Korlan USB2CAN User Guide

High quality isolated USB to CAN interface

07-03-2023

Table of contents

1. Introduction

Korlan USB2CAN provides a convenient computer (USB) to CAN 2.0 network interface.
You can send and receive CAN messages. Write your program for communicating with industrial, medical, and automotive devices equipped with CAN interface. Korlan USB2CAN comes in two different versions: DB9 or OBD2. The Korlan CANBUS USB adapter connects a CAN bus to the USB port of any device running the Linux or Windows operating system, which also supplies the power to the adapter (no power supply is required). In the downloads section, you can find all the related drivers, software applications, and programming examples for Windows 10/11 and Linux operating systems.

1.1 Usage warning

Use your device with caution and a complete understanding of the risks. This warning informs you that the operation of this device may be dangerous. Your actions can influence the behavior of a CAN-based distributed embedded system, and depending on the application, the consequences of your improper actions could cause serious operation malfunction, loss of information, damage to equipment, and physical injury to yourself or others.

1.2 Technical specification

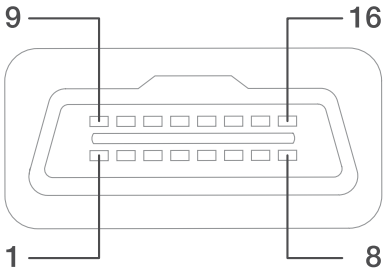
CONNECTORS	
Computer	USB 2.0 Full speed, Type A connector
CAN	D-SUB, 9 pins. CAN-CIA standard interface pin assignment or OBD2, 16 pins. Standard interface pin assignment.
CAN	
Specification	ISO 11898-2 High-speed CAN 2.0A (standard format) and 2.0B (extended format)
Bit rates	50, 50, 100, 125, 250, 500, 800, 1000 Kbit/s or user definable
Controller	ARM 32bit Cortex-M0 (STM32F072)
Transceiver	TI ISO1050
Galvanic isolation	Up to 2.5kV, separate for each CAN channel
Termination	None
MEASURES	
Size	110 x 36.7 x 16.2 mm (L x W x H), no cable
Weight	DB9 version: 100g with cable, OBD2 version: 140g with cable
ENVIRONMENT	
Operating temperature	From -35°C to +55°C
Relative humidity	15-90%, not condensing
Usage	Indoor only
OTHER	
Available drivers	Windows 2000, Windows XP, Windows Vista, Windows 7, Windows 10, Linux
3rd party ptotocol support	Driver for VSCP protocol

CONNECTORS	
	Open source CANAL API DLL for Windows
	Linux SocketCAN compatible

2. Korlan USB2CAN converter

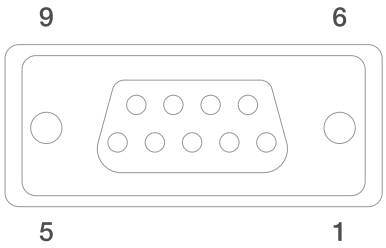
2.1 CAN connector pinout

FIGURE 2-1. DB9 CONNECTOR



PIN	SIGNAL	DESCRIPTION
1	-	No connection
2	CANL	CANL bus line (dominant low)
3	CAN GND	CAN Ground
4	-	No connection
5	CAN_SHLD	Connected to CAN GND via 100 Ω /0.1uF
6	CAN GND	CAN Ground
7	CANH	CANH bus line (dominant high)
8	-	No connection
9	-	No connection

FIGURE 2-2. ODB2 CONNECTOR



PIN	DESCRIPTION
4	GND
5	GND
6	CAN Bus High
14	CAN Bus LOW

Note: All other pins on OBD2 are not used and not connected.

2.2 LED indication

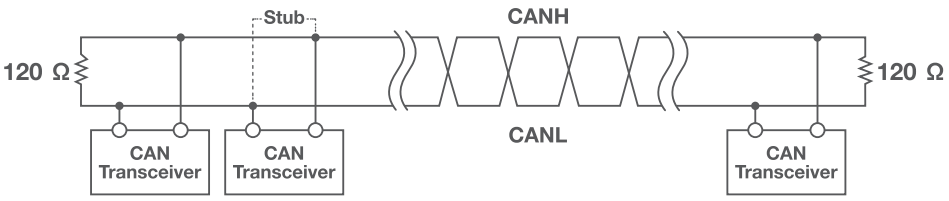
Korlan USB2CAN has three LEDs (Power, Error, Info) for device status indication.
Device status Indication modes are listed in the table below.

1. **INFO** – LED (Info)
2. **STAT** – LED (Status)
3. **PWR** – LED (Power)

LED	STATUS DESCRIPTION
PWR LED on	Device is powered on.
INFO LED on, STAT LED off	CAN interface enabled, device is ready to receive or send data.
STAT LED on, INFO LED off	CAN interface is not enabled on a host device.
INFO LED on, STAT LED blinking	CAN interface sending data.
INFO LED blinking fast	CAN bus passive error.
INFO LED blinking slow	CAN bus warning.
STAT LED blinking slow	CAN bus-off error.

2.3 CAN bus termination

A High-speed CAN bus (ISO 11898-2) must be terminated on both ends with 120 Ohm resistors.



Standard termination example.
The 120-Ω characteristic impedance twisted-pair cable must be terminated with an impedance of the same value to minimized reflected waves that occur from miss-matched impedances. Bad terminal may cause signal reflections and the transceivers of the connected CAN nodes (CAN- interface, control device) may not work.

The USB2CAN does not have an internal termination. Device must be used on a terminated CAN bus.

2.4 Testing modes

Korlan USB2CAN supports three testing modes:

- **Loopback** - mode is used to test if the device is applicable when only one device is available. If Korlan USB2CAN is in loopback mode, everything sent through the CAN interface is sent back to the device. The CAN interface TX pins are connected to its RX pins.
- **Silent** - in this mode, the interface only listens CAN bus. No data or ACK frames are sent.
- **Silent loopback** - is a self-test mode that can be used in a working CAN system without interfering with other devices connected to the CAN bus.

3. Work on Linux

3.1 Driver and configuration

Mainline Linux has supported the Korlan USB2CAN converter since version 3.9. It works with Linux distributions like Ubuntu and Debian with no additional driver installation. For Raspbian, the driver must be built manually. Instruction on building CAN driver for Raspbian can be found on 8devices WIKI pages. https://www.8devices.com/wiki/korlan:compile-raspberry

Recommended Linux distributions:

- Ubuntu 18.04 or later
- Debian 8 (Jessie) or later
- Raspbian Bullseye

You can verify if Korlan USB2CAN device is recognized by the system and drivers loaded correctly using the usb-devices command. In the output, you will see your device:

```
usb-devices
T:  Bus=03 Lev=02 Prnt=02 Port=02 Cnt=03 Dev#=  7 Spd=12  MxCh= 0
D:  Ver= 2.00 Cls=ff(vend.) Sub=ff Prot=00 MxPS=64 #Cfgs=  1
P:  Vendor=0483 ProdID=1234 Rev=01.00
S:  Manufacturer=8Devices
S:  Product=USB2CAN converter
S:  SerialNumber=9508BBCC
C:  #Ifs= 1 Cfg#= 1 Atr=80 MxPwr=100mA
I:  If#=0x0 Alt= 0 #EPs= 4 Cls=ff(vend.) Sub=ff Prot=ff Driver=usb_8dev
```

If you use several Korlan USB2CAN devices, you will see all of them in USB-devices output with different SeriaNumber values. The first device in the output will correspond to the can0 socketcan interface, the second - can1, etc.
If the driver is loaded correctly, the CAN interface should be seen by issuing the command:

```
ip link show
4: can0: <NOARP,ECHO> mtu 16 qdisc noop state DOWN mode DEFAULT group default qlen 10
link/can
```

Set can0 interface speed to 125 Kbps:

```
sudo ip link set can0 up type can bitrate 125000 sample-point 0.875
```

Bring the can0 interface to UP state (INFO LED lights on, STAT LED lights off):

```
sudo ip link set can0 up
```

To bring down the interface (STAT LED lights up, INFO LED lights off):

```
sudo ip link set can0 down
```

You can set different options when configuring can interface.For example:

To configure can0 interface to work in loopback mode:

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

To configure can0 interface to work in silent mode:

```
sudo ip link set can0 up type can bitrate 125000 listen-only on
```

To disable auto retry on can0 interface and ensure that if an error occurs, Korlan does not try to resend a TX frame:

```
sudo ip link set can0 up type can bitrate 125000 one-shot on
```

If you want to get more information about configuration options type:

```
sudo ip link set can0 type can help
```

3.2 Tools - can-utils (send/receive CAN packets)

The programs allow you to get CAN communications instantly using two commands, `cansend`, and `candump`.

Command to install the tool:

```
sudo apt-get install can-utils
```

Send byte of information (1122334455667788) to can device with id = “1f334455”:

```
cansend can0 1f334455#1122334455667788
```

Receive everything on the can1 interface:

```
candump can1
```

3.3 Python programming examples for Linux

Python programming examples rely on the python-can module and socket can drivers. Before running code examples, you shall bring the CAN interface to the UP state using the command:

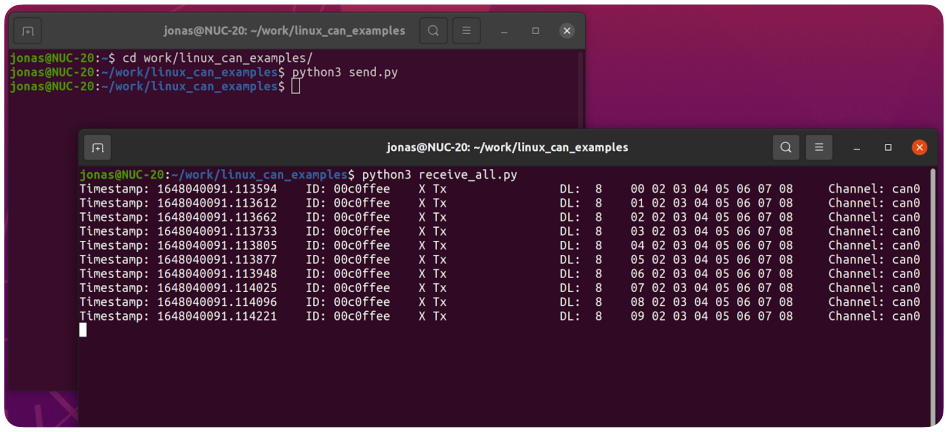
```
sudo ip link set can0 up type can bitrate 125000 sample-point 0.875
```

Example receive_all.py - receives and prints all messages on the screen until Ctr-C is pressed.

```
import can
with can.interface.Bus(channel='can0', interface='socketcan', bitrate=1000000) as bus:
    try:
        while True:
            msg = bus.recv(1)
            if msg is not None:
                print(msg)
    except KeyboardInterrupt:
        pass  # exit normally
```

Example send.py - sends 10 CAN messages to the bus.

```
import can
bus = can.Bus(channel='can0', interface='socketcan', bitrate=1000000)
for i in range(10):
   msg = can.Message(arbitration_id=0xc0ffee, data=[i,2,3,4,5,6,7,8],
is_extended_id=True)
   bus.send(msg)
```



3.4 Firmware upgrade

Install dfu-util tool

```
sudo apt-get install dfu-util
```

Switch to the internal bootloader using the set_dfu utility provided with the firmware:

```
sudo ./set_dfu
```

Flash new firmware as root:

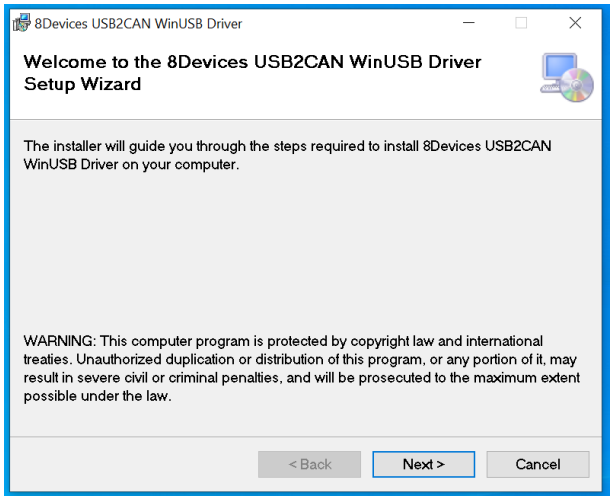
```
sudo dfu-util -a 0 -s 0x8008000 -D USB2CAN-v2.0.bin
```

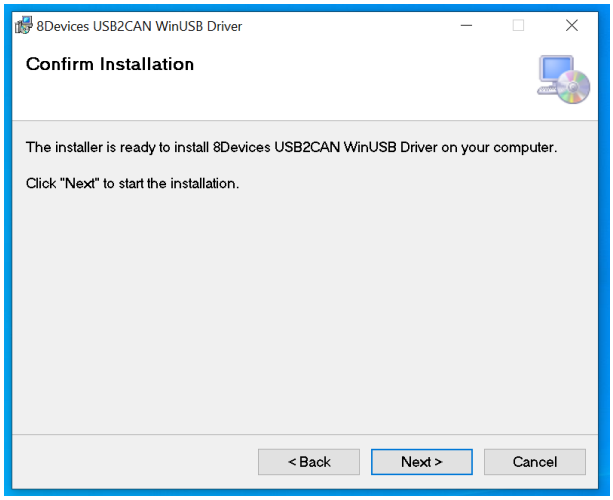
4. Work on Windows

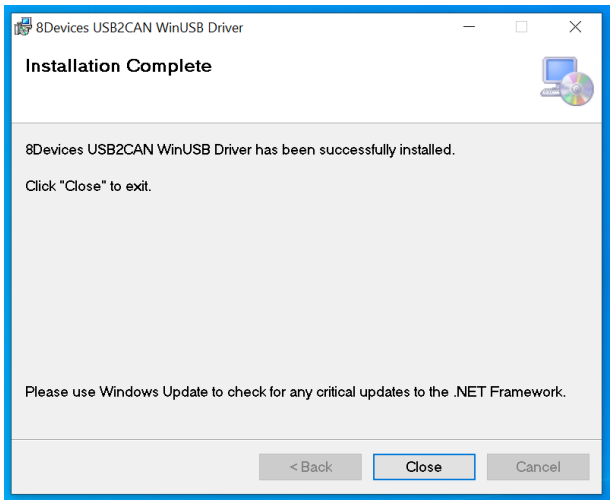
4.1 Installing Windows 10/11 WinUSB driver

Download the Korlan USB2CAN driver from
https://www.8devices.com/media/products/usb2can_korlan/downloads/usb2can_winusb.msi.

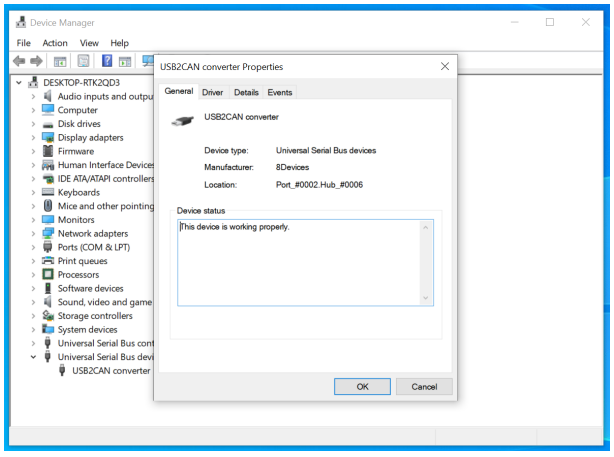
Install it following the procedure.

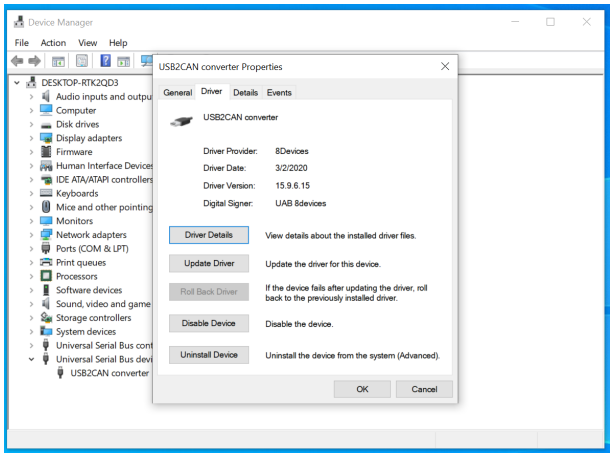






Check if the device is functioning correctly using Windows Device Manager





4.3 Python programing examples for Windows

You will require a CANAL DLL library from the 8devices WEB site. It's been provided for 64 and 32-bit environments
https://www.8devices.com/media/products/usb2can_korlan/downloads/korlan_python-can_examples.zip

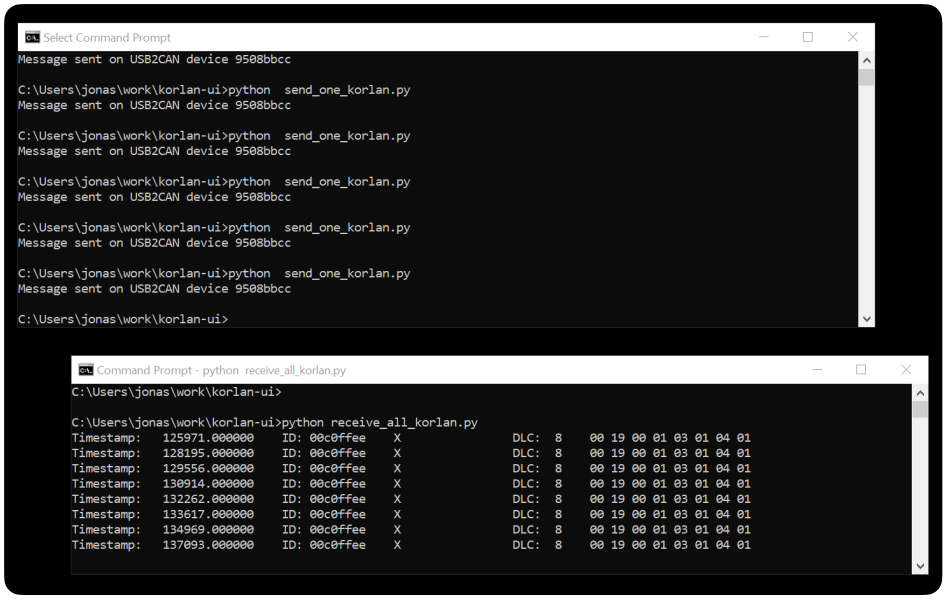
Examples are based on python-can library programming examples and are slightly modified, adding Korlan USB2CAN specific information and providing an explicit path to the usb2can.dll library.
You will need to install a python-can library using your favorite Python package installer.

send_one_korlan.py - sends a single CAN bus message.

```
import can

with can.interface.Bus(bustype="usb2can", channel="9508bbcc", bitrate=1000000, dll='./
usb2can.dll') as bus:
    msg = can.Message(arbitration_id=0xC0FFEE, data=[0, 25, 0, 1, 3, 1, 4, 1],
is_extended_id=True)
    try:
        bus.send(msg)
        print(f"Message sent on {bus.channel_info}")
    except can.CanError:
        print("Message NOT sent”)
```

receve_all_korlan.py - receiving and printing CAN bus messages in the loop until Ctrl-C is pressed.

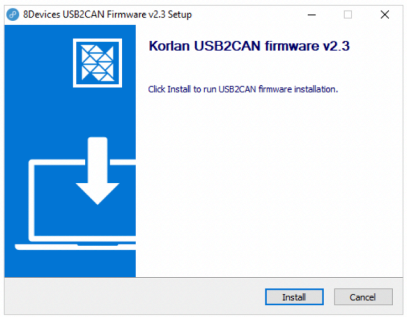
```
import can

with can.interface.Bus(bustype="usb2can", channel="d3365afb", bitrate=1000000, dll='./
usb2can.dll') as bus:
    try:
        while True:
            msg = bus.recv(1)
            if msg is not None:
                print(msg)
    except KeyboardInterrupt:
        pass  # exit normally
```

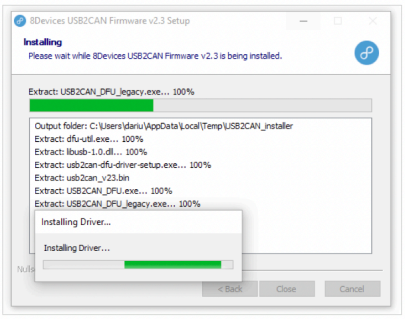


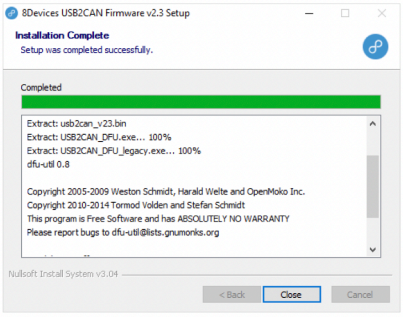
4.4 Firmware upgrade

Firmware upgrade to version 2.3 is straightforward on Windows operating system. Follow the instructions below.

1. Connect your Korlan USB2CAN adapter to the USB port of your computer.
2. Download Korlan USB2CAN Firmware Upgrade Tool v2.3 from our website: https://www.8devices.com/media/products/usb2can_korlan/downloads/USB2CAN_v23.exe
3. Run the USB2CAN_v23.exe file to start the firmware upgrade.
4. Follow instructions and wait until the firmware upgrade is finished before disconnecting Korlan USB2CAN device.







5. Korlan USB2CAN test application - kcan

5.1 About the application

This application is designed for Korlan USB2CAN device testing. It is written in python and relies on a versatile python-can library. The application is supported on Linux and Windows operating systems. 8devices provide the source code of kcan application under a very friendly license for your reference and experiments. The code with installation instructions is available on github: https://github.com/8devices/korlan-usb2can-test-application.
If you are using Windows operating system and you do not need the source code, you can also use this application simply by installing and running kcan.exe file on your computer, which you can find here: https://wiki.8devices.com/korlan.

Notes:
- Make sure to select the proper device ID and set the correct bit rates when using this program. Bit rates between two communicating devices must have the same value.
- You can run multiple instances of kcan application on your computer to control multiple Korlan USB2CAN devices.
- If you use Linux operating system, you have to run the application with sudo for it to work properly.

5.2 Instructions for using the program

Connect Korlan device to your computer and run the Korlan test application. You should see configuration tab displayed in the program's window.

Select desired Korlan USB2CAN device id from dropdown menu if you have connected multiple Korlan devices on your computer.

Select the desired bit rate from the dropdown menu.

Select wanted options for Korlan's interface:
- Loopback.
- Silent.
- Disable auto retry.

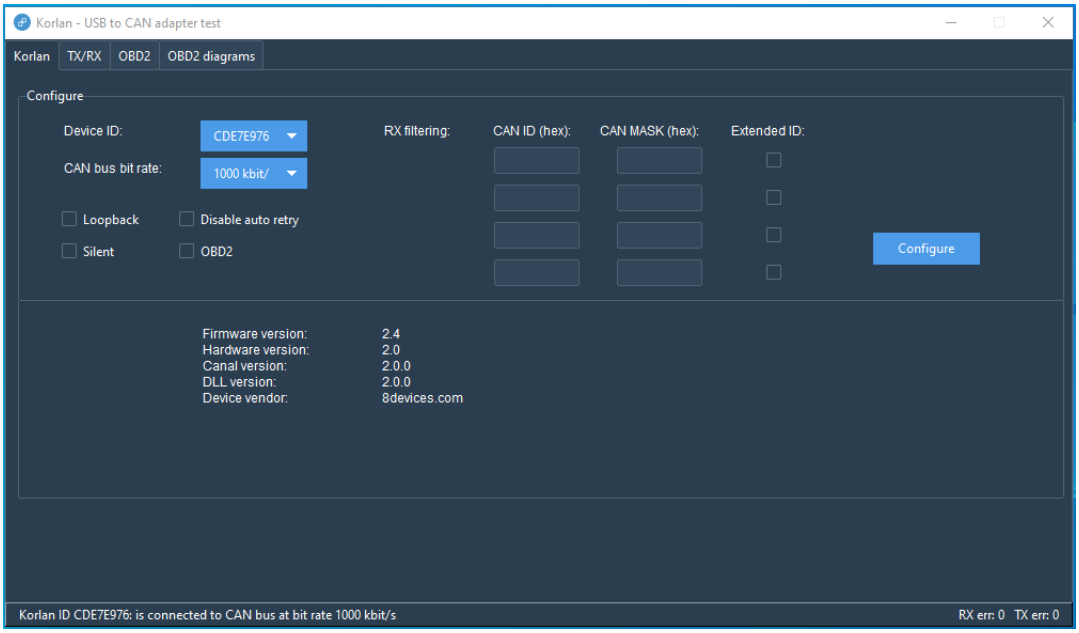
Loopback, silent and silent loopback testing modes are explained on page 7 of this document.
Disable auto retry ensures that if an error occurs, Korlan does not try to resend a TX frame.

Set filters if you want to receive only certain RX messages with specific can ids. Enter desired CAN ID and CAN MASK.

The filter mask is used to determine which bits in the ID of the received frame are compared with the filter:
- If a mask bit is set to a zero, the corresponding ID bit of a received frame will automatically be accepted, regardless of the value of the filter bit.
- If a mask bit is set to a one, the corresponding ID bit of a received frame will be compared with the value of the filter bit; if they match it is accepted otherwise the frame is rejected.

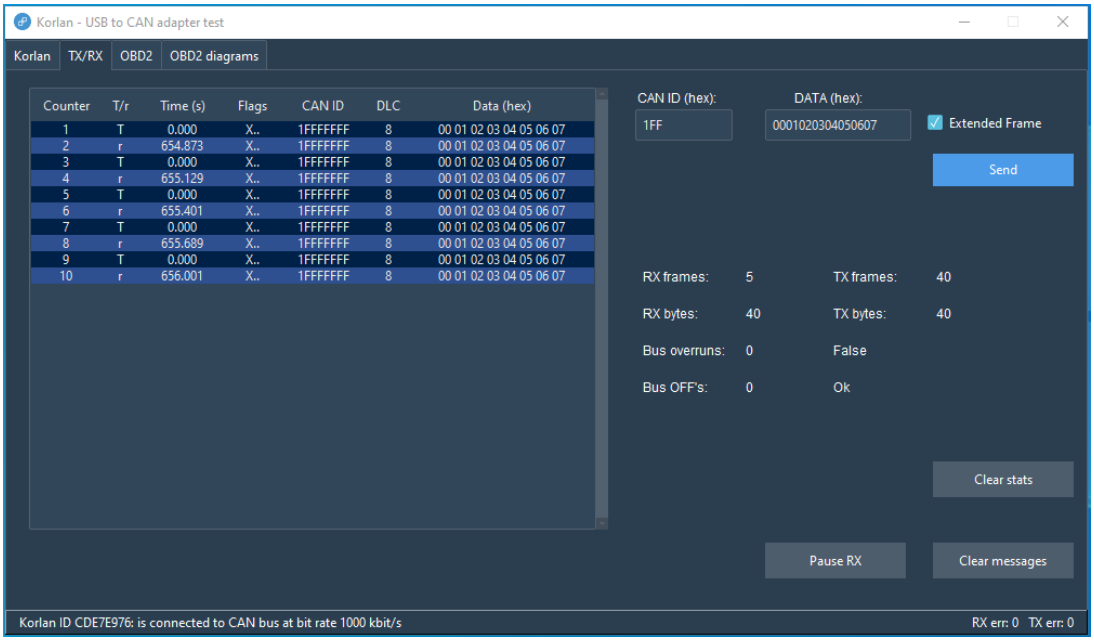
If you want every bit to match the filter, set can mask to 1FFFFFFF.

Click on the Configure button. You will observe device details and Korlan connection status on the status bar at the bottom of the window.

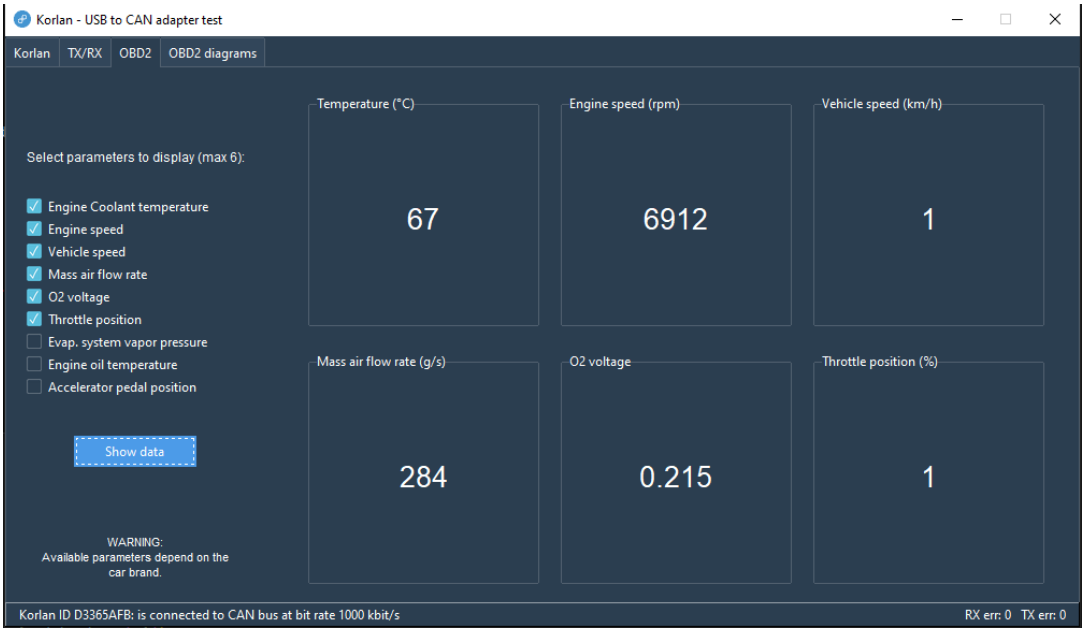


Now you can proceed to the TX/RX tab and observe received and transmitted messages on a scrollable messages pane. You shall set desired CAN ID and DATA for transmitting messages, then click on the button Send.

If you want, you can reset bus statistics by clicking Clear stats button or clear message pane by clicking Clear messages button.
Also, you can stop RX message display for more attentive inspection by clicking on the Pause RX button.



In OBD2 tab you can select up to 6 parameters and press Show data button to get human-readable data about your vehicle. You will have to connect Korlan to your vehicle's OBD2 connector which is near the steering wheel. Note that some of the suggested parameters won't be available on your vehicle, because parameters ids depend on the car brand.



In OBD2 diagrams tab you can see OBD2 data displayed in the real time diagrams. You can choose up to 4 parameters to be displayed.