ADVANTECH
Enabling an Intelligent Planet

Advantech AE Technical Share Document

Date	2015 / 12 / 04	SR#	1-2307574301
Category	□ FAQ■ SOP	Related OS	Linux Operation System
Abstract	How to test CAN port with Advantech CAN card on Linux system?		
Keyword	CAN Communication cards / CAN2.0 / CAN2.0A / CAN2.0B / PCI CAN Communication cards / PC/104 CAN Communication Modules / PCI/104 CAN Communication Modules / CAN Monitor / CAN Test		
Related Product	PCI-1680U / PCI-1682U / PCL-841 / PCM-3680 / PCM-3680I		

■ **Problem Description:**

This is a SOP document to describe how to test CAN port with Advantech CAN bus communication card on Linux OS. By connecting the wires correctly, those two CAN port testing can help user to verify the operation of CAN ports by using Advantech Linux CAN example.

■ **Brief Solution :**

Please refer this steps to test CAN port.

1. **Please insert CAN card in your PC first. And put Advantech CAN Linux driver in your OS too.**
2. **Connect with Advantech two CAN port, then connect with each CAN_L, CAN_H pin, as Figure 1: (Port A's pin2 to Port B's pin2; Port A's pin7 to Port B's pin7)**

Pin	Signal
2	CAN_L
3	GND
7	CAN_H

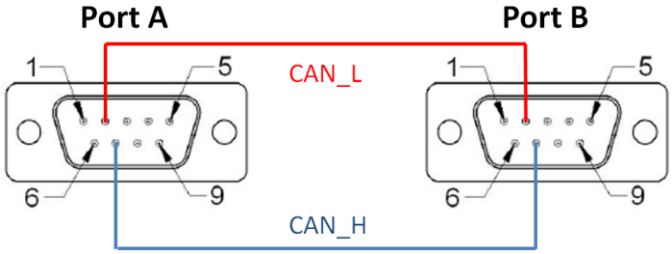

Figure 1. Connect with CAN_L & CAN_H of Port A & Port B.

3. **Open one terminal: into example folder, type #make for compiling all example files, as Figure 2.**

```
root@ubuntu:/home/ubuntu/Desktop/advcan_source_v2.17/advcan_source_v2.17# cd examples/
root@ubuntu:/home/ubuntu/Desktop/advcan_source_v2.17/advcan_source_v2.17/examples# ls
acceptance.c  Makefile           receive-select.c  showstat.c        transmit-nonblock.c
baud.c        receive-block.c    selfreception.c   singlefilter.c    transmit-select.c
can4linux.h   receive-nonblock.c send-ioctl.c      transmit-block.c
root@ubuntu:/home/ubuntu/Desktop/advcan_source_v2.17/advcan_source_v2.17/examples# make
gcc -I../can4linux -o baud baud.c -lrt
gcc -I../can4linux -o acceptance acceptance.c -lrt
gcc -I../can4linux -o receive-block receive-block.c -lrt
gcc -I../can4linux -o transmit-block transmit-block.c -lrt
gcc -I../can4linux -o receive-nonblock receive-nonblock.c -lrt
gcc -I../can4linux -o transmit-nonblock transmit-nonblock.c -lrt
gcc -I../can4linux -o receive-select receive-select.c -lrt
gcc -I../can4linux -o send-ioctl send-ioctl.c -lrt
send-ioctl.c: In function 'main':
send-ioctl.c:122:7: warning: format '%d' expects argument of type 'int', but argument 2 has type 'long unsig
ned int' [-Wformat=]
       printf(" using canmsg_t with %d bytes\n", sizeof(canmsg_t));
       ^
gcc -I../can4linux -o selfreception selfreception.c -lrt
gcc -I../can4linux -o singlefilter singlefilter.c -lrt
gcc -I../can4linux -o transmit-select transmit-select.c -lrt
gcc -I../can4linux -o showstat showstat.c -lrt
root@ubuntu:/home/ubuntu/Desktop/advcan_source_v2.17/advcan_source_v2.17/examples#
```

Figure 2. Type #make for CAN example (.c file)

4. **After #make successful, please open another new terminal, and into example folder too.**
   One terminal run "transmit-select" example. (Type: # ./transmit-select can0)
   P.S "can0" means transmit CAN data from can PortA.

```
root@ubuntu:/home/ubuntu/Desktop/advcan_source_v2.17/advcan_source_v2.17/examples# clear

root@ubuntu:/home/ubuntu/Desktop/advcan_source_v2.17/advcan_source_v2.17/examples# ls
acceptance     Makefile          receive-select    send-ioctl.c    transmit-block      transmit-select.c
acceptance.c   receive-block     receive-select.c  showstat        transmit-block.c
baud           receive-block.c   selfreception     showstat.c      transmit-nonblock
baud.c         receive-nonblock  selfreception.c   singlefilter    transmit-nonblock.c
can4linux.h    receive-nonblock.c send-ioctl       singlefilter.c  transmit-select
root@ubuntu:/home/ubuntu/Desktop/advcan_source_v2.17/advcan_source_v2.17/examples# ./transmit-select can0
using CAN device /dev/can0, got can_fd  3
write down 5
got one message Error 16
 - ERROR PASSIVE
```

   Another terminal run "receive-select" example. (Type: # ./receive-select can1)
   P.S "can1" means receive CAN data from can PortB.

```
ubuntu@ubuntu:~$ sudo su
root@ubuntu:/home/ubuntu# cd Desktop/advcan_source_v2.17/advcan_source_v2.17/examples/
root@ubuntu:/home/ubuntu/Desktop/advcan_source_v2.17/advcan_source_v2.17/examples# ls
acceptance     Makefile          receive-select    send-ioctl.c    transmit-block      transmit-select.c
acceptance.c   receive-block     receive-select.c  showstat        transmit-block.c
baud           receive-block.c   selfreception     showstat.c      transmit-nonblock
baud.c         receive-nonblock  selfreception.c   singlefilter    transmit-nonblock.c
can4linux.h    receive-nonblock.c send-ioctl       singlefilter.c  transmit-select
root@ubuntu:/home/ubuntu/Desktop/advcan_source_v2.17/advcan_source_v2.17/examples# ./receive-select can1
```

5. **If everything works fine, two terminal will be like this:**

   can0 (PortA) -> transmit CAN data

```
write down 11565
write down 11570
write down 11575
write down 11580
write down 11585
write down 11590
write down 11595
write down 11600
write down 11605
write down 11610
write down 11615
write down 11620
write down 11625
write down 11630
write down 11635
write down 11640
write down 11645
write down 11650
```

   can1 (PortB) -> receive CAN data

```
count: 9900
count: 10000
count: 10100
count: 10200
count: 10300
count: 10400
count: 10500
count: 10600
count: 10700
count: 10800
count: 10900
count: 11000
count: 11100
count: 11200
count: 11300
count: 11400
count: 11500
count: 11600
```

6. **Then please change can port setting; can1 to transmit data, can0 to receive data. Please refer it.**
   One terminal run "receive-select" example. (Type: # ./ receive-select can0)

```
root@ubuntu:/home/ubuntu/Desktop/advcan_source_v2.17/advcan_source_v2.17/examples# ls
acceptance     logfile.txt       receive-nonblock.c  send-ioctl     singlefilter.c      transmit-select
acceptance.c   Makefile          receive-select      send-ioctl.c   transmit-block      transmit-select.c
baud           receive-block     receive-select.c    showstat       transmit-block.c
baud.c         receive-block.c   selfreception       showstat.c     transmit-nonblock
can4linux.h    receive-nonblock  selfreception.c     singlefilter   transmit-nonblock.c
root@ubuntu:/home/ubuntu/Desktop/advcan_source_v2.17/advcan_source_v2.17/examples# ./receive-select can0
```

   Another terminal run "transmit-select" example. (Type: # ./ transmit-select can1)

```
root@ubuntu:/home/ubuntu/Desktop/advcan_source_v2.17/advcan_source_v2.17/examples# ls
acceptance     logfile.txt       receive-nonblock.c  send-ioctl     singlefilter.c      transmit-select
acceptance.c   Makefile          receive-select      send-ioctl.c   transmit-block      transmit-select.c
baud           receive-block     receive-select.c    showstat       transmit-block.c
baud.c         receive-block.c   selfreception       showstat.c     transmit-nonblock
can4linux.h    receive-nonblock  selfreception.c     singlefilter   transmit-nonblock.c
root@ubuntu:/home/ubuntu/Desktop/advcan_source_v2.17/advcan_source_v2.17/examples# ./transmit-select can1
using CAN device /dev/can1, got can_fd  3
write down 5
```

7. **As above test result, if both way can work very well with transmit & receive example, that mean these CAN card function is fine.**