

NKCAN Linux user manual

Version: 1.0.0
Data: 24/06/04
Author: EA



NKCAN_Linux user manual

NODKA

1.0.0

Contents of the directory:

1. History	2
2. product presentation	3
2.1. Specification parameters	3
2.2. System requirements	4
2.3. Supply information	4
3. Drivers and API	5
3.1. Linux	5
3.1.1. Install driver	5
3.1.2. Uninstall driver	7
3.1.3. Configure the CAN bus	8
3.1.4. Use CAN-utils to operate the CAN bus	8
3.1.5. Routine	9
4. upgrade	13
4.1. Upgrade the software	13
4.2. Upgrade Procedure	13

1. History

Version	Date	Author	Description
1.0.0	2023-1-10	EA	nitial version

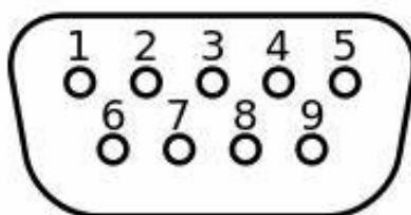
2. product presentation

NKCAN adapter CAN easily connect a computer to the CAN network via USB, CAN be connected to the CAN bus, with the corresponding cable can be connected to a maximum of 2 CAN, each CAN channel separately isolated USB, maximum 500V. The card supports new features of the CAN2.0A and CAN2.0B protocol specifications, so CAN nodes CAN be used in existing CAN networks.

2.1. Specification parameters

- High-speed USB 2.0 adapter (compatible with USB 1.1 and USB 3.0).
- Connect the CAN bus via connection cable and D-Sub, 9-pin connector.
- The 120 ohm terminal resistor is placed on the D-Sub connector and can be switched on or off.
- Compatible with CAN specifications 2.0A(11-bit ID) and 2.0B(29-bit ID).
- CAN supports a bit rate ranging from 5k bits/s to 1 MB bits/s.
- Bus load measurement includes error frames and overload frames.
- CAN power via USB.
- Operating temperature range -40-85°C (-40 to 185 °F).

D-Sub interface pin diagram



Pin	Pin definition
1	No connection/Optional +5V output
2	CAN_L
3	GND
4	No connection
5	No connection
6	GND
7	CAN_H

8	No connection
9	No connection

2.2. System requirements

- Free full-size miniPCIE slots on the computer.
- Operating system Windows 10, Windows 11 (x64/ARM64), or Linux (32/64 bit).

2.3. Supply information

- NKCAN card
- Connection cables, including D-Sub plugs for each channel.
- Drivers for Windows 11 (x64/ARM64), 10 (x64), and Linux
- User manual in PDF format.

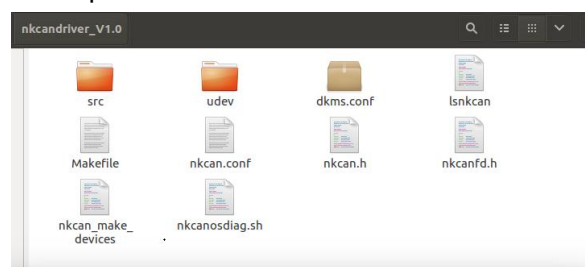
3. Drivers and API

3.1. Linux

This chapter explains how to build and install drivers on the Linux platform, including testing methods.

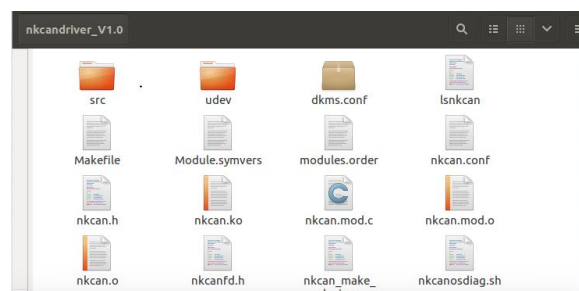
3.1.1. Install driver

1. Open the terminal under this file and enter: make



```
demo@demo: ~/nkcdriver_V1.0
demo@demo:~/nkcdriver_V1.0$ make
**
** Making nkcan driver in chardev mode
**
** Host OS Release=Ubuntu v16.04
** Host machine kernel version=4.15.0-142-generic
** Driver kernel version=4.15.0-142-generic (4.15.0)
** Path to kernel sources=/usr/src/linux-headers-4.15.0-142-generic
** use KBUILD=yes
** use DKMS=
** gcc version=5.4.0
**
make -C /usr/src/linux-headers-4.15.0-142-generic EXTRA_CFLAGS="-I/home/demo/nkcdriver_V1.0 -DNO_DEBUG -DMODVERSIONS -DNO_PARPORT_SUBSYSTEM -DUSB_SUPPORT -DPCI_SUPPORT -DPCIE_SUPPORT -DISA_SUPPORT -DNO_DONGLE_SUPPORT -DNO_PCCARD_SUPPORT -DNO_NETDEV_SUPPORT -DNO_RT -Wno-date-time" V=0 modules SUBDIRS=/home/demo/nkcdriver_V1.0
make[1]: Entering directory '/usr/src/linux-headers-4.15.0-142-generic'
make[1]: Warning: File '/home/demo/nkcdriver_V1.0/src/nkcan_main.c' has modified
```

2. use command: Make netdev



```
demo@demo: ~/nkcdriver_V1.0
demo@demo:~/nkcdriver_V1.0$ make netdev
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-142-generic'
demo@demo:~/nkcdriver_V1.0$ make netdev
make NET-DEV_SUPPORT
make[1]: Entering directory '/home/demo/nkcdriver_V1.0'
**
** Making nkcan driver in netdev mode
**
** Host OS Release=Ubuntu v16.04
** Host machine kernel version=4.15.0-142-generic
** Driver kernel version=4.15.0-142-generic (4.15.0)
** Path to kernel sources=/usr/src/linux-headers-4.15.0-142-generic
** use KBUILD=yes
** use DKMS=
** gcc version=5.4.0
**
make -C /usr/src/linux-headers-4.15.0-142-generic EXTRA_CFLAGS="-I/home/demo/nkcdriver_V1.0 -DNO_DEBUG -DMODVERSIONS -DNO_PARPORT_SUBSYSTEM -DUSB_SUPPORT -DPCI_SUPPORT -DPCIE_SUPPORT -DISA_SUPPORT -DNO_DONGLE_SUPPORT -DNO_PCCARD_SUPPORT -DNO_NETDEV_SUPPORT -DNO_RT -Wno-date-time" V=0 modules SUBDIRS=/home/demo/nkcdriver_V1.0
```

The.o/.ko file is generated when the compilation is complete.

3. use command: `sudo make install`
4. use command: `sudo modprobe nkcan`

```
demo@demo:~/nkdriver_V1.0$ sudo make install
mkdir -p /usr/local/bin
cp -f udev/nkcan_usb_minor_check.bash /usr/local/bin
chmod 744 /usr/local/bin/nkcan_usb_minor_check.bash
cp -f udev/45-nkcan.rules /etc/udev/rules.d
Info: Copied 45-nkcan.rules to /etc/udev/rules.d.
cp -f udev/blacklist-nk.conf /etc/modprobe.d
chmod 644 /etc/modprobe.d/blacklist-nk.conf
Info: mainline drivers removed and blacklisted in
/etc/modprobe.d/blacklist-nk.conf
udevadm control --reload-rules
- manually installing nkcan under /lib/modules/4.15.0-142-generic/misc...
mkdir -p /lib/modules/4.15.0-142-generic/misc
cp -f nkcan.ko /lib/modules/4.15.0-142-generic/misc/nkcan.ko
- Building dependencies...
depmod 4.15.0-142-generic
demo@demo:~/nkdriver_V1.0$ sudo modprobe nkcan
```

5. After the driver is installed, check whether the driver is correctly installed , use command: `lsmod`.

```
demo@demo:~/nkdriver_V1.0$ lsmod
Module              Size  Used by
nkcan                151552  0
nls_iso8859_1        16384  0
uas                  24576  0
usb_storage          69632  1 uas
can_dev              28672  1 nkcan
snd_soc_skl           86016  0
intel_rapl            20480  0
snd_soc_skl_ipc       65536  1 snd_soc_skl
snd_hda_ext_core      24576  1 snd_soc_skl
snd_soc_sst_dsp       32768  1 snd_soc_skl_ipc
snd_soc_sst_ipc       16384  1 snd_soc_skl_ipc
snd_soc_acpi          16384  1 snd_soc_skl
intel_telemetry_pltdrv 20480  0
snd_soc_core          241664  1 snd_soc_skl
intel_punit_ipc       16384  1 intel_telemetry_pltdrv
snd_hda_codec_hdmi    49152  1
snd_compress          20480  1 snd_soc_core
ac97_bus              16384  1 snd_soc_core
intel_telemetry_core   16384  1 intel_telemetry_pltdrv
intel_pmc_ipc          20480  1 intel_telemetry_pltdrv
snd_pcm_dmaengine      16384  1 snd_soc_core
snd_hda_intel         45056  2
```

6. View CAN devices, use command: `ifconfig -a`

```
demo@demo:~/nkdriver_V1.0$ ifconfig -a
can0                Link encap:未指定  硬件地址 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
NOARP MTU:16  跃点数:1
接收数据包:0 错误:0 丢弃:0 过载:0 帧数:0
发送数据包:0 错误:0 丢弃:0 过载:0 载波:0
碰撞:0 发送队列长度:10
接收字节:0 (0.0 B) 发送字节:0 (0.0 B)

can1                Link encap:未指定  硬件地址 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
NOARP MTU:16  跃点数:1
接收数据包:0 错误:0 丢弃:0 过载:0 帧数:0
发送数据包:0 错误:0 丢弃:0 过载:0 载波:0
碰撞:0 发送队列长度:10
接收字节:0 (0.0 B) 发送字节:0 (0.0 B)

enp1s0              Link encap:以太网  硬件地址 00:60:17:10:00:d9
UP BROADCAST MULTICAST MTU:1500 跃点数:1
接收数据包:0 错误:0 丢弃:0 过载:0 帧数:0
发送数据包:0 错误:0 丢弃:0 过载:0 载波:0
碰撞:0 发送队列长度:1000
接收字节:0 (0.0 B) 发送字节:0 (0.0 B)
```

After the preceding steps are complete, the driver installation is complete.

3.1.2. Uninstall driver

1. use command: `sudo make install`

```
demo@demo:~/nkdriver_V1.0$ sudo make uninstall
rmmod nkcan 2> /dev/null || true
if [ -f /lib/modules/4.15.0-142-generic/misc/nkcan.ko ]; then\
rm -f /lib/modules/4.15.0-142-generic/misc/nkcan.ko;\
depmod 4.15.0-142-generic;\
fi
rm -f /usr/src/nk-linux-driver-8.15.2 \
/tmp/nk-linux-driver-8.15.2-dkms.stderr
rm -rf /var/lib/dkms/nk-linux-driver
rm -f /usr/include/nkcan.h \
/usr/include/nkcanfd.h \
/usr/local/bin/nkcan_make_devices \
/usr/local/bin/lsnkcan \
/usr/local/bin/nkcanosdiag.sh \
/etc/udev/rules.d/45-nkcan.rules \
/usr/local/bin/nkcan_usb_minor_check.bash \
/etc/modprobe.d/blacklist-nk.conf \
/etc/modprobe.d/nkcan.conf
```

2. When the driver is uninstalled, you can check whether the driver is uninstalled, use command: `lsmod`

```
demo@demo:~/nkdriver_V1.0$ lsmod
Module                  Size  Used by
nls_iso8859_1           16384  0
uas                     24576  0
usb_storage             69632  1 uas
can_dev                 28672  0
snd_soc_skl             86016  0
intel_rapl              20480  0
snd_soc_skl_ipc         65536  1 snd_soc_skl
snd_hda_ext_core        24576  1 snd_soc_skl
snd_soc_sst_dsp         32768  1 snd_soc_skl_ipc
snd_soc_sst_ipc         16384  1 snd_soc_skl_ipc
snd_soc_acpi            16384  1 snd_soc_skl
intel_telemetry_pltdrv  20480  0
snd_soc_core            241664  1 snd_soc_skl
intel_punit_ipc         16384  1 intel_telemetry_pltdrv
snd_hda_codec_hdmi      49152  1
snd_compress            20480  1 snd_soc_core
ac97_bus                16384  1 snd_soc_core
intel_telemetry_core    16384  1 intel_telemetry_pltdrv
intel_pmc_ipc           20480  1 intel_telemetry_pltdrv
snd_pcm_dmaengine       16384  1 snd_soc_core
snd_hda_intel           45056  4
```

You can see that there is no nkcan driver.

3. View CAN devices, use command: `ifconfig -a`

```
demo@demo:~/nkdriver_V1.0$ ifconfig -a
enp1s0  Link encap:以太网 硬件地址 00:60:17:10:00:d9
        UP BROADCAST MULTICAST  MTU:1500  跃点数:1
        接收数据包:0 错误:0 丢弃:0 过载:0 帧数:0
        发送数据包:0 错误:0 丢弃:0 过载:0 载波:0
        碰撞:0 发送队列长度:1000
        接收字节:0 (0.0 B) 发送字节:0 (0.0 B)
        Memory:91200000-9121ffff
enp2s0  Link encap:以太网 硬件地址 00:60:17:10:00:da
        UP BROADCAST MULTICAST  MTU:1500  跃点数:1
        接收数据包:0 错误:0 丢弃:0 过载:0 帧数:0
        发送数据包:0 错误:0 丢弃:0 过载:0 载波:0
        碰撞:0 发送队列长度:1000
        接收字节:0 (0.0 B) 发送字节:0 (0.0 B)
        Memory:91100000-9111ffff
lo      Link encap:本地环回
        inet 地址:127.0.0.1 掩码:255.0.0.0
        inet6 地址: ::1/128 Scope:Host
        UP LOOPBACK RUNNING  MTU:65536  跃点数:1
        接收数据包:29748 错误:0 丢弃:0 过载:0 帧数:0
        发送数据包:29748 错误:0 丢弃:0 过载:0 载波:0
        碰撞:0 发送队列长度:1000
```

You can see that no CAN device exists.

3.1.3. Configure the CAN bus

The following example configures "can0" to a bit rate of 500000.

1.View CAN devices

use command:

ifconfig -a

```
demo@demo:~/nkcandriver_V1.0$ ifconfig -a
enp1s0    Link encap:以太网 硬件地址 00:60:17:10:00:d9
          UP BROADCAST MULTICAST  MTU:1500  跃点数:1
          接收数据包:0  错误:0  丢弃:0  过载:0  帧数:0
          发送数据包:0  错误:0  丢弃:0  过载:0  载波:0
          碰撞:0  发送队列长度:1000
          接收字节:0 (0.0 B)  发送字节:0 (0.0 B)
          Memory:91200000-9121ffff

enp2s0    Link encap:以太网 硬件地址 00:60:17:10:00:da
          UP BROADCAST MULTICAST  MTU:1500  跃点数:1
          接收数据包:0  错误:0  丢弃:0  过载:0  帧数:0
          发送数据包:0  错误:0  丢弃:0  过载:0  载波:0
          碰撞:0  发送队列长度:1000
          接收字节:0 (0.0 B)  发送字节:0 (0.0 B)
          Memory:91100000-9111ffff

lo        Link encap:本地环回
          inet 地址:127.0.0.1  掩码:255.0.0.0
          inet6 地址: ::1/128  Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  跃点数:1
          接收数据包:29748  错误:0  丢弃:0  过载:0  帧数:0
          发送数据包:29748  错误:0  丢弃:0  过载:0  载波:0
          碰撞:0  发送队列长度:1000
```

can0 and can1 in the figure above are our Can card devices.

2.Set the baud rate of the can device,use command:

sudo ip link set can0 type can bitrate 500000

3.Start the can device,use command:

sudo ip link set up can0

4.Turn off the can device,use command:

sudo ip link set down can0

3.1.4. Use CAN-utils to operate the CAN bus

When the can device is enabled, a number of utilities will be available for can communication. First we install can-utils by entering the following command into the command terminal:

sudo apt-get install can-utils

➤ candump

candump displays the received can messages in real time.

To display all can messages on the device can0 in real time, use command:

candump can0

Received data:

can0 456 [8] 12 34 56 78 98 76 54 32

This instruction will receive a message on interface can0 with ID 0x456, data length 8, and data [0x12, 0x34, 0x56, 0x78, 0x98, 0x76, 0x54, 0x32].

➤ **cansend**

cansend CAN send a single CAN frame onto the bus. You will have to specify the device, identifier, and bytes of data to send. use command:

```
cansend can0 123#1122334455667788
```

This instruction will send a message on the can0 interface with ID 0x123 and data [0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88]. Note that this data is given in hexadecimal.

➤ **cangen**

cangen CAN generate random CAN data, which is useful for testing, for more usage information, use command:

```
cangen -help
```

➤ **cansniffer**

cansniffer displays CAN messages received on the bus and filters out frames with unchanged data. This is very useful for reverse-engineering CAN bus systems. For more information, use command:

```
cansniffer -help
```

3.1.5. Routine

The CAN bus card supports the socket-can interface for Linux systems. The following code snippet is a working example of the SocketCAN API, which uses the original interface to send and receive packets of CAN1 and CAN2:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <net/if.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <linux/can.h>
#include <linux/can/raw.h>
#include <string.h>
#define commde "echo 123456 |sudo -S ip link set can0 type can bitrate 500000"
#define up "sudo ip link set up can0"
#define commde1 "echo |sudo -S ip link set can1 type can bitrate 500000"
#define up1 "sudo ip link set up can1"
```

```
int main(void)
{
    system(commde);
    system(up);
    system(commde1);
    system(up1);
    int nbytes =0 ;
    struct sockaddr_can sockCan0,sockCan1;
    const char *ifname = "can0";
    const char *ifname1 = "can1";
    socklen_t len = 0;
    struct ifreq ifr;
    struct can_frame frame[2] = {{0},{0}};
    int s = socket(PF_CAN,SOCK_RAW,CAN_RAW);
    struct sockaddr_can addr;
    addr.can_family = AF_CAN;
    addr.can_ifindex = 0;
    bind(s,(struct sockaddr *)&addr,sizeof(addr));

    strcpy(ifr.ifr_name,ifname);
    ioctl(s,SIOCGIFINDEX,&ifr);
    sockCan0.can_family = AF_CAN;
    sockCan0.can_ifindex = ifr.ifr_ifindex;

    strcpy(ifr.ifr_name,ifname1);
    ioctl(s,SIOCGIFINDEX,&ifr);
    sockCan1.can_family = AF_CAN;
    sockCan1.can_ifindex = ifr.ifr_ifindex;
    setsockopt(s,SOL_CAN_RAW,CAN_RAW_FILTER,NULL,0);
    struct can_filter rfilter[2];
    /*rfilter[0].can_id = 0x123;
    rfilter[0].can_mask = CAN_EFF_MASK;
    rfilter[1].can_id = 0x124;
    rfilter[1].can_mask = CAN_EFF_MASK;*/
    setsockopt(s,SOL_CAN_RAW,CAN_RAW_FILTER,rfilter,sizeof(rfilter));

    frame[0].can_id = 3;
    frame[0].can_dlc = 8;
    frame[0].data[0] = 0x40;
    frame[0].data[1] = 0x01;
    frame[0].data[2] = 0x10;
    frame[0].data[3] = 0x00;
    frame[0].data[4] = 0x00;
```

```
frame[0].data[5] = 0x00;
frame[0].data[6] = 0x00;
frame[0].data[7] = 0x00;

frame[1].can_id = 123;
frame[1].can_dlc = 8;
frame[1].data[0] = 0x40;
frame[1].data[1] = 0x01;
frame[1].data[2] = 0x10;
frame[1].data[3] = 0x00;
frame[1].data[4] = 0x00;
frame[1].data[5] = 0x00;
frame[1].data[6] = 0x00;
frame[1].data[7] = 0x00;
for(int c=0; c<300; c++)
{
    nbytes = sendto(s,&frame[0],sizeof(struct can_frame),0,(struct
sockaddr*)&sockCan0,sizeof(sockCan0));
    frame[0].can_id++;
    usleep(2000);
    printf("%s Wrote %d bytes\n",ifname,nbytes);
    if(nbytes < sizeof(struct can_frame))
    {
        printf("Error:sendto can0\n");
    }
    nbytes = 0;
    nbytes = sendto(s,&frame[1],sizeof(struct can_frame),0,(struct
sockaddr*)&sockCan1,sizeof(sockCan1));
    frame[1].can_id++;
    usleep(1000);
    printf("%s Wrote %d bytes\n",ifname1,nbytes);
    if(nbytes < sizeof(struct can_frame))
    {
        printf("Error:sendto can1\n");
    }
}
while(1)
{
    for(int j=0; j<2; j++)
    {
        recvfrom(s,&frame[j],sizeof(struct can_frame),0,(struct
sockaddr*)&addr,&len);
        printf("Receive OK\n");
    }
}
```

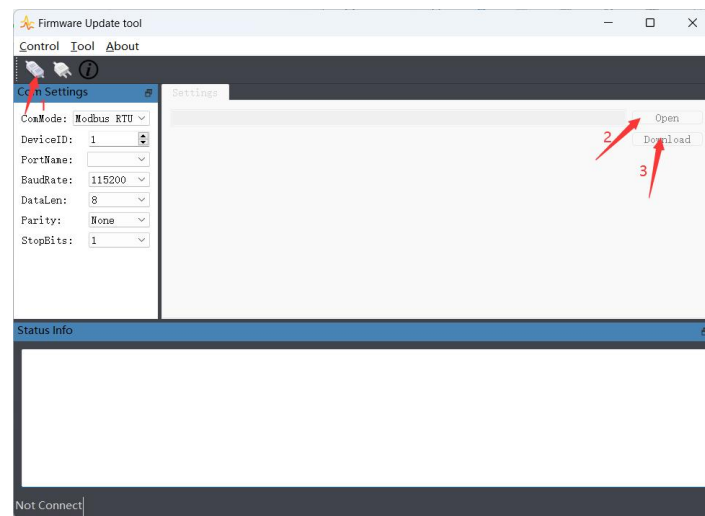
```
        printf("ID=0x%X          DLC=%d          Data:
",frame[j].can_id,frame[j].can_dlc);
        for(int i = 0; i<frame[j].can_dlc; ++i)
        {
            printf("0x%02X ",frame[j].data[i]);
        }
        printf("\n");
    }
}
close(s);
return 0;
```

4. upgrade

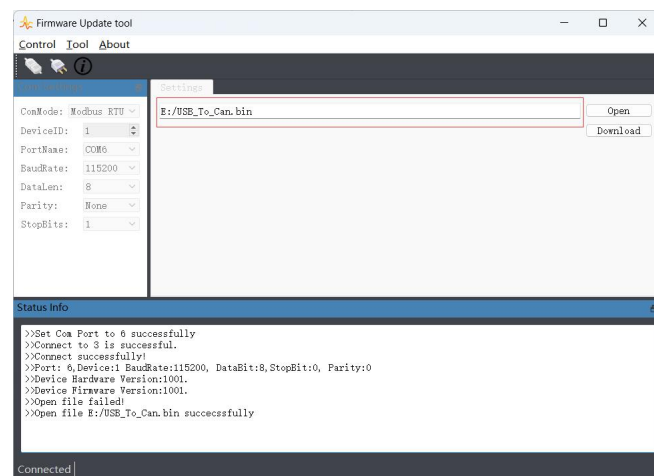
4.1. Upgrade the software

Use the upgrade software NKMcuFwUpdateTool delivered by the company , Supports windows10, windows11.

method of application:



Upgrade the file to a bin file.



4.2. Upgrade Procedure

When CAN0 sends an extension frame with ID 0x1fffffe and data 0xFFFFFfee and sends this data, the upgrade starts. The upgrade needs to re-insert the USB to identify the serial port. All files required for the upgrade are bin files. After upgrading using the upgrade software, re-plug the USB to complete the upgrade.