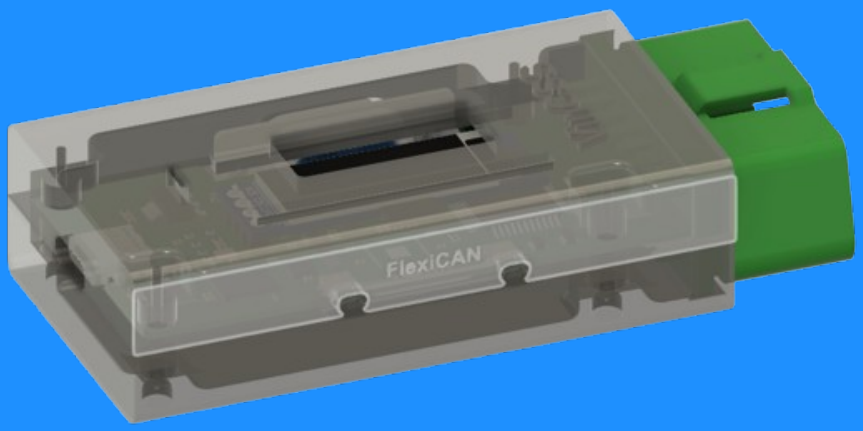
FlexiCAN User Manual

USB-to-CAN adapter for vehicle penetration testing.

Connected Car Center



DocID: VCSSFLEXICAN250924

Version: V1.0.0

December 16, 2024

Contents

1. Get Started

This document will help you set up the environment and tools needed to use FlexiCAN. At the same time, it also guides you to use this new tool effectively and professionally.

Introduction

FlexiCAN is one of the easiest and least expensive ways to connect a computer to a CAN bus network for monitoring and transmitting CAN and CAN FD data. Featuring a modern USB Type C connector, a super-flexible OBD-II connector, yet retaining a standard 9-pin D-SUB connector, the sleek, ergonomically designed FlexiCAN housing is rugged enough for everyday use yet small and flexible enough for use in space-constrained applications.

FlexiCAN can handle up to 20,000 messages per second, each time-stamped to 50 microsecond accuracy. No external power supply is required with passive noise filtering as standard.

FlexiCAN is a device that allows you to flexibly configure the can_high can_low pins of the canbus to any pin on the OBD-II port via a software-based hot-swappable semiconductor switching matrix. It allows you to route connections to non-standard but hidden canbus ports that exist on the OBD-II port without the need for proprietary connection cables.

Product Parameters

Parameter	Detailed description
Bitrate	20 - 1000kbps
CAN FD Bitrate	Up to 5 Mbit/s
Categories	Interfaces, Hardware, CAN
Channels	01
Connectors	DSUB 9, OBD-II, Type-C USB
Config	Onboad Shell, Button
Display	LED Signal, OLED Screen
Material	PETG, UV-Resin
Temp range	-10 °C to 60 °C

- The table above shows the basic parameters of the device, please see the ***advanced section*** below to better understand the device.

Major Features

- Optionally configure the connection between OBD-II and the can device
- Configure termination resistor (120R) by software
- Fast, flexible access commands
- Quick and easy plug and play installation.
- Support CAN FD, up to 5 Mbit/s.
- Supports both 11-bit (CAN 2.0A) and 29-bit (CAN 2.0B active) identifiers.
- Fully compatible with Linux, CAN Socket.
- High-speed CAN connection (ISO 11898-2 compliant), up to 1 Mbit/s.
- Support silent mode for analysis tools – listen to the bus without causing interference. (comming soonn…)

First start

Thank you so much for choosing Flexi CAN! We look forward to you exploring all the possibilities this device has to offer and hearing about the great things you will achieve with it. Enjoy your new Flexi CAN.

Preparation

Flexi CAN is designed to work perfectly with Linux and SocketCAN, of course it also works with Windows, but to get you started quickly and easily we will start with Linux, in this case I use Ubuntu 22.04, my experience is that you should use newer distributions as it supports SocketCAN better.

SocketCAN is built into the kernel provided by Linux distributions like Ubuntu, Debian, Arch…. You will need to install the can-utils package to use the SocketCAN utilities (candump, cansend, etc):

```
sudo apt update

sudo apt-get install can-utils
```

To control and configure the FlexiCAN connection, you will need a serial to connect to the FlexiCAN shell, in this case, I use `screen`

```
sudo apt install screen
```

Note: On linux: to exit `screen` , you can kill the session with CTRL + a, then , and confirm it with y On MacOS: to exit `screen` , you can kill the session with CTRL + a, then k, and confirm it with y

Powering on

Turn on FlexiCAN by plugging the USB Type C cable into the Type C port of the device, then plugging the device into the OBD-II port of the vehicle. Note, reversing the order of plugging will not damage the FlexiCAN, but it may affect the canbus network on the vehicle, because the behavior that occurs when the FlexiCAN is not powered is uncontrollable, it may become a passive resistor between the two CAN high CAN low pins of the network, it may also cause a reflection effect on the can network or something else…

First Config

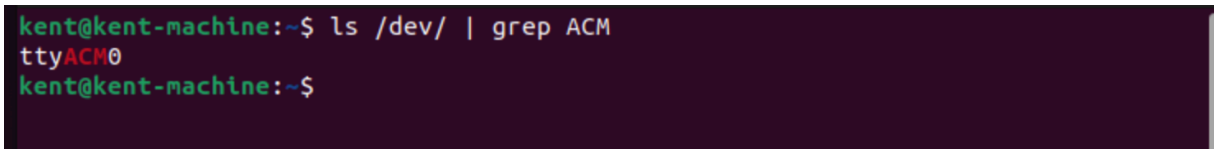
Configuring FlexiCAN lets it know which of the 16 pins of the OBD-II port you want to connect the canbus tool to. To configure it we start by connecting to the FlexiCAN shell:

```
1  sudo screen /dev/ttyACM0 921600
```

In there:

- sudo screen - run screen as ‘`sudo`
- /dev/ttyACM0 - the tty number of the FlexiCAN shell, bạn có thể tìm nó bằng lệnh `ls /dev/tty*`
- 921600 - the speed of the serial connection

In this example, the FlexiCAN shell is located at /dev/ttyACM0



After running the command `sudo screen /dev/ttyACM0 921600` successfully, press `Enter`, FlexiCAN shell will appear

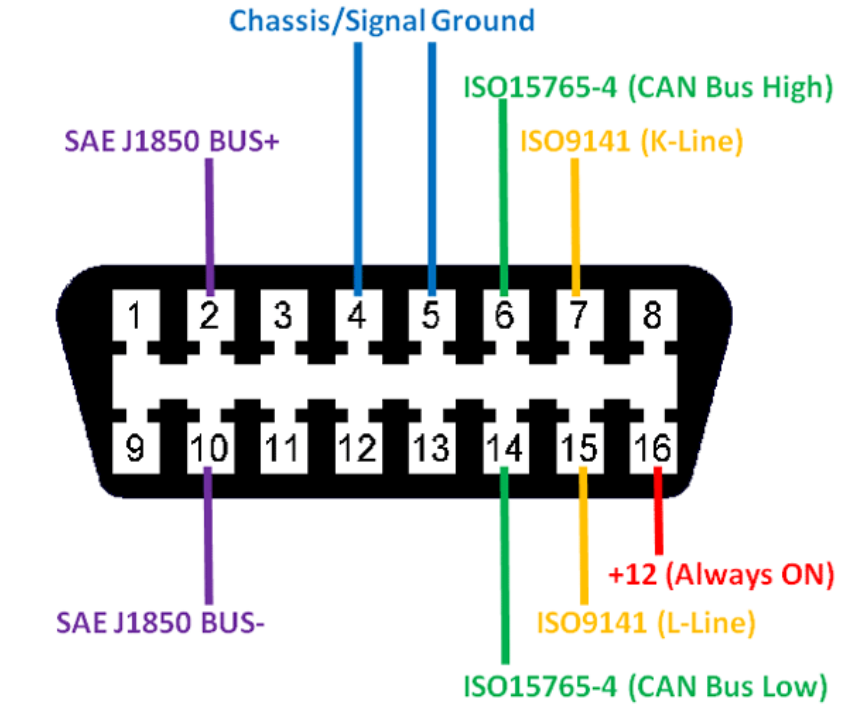
Now we can start configuring the device, to understand more about the commands and get instructions for each command, type `help` , similarly, type the commands and add`--help` after to get instructions

```
root@vincss:/$ help
total function 15
  OBDII_layout     -> /bin
  toggle_resistor  -> /bin
  reboot_dfu       -> /bin
  reset_factory    -> /bin
  change_hostname  -> /bin
  change_username  -> /bin
  change_passwd    -> /bin
  active_profile   -> /bin
  change_profile   -> /bin
  remove_profile   -> /bin
  add_profile      -> /bin
  list_profiles    -> /bin
  exit             -> /sbin
  help             -> /sbin
  shsize           -> /sbin

total variable 2
  $ZERO r-   1
  $PATH r-  11
root@vincss:/$
```

Connect to the Bus

Now let's start configuring FlexiCAN to connect to a bus, in this case I will configure it to connect to the standard canbus location in the OBD-II port, where CAN High is pin 6, CAN Low is pin 14.



To configure, run the command `add_profile OBD-II 6 14 active` and get the result:

```
root@FlexiCAN:/$ list_profiles
No profiles is available!
root@FlexiCAN:/$ add_profile
Add profile
add_profile <PROFILE NAME> <CAN HIGH PIN> <CAN LOW PIN> <ACTIVE|NOTACTIVE>
Default profile is not active.
root@FlexiCAN:/$ add_profile "OBD-II" 6 14
Profile added successful!
root@FlexiCAN:/$ list_profiles
-----------------------------------------------------
| ID|               Name|CAN high| CAN low|    Status|
-----------------------------------------------------
|  0|             OBD-II|       6|      14|    active|
-----------------------------------------------------
root@FlexiCAN:/$
```

In there:

- add_profile is the command to configure the canbus link via the switching matrix
- “OBD-II” : Name of the configuration/profile
- 6 : connect can_h pin to pin 6 on OBD-II port
- 14: connect can_l pin to pin 14 on OBD-II port
- active: active this profile or not. It depends on how many profiles on your device. If there is only one profile then that profile will be actived by default.

The device is now connected according to the above configuration and ready to use. You can save up to 16 profiles.

However, I wanted to see how many configurations the device had saved, so I ran the `list_profiles` command to see the configurations saved on the device.

```
root@FlexiCAN:/$ list_profiles
-----------------------------------------------------
| ID|               Name|CAN high| CAN low|    Status|
-----------------------------------------------------
|  0|             OBD-II|       6|      14|    active|
-----------------------------------------------------
|  1|           Body CAN|       3|       5|not active|
-----------------------------------------------------
|  2|             PT CAN|       9|       8|not active|
-----------------------------------------------------
root@FlexiCAN:/$
```

We can see, the new configuration is saved in 0th position, and to connect to another CAN bus also located in this OBD-II port, such as `"Body CAN"` we can run the command `active 1` without reassigning **can_h can_l** again, as shown below, the connection is complete.

```
root@FlexiCAN:/$ active_profile --help
Active profile
active_profile <ID>
root@FlexiCAN:/$ active_profile 1
Active profile 1 done
root@FlexiCAN:/$ list_profiles
-------------------------------------------------
| ID|               Name|CAN high| CAN low|    Status|
-------------------------------------------------
|  0|            OBD-II|       6|      14|not active|
-------------------------------------------------
|  1|          Body CAN|       3|       5|    active|
-------------------------------------------------
|  2|            PT CAN|       9|       8|not active|
-------------------------------------------------
root@FlexiCAN:/$
```

Fly on the Bus

Now, let's initialize SocketCAN and start using it. Run the following command to configure cantools:

```
1  sudo ip link set dev can0 up type can bitrate 500000
2
3  sudo ip link set can0 txqueuelen 10000 #optional
```

This command will configure cantool on FlexiCAN at 500000 kbps bitrate, CAN 2.0, with txqueue len is 10000.

Run ifconfig you can see, cantools has been initialized successfully.

```
kent@kent-machine:~$ sudo ip link set dev can0 up type can bitrate 500000
kent@kent-machine:~$ ifconfig
can0: flags=193<UP,RUNNING,NOARP>  mtu 16
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 10  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Now you can start using canbus with can-utils on Linux. When initialized successfully you will see red (tx) and blue (rx) leds appear, red (tx) led will flash When you send data, the blue led (rx) will flash when receiving data.

Updating the firmware

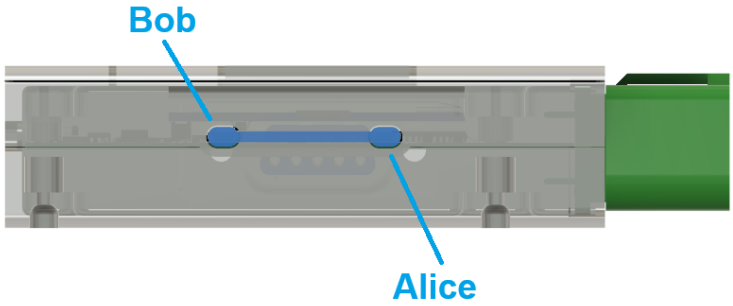
To update the FlexiCAN to the latest Firmware, put it into DFU mode using the `reboot_dfu` command

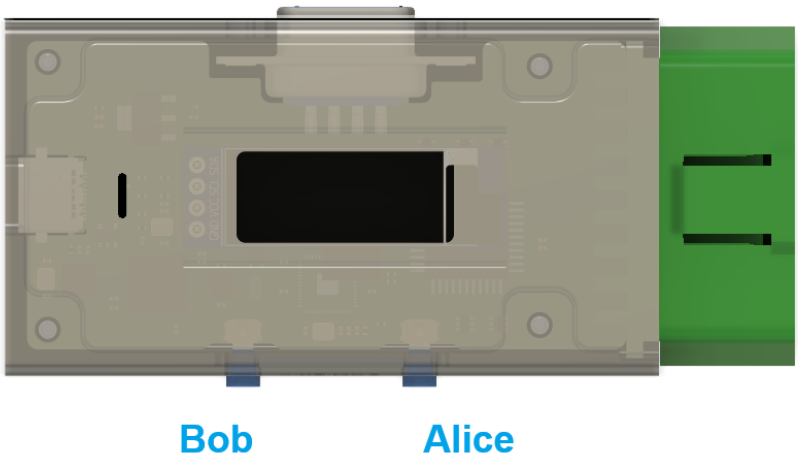
2. Advance Guide

This section will provide detailed instructions and tips to help you use the device quickly, efficiently and professionally.

Quick Select Button & play with the FlexiCAN's menu

On the side of the device, we will see two small buttons called Bob and Alice. Bob will be deactived by default





Once FelixCAN is configured, long press (greater than 2 seconds) the Alice button to switch to change configuration as below:

<> PT CAN means we are in select configuration mode. To switch between configuration, we need to press Bob button. Once we selected a right configuration, we can press Alice again to select.

If we short press (less than 2 seconds) the Alice button, FlexiCAN will show the setting menu as below:


1. Enable/disable 120


2. Reset factory

To select one of those, we can long press Bob button to select and press Alice to go back to main. For example, we enable 120R in the CAN connection as below: Before enable

Enable it

After enable

SocketCAN on Linux

This section will guide you in more detail on how to use FlexiCAN on Linux. In the getting started section, I mentioned the simple device configuration using the command:

```
1  sudo ip link set dev can0 up type can bitrate 500000
```

```
2
3  sudo ip link set can0 txqueuelen 10000
```

To configure cantool with FD CAN, run the following command:

```
1  sudo ip link set dev can0 up type can bitrate 500000 dbitrate 2000000 fd on
2
3  sudo ip link set can0 txqueuelen 10000
```

In there:

- dbitrate 2000000 means fd can is operating at bitrate 2Mbps

Then we can use `can-utils` to send and receive some can messages, here is an example

```
1   cansend can0 999#DEADBEEF                  # Send a frame to 0x999 with payload 0
        xdeadbeef
2   cansend can0 421##311223344                # Send a CAN FD frame, <can_id>##<flags>{data}
3   candump can0                               # Show all traffic received by can0
4   canbusload can0 500000                     # Calculate bus loading percentage on can0
5   cansniffer can0                            # Display top-style view of can traffic
6   cangen can0 -D 11223344DEADBEEF -L 8       # Generate fixed-data CAN messages
```

To use with python, you can pre-configure cantools via the command above or configure it directly via python:

```
1  import can
2
3  bus = can.Bus(interface="gs_usb", channel=dev.product, index=0, bitrate=250000)
```

Remember to install library before using: `pip install "python-can[gs_usb]"`

Usage: pass device `index` (starting from 0) if using automatic device detection.

Alternatively, pass `bus` and `address` to open a specific device. The parameters can be got by `pyusb` as shown below:

```
1  import usb
2  import can
3
4  dev = usb.core.find(idVendor=0x1D50, idProduct=0x606F)
5  bus = can.Bus(
6      interface="gs_usb",
7      channel=dev.product,
8      bus=dev.bus,
9      address=dev.address,
10     bitrate=250000
11 )
```

FlexiCAN is also fully compatible with Caring Caribou, a favorite and useful tool for vehicle testing (though I'm not a big fan of it).

udev Rules

During use, you will notice that having to type such a long script on the terminal to configure cantools every time you use it is a bit inconvenient, sometimes it is even hard to remember. This is mandatory

because it is the process of Linux and SocketCAN, I cannot change this, but I can help you solve it, this is the way I usually use.

First, we will create a script file, for example `init_flexican.sh` with the following content:

```
#!/bin/bash

# Loop through all CAN devices present
for can in $(ip -o link show | grep 'can[0-9]' | awk -F': ' '{print $2}')
do
    state=$(ip link show $can | grep 'state' | awk '{print $9}')

    if [ "$state" == "DOWN" ]; then
        echo "Initializing $can"
        sudo ip link set dev $can up type can bitrate 500000
        sudo ip link set $can txqueuelen 10000
    fi
done
```

Then run the following command to make sure the script has executable permissions:

```
chmod +x init_can.sh
```

Using udevto automatically run the script, I will create a `udev rules` to detect when a CAN device is plugged in and run the script. Create a file `/etc/udev/rules.d/99-can.rules` with the following content:

```
SUBSYSTEM=="net", ACTION=="add", KERNEL=="can[0-9]*", RUN+="/path/to/init_flexican.sh"
```

Replace `/path/to/init_flexican.sh` with the actual path to the script you created. Then run this command to apply the changes in `udev` :

```
sudo udevadm control --reload-rules
```

From now on, when FlexiCAN devices are plugged in,udev will detect them and run your script to initialize them. You can use them right away without having to type the initialization command anymore.

Terminal Resistor

Terminating resistors are necessary in CAN bus systems because CAN communication is bidirectional. Terminating resistors at each end absorb the CAN signal energy, ensuring that this energy is not reflected from the cable ends. Such reflections would introduce noise and potentially corrupt the signal.

The reflection challenge increases with cable length as well as the bit rate of the CAN bus. This is why it is important to add appropriate termination points in larger CAN networks.

For best results, the CAN bus termination must match the nominal impedance of the cable, which for ISO 11898-2 (High Speed CAN) is specified at 120 Ohm. Therefore, 120 Ohm termination adapters are considered standard for the CAN bus.

FlexiCAN allows you to add or remove this terminal resistor entirely in software, you don't need to unjump, solder pins or buy an external converter adapter, all you need to do is just do it via the command or enable via FlexiCAN's menu as above.

```
1 toggle_resistor
```

In there:

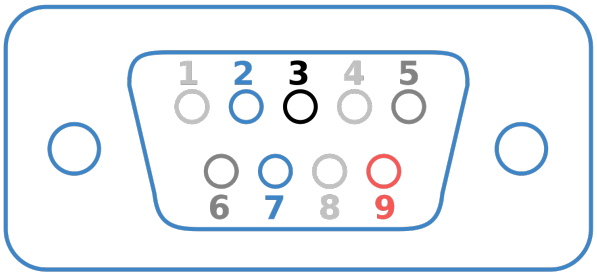
- toggle_resistor will call the command to configure the terminal resistor either enable or disable it, depends on current status of resistor.

The FlexiCAN OLED display will also display the status of the terminal resistor on the screen, where `120R` corresponds to a resistor connected to the network or no display anything corresponds to a resistor not connected.

External Port

FlexiCAN is equipped with 1 standard compliant DSUB DB9 canbus port (pin 9 - Power not connected).

Therefore, you can connect additional 3rd party cantools to the canbus network, or use FlexiCAN as an OBD-II to DB9 adapter.



1 Not connected
2 CAN low
3 Ground
4 Not connected
5 (Shield - optional)
6 (Ground - optional)
7 CAN high
8 Not connected
9 Power

———- END ———-