

FINTEK

F81604

USB to CANBUS

Driver Installation Guide

for Linux

v1.22

Feb 13, 2025

1 / 6

1. Preliminary

This document is for Fintek F81604 USB to 1/2 CANBUS driver installation in Linux and can-utils to verify the CANBUS.

2. Building Driver

1. Prepare the kernel tree & compiler tools for your distribution.
 - `sudo su`
 - `apt-get update`
 - `apt-get install build-essential fakeroot gcc kernel-package libncurses5-dev`
if your target system is Debian/Ubuntu based
2. Unzip the driver.zip
3. `cd driver`
4. if you build this driver for desktop linux, you can skip this.
 - Modify makefile default section from
`make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules`
to
`make -C <android_kernel_path> M=$(PWD) modules`
5. `make clean ; make ; make install`
6. `reboot`
7. Use the following command to check CANBUS device is available (can0/can1/... etc.):
`ls /sys/class/net/ -al`

```
root@code-H11H4-IM:/home/code# ls /sys/class/net/
nd/ net/
root@code-H11H4-IM:/home/code# ls /sys/class/net/ -al
total 0
drwxr-xr-x  2 root root 0 六  4 10:08 .
drwxr-xr-x 63 root root 0 六  4 08:37 ..
lrwxrwxrwx  1 root root 0 六  4 10:08 can0 -> ../../devices/pci0000:00/0000:00:14.0/usb1/1-3/1-3.2/1-3.2:1.0/net/can0
lrwxrwxrwx  1 root root 0 六  4 10:08 can1 -> ../../devices/pci0000:00/0000:00:14.0/usb1/1-3/1-3.2/1-3.2:1.0/net/can1
lrwxrwxrwx  1 root root 0 六  4 08:38 enp7s1 -> ../../devices/pci0000:00/0000:00:1d.1/0000:06:00.0/0000:07:01.0/net/enp7s1
lrwxrwxrwx  1 root root 0 六  4 08:37 lo -> ../../devices/virtual/net/lo
root@code-H11H4-IM:/home/code#
```

3. Configure CANBUS

The following examples will configure “can0” to bit-rate 250000, sample-point 0.875 and error restart with 100ms.

1. `sudo su`
2. `ip link set can0 down`
3. `ip link set can0 type can restart-ms 100`
4. `ip link set can0 type can bitrate 250000 sample-point 0.875`
 - This value should be fine-tune by customer or following table, and the clock should set with half clock source (24MHz / 2 = 12Mhz)

```
root@code-h11h4im:/home/code# can-calc-bit-timing -c 12000000 sjal000
```

Bit timing parameters for sjal000 (cmd-line) with 12.000000 MHz ref clock

nominal							real	Bitrt	nom	real	SampP			
Bitrate	TQ[ns]	PrS	PhS1	PhS2	SJW	BRP	Bitrate	Error	SampP	SampP	Error	BTR0	BTR1	
1000000	83	4	4	3	1	1	1000000	0.0%	75.0%	75.0%	0.0%	0x00	0x27	
800000	83	5	6	3	1	1	800000	0.0%	80.0%	80.0%	0.0%	0x00	0x2a	
500000	250	3	3	1	1	3	500000	0.0%	87.5%	87.5%	0.0%	0x02	0x05	
250000	250	6	7	2	1	3	250000	0.0%	87.5%	87.5%	0.0%	0x02	0x1c	
125000	500	6	7	2	1	6	125000	0.0%	87.5%	87.5%	0.0%	0x05	0x1c	
100000	1250	3	3	1	1	15	100000	0.0%	87.5%	87.5%	0.0%	0x0e	0x05	
50000	1250	6	7	2	1	15	50000	0.0%	87.5%	87.5%	0.0%	0x0e	0x1c	
20000	3333	6	6	2	1	40	20000	0.0%	87.5%	86.6%	1.0%	0x27	0x1b	
10000	5000	8	8	3	1	60	10000	0.0%	87.5%	85.0%	2.9%	0x3b	0x2f	

5. `ip link set can0 type can berr-reporting on`
6. `ifconfig can0 txqueuelen 1000`
7. `tc qdisc add dev can0 root handle 1: pfifo`
8. `ip link set can0 up`

If you want to change the CANBUS setting in your application, we can use “system()” to execute above command or “canconfig.c” in “config tools” programmatically via SocketCAN/Netlink.

3. Using can-utils to operate CANBUS

We can get can-utils with following command.

- Debian/Ubuntu
 - apt-get install can-utils
- Frdora/Centos/RHEL
 - yum install can-utils
- Source code download link
 - <https://github.com/linux-can/can-utils>

We'll use "candump" to receive data, "cangen" & "cansend" to send data. The "cangen" will send random data & ID and "cansend" will send specific data & ID to CANBUS.

```

root@code-H11H4-IM: /home/code [113x33]
連線(C) 編輯(E) 檢視(V) 視窗(W) 選項(O) 說明(H)
When incrementing the CAN data the data length code minimum is set to 1.
CAN IDs and data content are given and expected in hexadecimal values.

Examples:
cangen vcan0 -g 4 -I 42A -L 1 -D i -v -v    (fixed CAN ID and length, inc. data)
cangen vcan0 -e -L i -v -v -v              (generate EFF frames, incr. length)
cangen vcan0 -D 11223344DEADBEEF -L 8      (fixed CAN data payload and length)
cangen vcan0 -g 0 -i -x                    (full load test ignoring -ENOBUFFS)
cangen vcan0 -g 0 -p 10 -x                 (full load test with polling, 10ms timeout)
cangen vcan0                               (my favourite default :)

root@code-H11H4-IM:/home/code/ddd/old/hpeter/fintek/F81601/driver# cangen can0 -n 4
root@code-H11H4-IM:/home/code/ddd/old/hpeter/fintek/F81601/driver#

-----
root@code-H11H4-IM:/home/code#
root@code-H11H4-IM:/home/code#
root@code-H11H4-IM:/home/code#
root@code-H11H4-IM:/home/code#
root@code-H11H4-IM:/home/code#
root@code-H11H4-IM:/home/code#
root@code-H11H4-IM:/home/code# candump can1
can1 3DE [1] 10
can1 241 [8] 15 89 14 08 20 89 1D 09
can1 54A [0]
can1 60E [8] 68 CB 4C 0A A3 15 A8 37

root@code-H11H4-IM: /home/code [113x33]
連線(C) 編輯(E) 檢視(V) 視窗(W) 選項(O) 說明(H)
<can_id>##<flags>{data} for CAN FD frames

<can_id> can have 3 (SFF) or 8 (EFF) hex chars
{data} has 0..8 (0..64 CAN FD) ASCII hex-values (optionally separated by '.')
<flags> a single ASCII Hex value (0 .. F) which defines canfd_frame.flags

e.g. 5A1#11.2233.44556677.88 / 123#DEADBEEF / 5AA# / 123##1 / 213##311
1F334455#1122334455667788 / 123#R for remote transmission request.

root@code-H11H4-IM:/home/code# cansend can0 123#R
root@code-H11H4-IM:/home/code#

-----
root@code-H11H4-IM:/home/code# candump can1
can1 123 [8] 11 22 33 44 55 66 77 88
can1 123ABCDE [8] 11 22 33 44 55 66 77 88
can1 123 [0] remote request

-----
May 9 14:20:00 code-H11H4-IM kernel: [ 367.753899] sja1000_f81601 0000:01:00:0 can0: bit-timing not yet defined
May 9 14:20:16 code-H11H4-IM kernel: [ 383.815728] sja1000_f81601 0000:01:00:0 can0: setting BTR0=0x01 BTR1=0x1
c
May 9 14:20:16 code-H11H4-IM kernel: [ 383.818759] sja1000_f81601 0000:01:00:0 can1: setting BTR0=0x01 BTR1=0x1
c
May 9 14:20:38 code-H11H4-IM kernel: [ 405.567568] can: controller area network core (rev 20170425 abi 9)
May 9 14:20:38 code-H11H4-IM kernel: [ 405.567596] NET: Registered protocol family 29
May 9 14:20:38 code-H11H4-IM kernel: [ 405.575128] can: raw protocol (rev 20170425)

[0] 0:~bash- 1:~bash* "code-H11H4-IM" 14:22 09- ㄝ -18

```

We can access the website to get more detail usage and source code.

Manpage manual:

<http://manpages.ubuntu.com/manpages/bionic/man1/candump.1.html>

<http://manpages.ubuntu.com/manpages/bionic/man1/cangen.1.html>

<http://manpages.ubuntu.com/manpages/bionic/man1/cansend.1.html>

Source code:

<https://github.com/linux-can/can-utils/blob/master/candump.c>

<https://github.com/linux-can/can-utils/blob/master/cangen.c>


<https://github.com/linux-can/can-utils/blob/master/cansend.c>

4. Output pin control (CTRL1/2)

We can use following command to read or set F81604 CTRL1/2 pin to logic high or low, default is logic low.

1. `cat /sys/class/net/<can target>/terminator_control`
view current CTRL pin status
2. `echo 1 > /sys/class/net/<can target>/terminator_control`
set CTRL pin logic high
3. `echo 0 > /sys/class/net/<can target>/terminator_control`
set CTRL pin logic low

e.g.

 root@code: /home/code

```
root@code:/sys/class/net/can2#
root@code:/sys/class/net/can2#
root@code:/sys/class/net/can2# cat /sys/class/net/can0/terminator_control
0
root@code:/sys/class/net/can2# echo 1 > /sys/class/net/can0/terminator_control
root@code:/sys/class/net/can2# cat /sys/class/net/can0/terminator_control
1
root@code:/sys/class/net/can2#

36: can0: <NOARP,ECHO> mtu 16 qdisc noop state DOWN group default qlen 10
    link/can
37: can1: <NOARP,ECHO> mtu 16 qdisc noop state DOWN group default qlen 10
    link/can
```

4. Q&A

Q1: Encounter “No Buffer space available” with “cangen” tools.

```
root@code-H11H4-IM:/home/code#  
root@code-H11H4-IM:/home/code# cangen can0 -g 0  
write: No buffer space available  
root@code-H11H4-IM:/home/code#
```

A1: To enlarge tx buffer by command “ifconfig can0 txqueuelen 1000” or ignore the message with parameter “cangen -i”

Q2: Can’t load driver when system reboot with Kylin (銀河麒麟).

A2: Run:

```
sudo kysec_set -n exectl -v original /lib/modules/`uname -r`/updates/f81604.ko
```

to entrust the driver and reboot.