adapter is a compact and easy-to-use device that allows you to add a CAN interface to any computer running Linux. It uses the UCAN wire protocol on the USB interface.

Note: The latest version of this document is always available at https://www.theobroma-systems.com/seal .

1.1 Pinout

The signal assignment of the three wire clamps and the location of the DFU button are shown in Fig. 1.1.

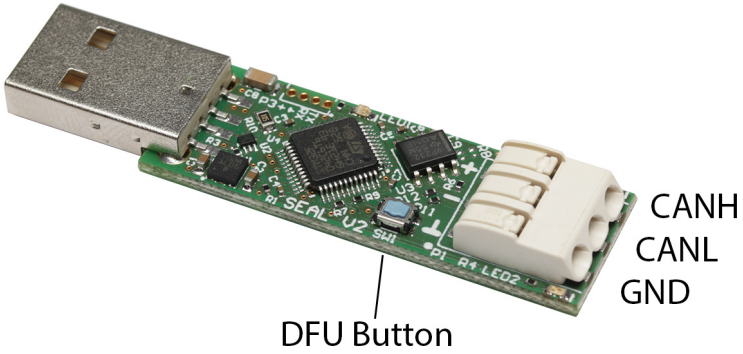

Fig. 1.1: Pinout & button location

1.2 Wire clamp operation

Seal uses spring-loaded wire clamps. Supported conductors are:

Wire type	Solid or stranded
Cross-section	0.2 - 0.75 mm^2 24 - 18 AWG
Strip length	7 - 9 mm 0.28 - 0.35 inch

To insert the cable, push down the spring lever with a cross-head screwdriver or with a small flat-head screwdriver (Fig. 1.2). Releasing the spring lever secures the cable.

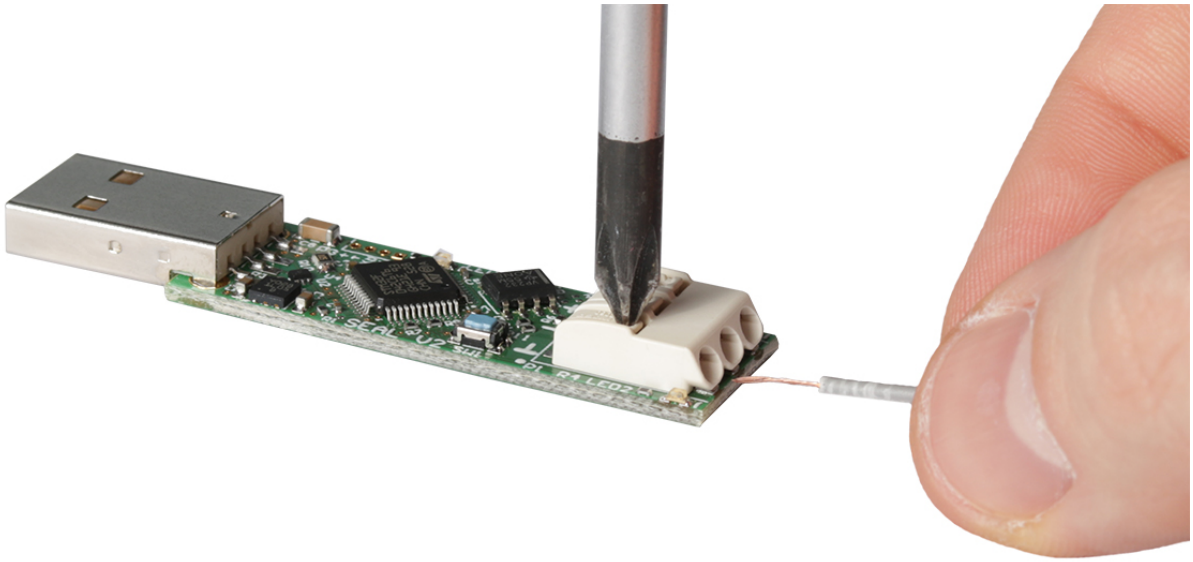

Fig. 1.2: Cable insertion

1.3 Linux driver

The driver is provided as an open-source loadable kernel module named UCAN. Clone the git repository:

```
git clone https://git.theobroma-systems.com/ucan-linux-module.git
```

Follow the `README.md` file that is included in the git repository and available on the web at:

https://git.theobroma-systems.com/ucan-linux-module.git/tree/README.md .

1.4 Using the CAN interface

Once you have the UCAN kernel module loaded, connect the *Seal* device to USB. You should get kernel output similar to this:

```
usb 1-2.2: new full-speed USB device number 18 using uhci_hcd
usb 1-2.2: New USB device found, idVendor=2294, idProduct=425a
usb 1-2.2: New USB device strings: Mfr=1, Product=2, SerialNumber=3
usb 1-2.2: Product: STM32 CAN Full Speed
usb 1-2.2: Manufacturer: Theobroma Systems
usb 1-2.2: SerialNumber: 2057394E5843
usb 1-2.2: Found UCAN device on interface #0
usb 1-2.2: Device Reports:
  Frequency [Hz]           : 48000000
  TX Fifo [length]         : 3
  Time Segment 1 [min,max] : 1 - 16
  Time Segment 2 [min,max] : 1 - 8
  SWJ [max]                : 4
  Prescale [min-max,step]  : 1 - 1024, 1
  Supported modes          : Loopback Silent OneShot BusErrReport [1b]
```

Also, a `can0` interface should show up on the `ip addr` output:

```
ip addr
...
5: can0: <NOARP> mtu 16 qdisc noop state DOWN group default qlen 10
link/can
```

Note: If there are multiple CAN interfaces available they are numbered incrementally (e.g. `can0`, `can1`, `can2`, …).

Bring the interface up using:

```
sudo ip link set can0 up type can bitrate 125000
```

Note: Bitrates up to 1Mbit/s are supported.

To monitor running CAN traffic use:

```
candump -i can0
```

To send a single CAN frame to a random ID use:

```
cangen -n 1 can0
```

1.5 Monitoring CAN link status

Details about the health of the CAN link can be shown using:

```
ip -details -statistics link show can0
```

Example output is shown below:

```
# ip -details -statistics link show can0
4: can0: <NOARP,UP,LOWER_UP> mtu 16 qdisc pfifo_fast state UNKNOWN mode DEFAULT group default qlen 10
    link/can  promiscuity 0
    can <BERR-REPORTING> state ERROR-ACTIVE restart-ms 0
          bitrate 125000 sample-point 0.875
          tq 500 prop-seg 6 phase-seg1 7 phase-seg2 2 sjw 1
          ucan: tseg1 1..16 tseg2 1..8 sjw 1..4 brp 1..1024 brp-inc 1
          clock 48000000
          re-started bus-errors arbit-lost error-warn error-pass bus-off
          0          0          0          2          2          0         numtxqueues 1 numrxqueues 1
    RX: bytes  packets  errors  dropped overrun mcast
    68         36       16      0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
    1161425    202035   39      0       0       0
```

The most important fields are explained below:

- `state ERROR-ACTIVE`: The CAN interface is currently in the ERROR-ACTIVE state. Possible states are:
  - STOPPED (interface not yet enabled by the user)
  - ERROR-ACTIVE (normal operation)

  - ERROR-WARNING (normal operation but elevated error rates are seen on the bus)
  - ERROR-PASSIVE (normal operation with reduced error jamming on the bus)
  - BUS-OFF (disconnected from the bus due to too many errors)
- `bitrate 125000 sample-point 0.875`: the CAN interface runs at 125000 bps and data is sampled at 87.5% of the bit time
- `error-warn 2`: number of times the CAN interface has moved from state ERROR-ACTIVE to ERROR-WARN
- `error-pass 2`: number of times the CAN interface has moved from state ERROR-WARN to ERROR-PASSIVE
- `bus-off 0`: number of times the CAN interface has moved from state ERROR-PASSIVE to BUS-OFF
- `RX packets 36`: number of good CAN data frames that have been received
- `RX errors 16`: number of corrupted CAN data frames have been received
- `TX packets 202035`: number of CAN data frames have been sent
- `TX errors 39`: number transmission errors that have been detected (like a missing ACK)

1.6 Serial number

Every *Seal* device is equipped with a unique 96-bit serial number. If you have multiple *Seal* devices connected, this serial number can be used to uniquely identify each device.

To get the serial number of the *Seal* device that represent `can0` run:

```
# udevadm info /sys/class/net/can0 | grep ID_SERIAL_SHORT
E: ID_SERIAL_SHORT=003B00415750511920353631
```

To read the serial numbers of all connected *Seal* devices (two in this example):

```
# lsusb -d 2294:425b -v | grep iSerial
  iSerial                 3 0034001A5843431720383934
  iSerial                 3 0034001B5843431720383934
```

1.7 Firmware upgrade

Upgrading the firmware on *Seal* is a two-step process:

1. Start *Seal* in DFU firmware upgrade mode
2. Load the new firmware using `dfu-util`

The steps are described below.

To start *Seal* in DFU firmware upgrade mode, press and hold the DFU button (see figure Fig. 1.1), then connect USB. Release the button.

Seal will show up as an `STM Device in DFU Mode` as shown below:

```
lsusb
...
Bus 001 Device 020: ID 0483:df11 STMicroelectronics STM Device in DFU Mode
```

You can then load the new firmware using:

```
dfu-util -a 0 -D seal.dfu
```

The output should look like this:

```
dfu-util 0.9

Copyright 2005-2009 Weston Schmidt, Harald Welte and OpenMoko Inc.
Copyright 2010-2016 Tormod Volden and Stefan Schmidt
This program is Free Software and has ABSOLUTELY NO WARRANTY
Please report bugs to http://sourceforge.net/p/dfu-util/tickets/

Match vendor ID from file: 0483
Match product ID from file: df11
Opening DFU capable USB device...
ID 0483:df11
Run-time device DFU version 011a
Claiming USB DFU Interface...
Setting Alternate Setting #0 ...
Determining device status: state = dfuDNLOAD-IDLE, status = 0
aborting previous incomplete transfer
Determining device status: state = dfuIDLE, status = 0
dfuIDLE, continuing
DFU mode device DFU version 011a
Device returned transfer size 2048
DfuSe interface name: "Internal Flash  "
file contains 1 DFU images
parsing DFU image 1
image for alternate setting 0, (1 elements, total size = 14512)
```

```
parsing element 1, address = 0x08000000, size = 14504
Download        [=========================] 100%        14504 bytes
Download done.
done parsing DfuSe file
```

Unplug and replug *Seal* to start the new firmware.

1.8 Further reading

Controller area network (CAN) ISO standard	https://www.iso.org/standard/63648.html
Linux SocketCAN documentation	https://www.kernel.org/doc/html/latest/networking/can.html