



ExPC Fintek PCIe CAN Series

EGPC-B4S1

EGPC-B2S1

EGPC-B1S1

EMPC-B2S1

User Manual

Rev 1.3

Copyright Information

2005-2023 ©Innodisk Corporation. All Rights Reserved

Innodisk™ is trademark or registered trademark of Innodisk Corporation.

This document is subject to change and revision without notice. No part of this document may be reproduced in any form by any photographic, electronic, mechanical or other means, or used in any information storage and retrieval system, without prior written permission from Innodisk Corporation.

All other product and brand names in this document are trademarks or registered trademarks of their respective owners.

版權說明

2005-2023 ©宜鼎國際股份有限公司

Innodisk™ 是宜鼎國際股份有限公司之註冊商標。

本文件得不經通知即更改或修訂。本文件中出現任何文字敘述、文件格式、圖形、照片、方法及過程等內容，除另特別註明，版權均屬宜鼎國際股份有限公司所有，受到相關之智慧財產權保護法之保障。任何個人、法人或機構未經宜鼎國際股份有限公司的書面（包括電子文件）授權，不得以任何形式複製或引用本文件之全部或片段。

其他出現在本文件的品牌或產品乃歸屬原公司所有之商標或註冊。

Revision History

| Revision | Date | Description |
|----------|------------|--|
| 1.0 | 2020/06/18 | Initial Release |
| 1.1 | 2022/5/16 | Merge all Fintek CAN user manual |
| 1.2 | 2022/8/26 | Add Software API Description |
| 1.3 | 2023/4/27 | Correct API function name and add new function |
| | | |

Table of Contents

| | |
|--|------------|
| Revision History | ii |
| Table of Contents | iii |
| 1. Hardware Installation | 1 |
| 1.1. EGPC-B4S1 | 1 |
| 1.2. EGPC-B1S1 | 2 |
| 1.3. EMPC-B2S1 | 2 |
| 2. Windows OS | 2 |
| 2.1. Driver Installation | 2 |
| 2.2. Loop Back Test Program | 4 |
| 2.3. GUI CAN bus Tool | 8 |
| 2.4. Software API | 10 |
| 2.4.1. EGPCOpen | 10 |
| 2.4.2. EGPCOpenInactive | 10 |
| 2.4.3. EGPCInitCAN | 11 |
| 2.4.4. EGPCClose | 11 |
| 2.4.5. EGPCSetBaudRate | 11 |
| 2.4.6. EGPCGetBaudRate | 12 |
| 2.4.7. EGPCSetFilter | 12 |
| 2.4.8. EGPCClearFilter | 12 |
| 2.4.9. EGPCReceive | 13 |
| 2.4.10. EGPCSend | 13 |
| 2.4.11. EGPCReset | 14 |
| 2.4.12. EGPCShowVer | 14 |
| 2.4.13. EGPCWarnStr | 15 |
| 2.4.14. Error Codes | 15 |
| 3. Linux OS | 16 |
| 3.1. SocketCAN Driver Installation | 16 |
| 3.2. CAN-utils | 20 |
| 3.3. Loop Back Test Program | 21 |
| Contact us | 23 |

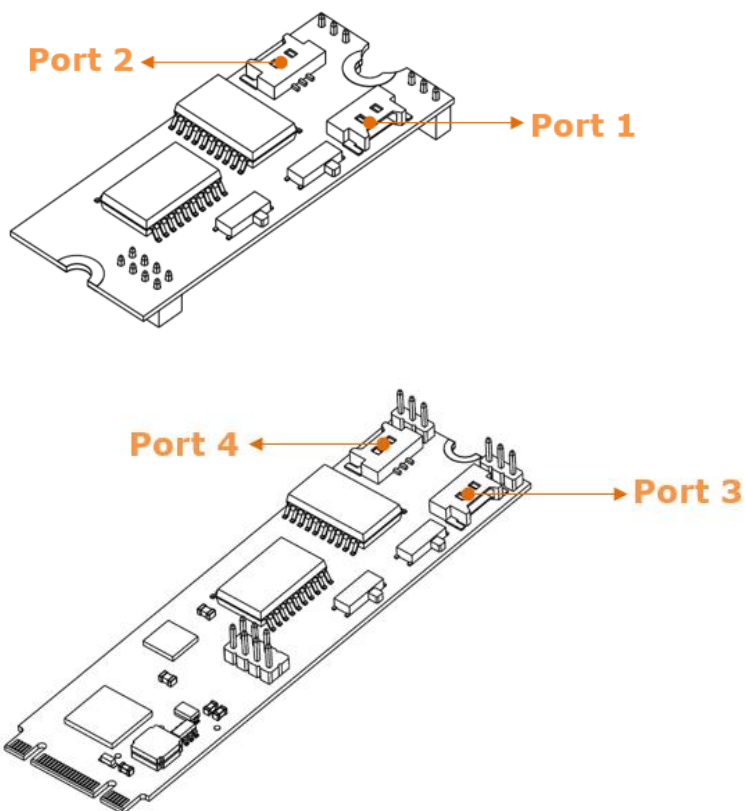
1. Hardware Installation

1.1. EGPC-B4S1

Install the module to M.2 B-M key slot which has PCIe interface.



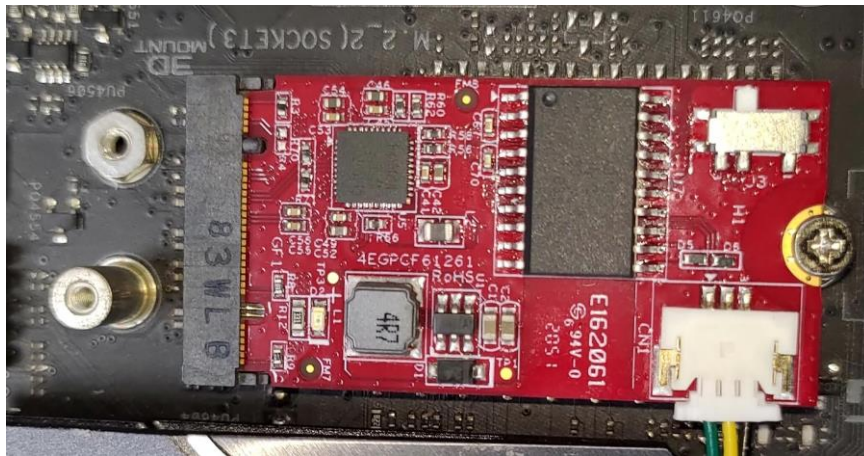
The following picture shows the port order in majority platform.



NOTE: Some platforms may show opposite port order in Windows.

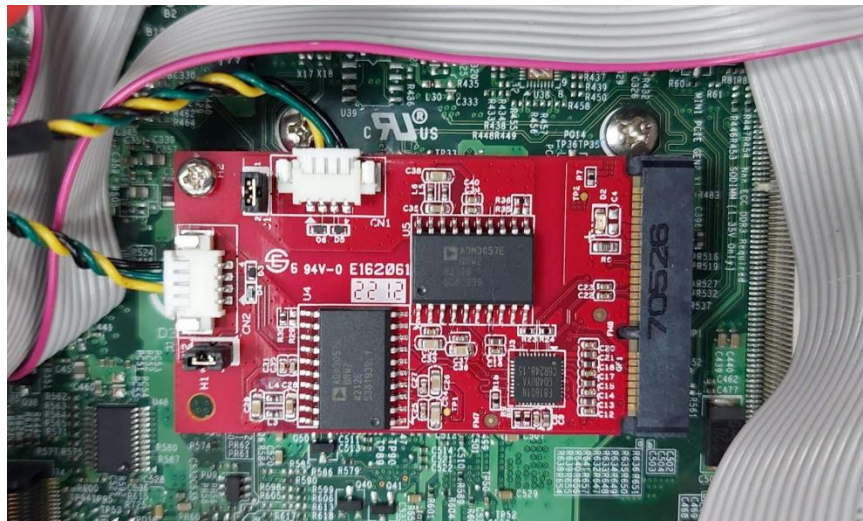
1.2. EGPC-B1S1

Install the module to M.2 B-M key slot which has PCIe interface.



1.3. EMPC-B2S1

Install the module to mPCIe slot which has PCIe interface.



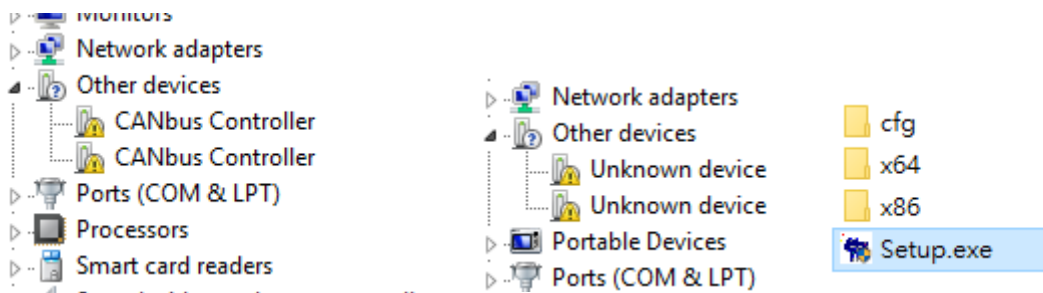
2. Windows OS

The following uses EGPC-B4S1 for driver installation and test as the example.

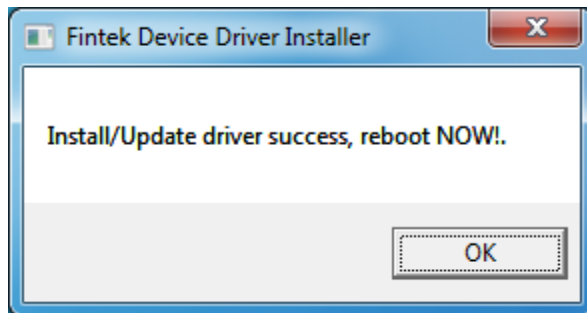
2.1. Driver Installation

The device named “CAN bus Controller” or “Unknown device” can be found in “Device Manager”.

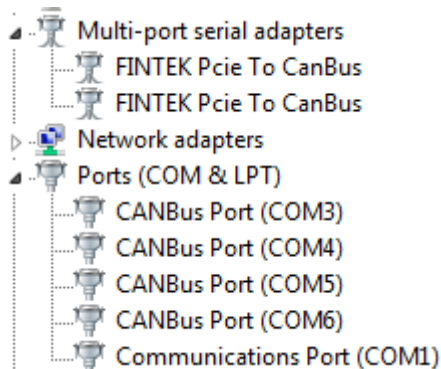
Run the driver package as administrator.



After installing, please reboot.



After reboot, CAN bus ports can be found in the device manager.



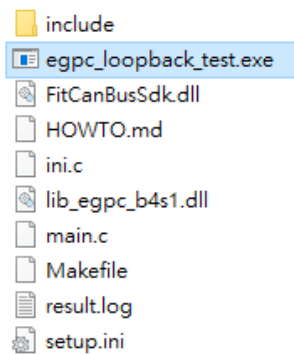
NOTE: Drivers for Windows 64bit need to be signed with digital certificate. The driver is signed with SHA-2 certificate

Windows 7 must be installed the hotfix **KB3033929** to support SHA-2 code signing.

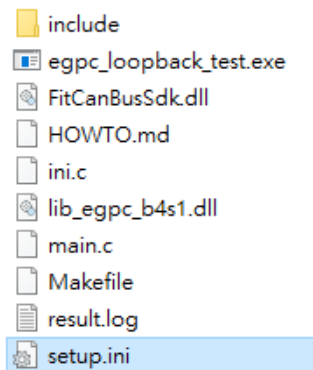
<https://technet.microsoft.com/en-ie/library/security/3033929>

2.2. Loop Back Test Program

We provide a loop back test program with source code in Windows to verify the module.



The test program can be configured by edit “setup.ini”. In majority cases, just keep the default setting.



If your CAN card is 4 port version, keep COM_Qty at 4, don't care COM port number setting.

```

;
; some setting for EGPC-B4S1 loopback test
;
[test]
COM_Qty=4                ; support: 2 port, 4 port
COM_1th=COM4             ; if COM_Qty=4, it is a don't care condition
COM_2nd=COM5             ; if COM_Qty=4, it is a don't care condition

```

If your CAN card is 2 port version, please modify COM_Qty to 2 and specify the COM port number.

```

;
; some setting for EGPC-B4S1 loopback test
;
[test]
COM_Qty=2                ; support: 2 port, 4 port
COM_1th=COM4             ; if COM_Qty=4, it is a don't care condition
COM_2nd=COM5             ; if COM_Qty=4, it is a don't care condition

```


Please connect any two ports with each other by using an adapter (MINI GENDER CHANGER) and connect the other two as well. Then run the “EGPC_B4S1_Sample.exe” directly. The test program will detect which ports are connected with each other then run the test automatically.



```
C:\Users\test\Desktop\Share\testing\loopback\egpc_loopback_test.exe

Config
-----
baudrate: 1000000
sleep_interval: 0
test_time: 5
data_length: 4
rtr: disable
id_type: 29Bit
-----

===== Open EGPC_B4S1 =====

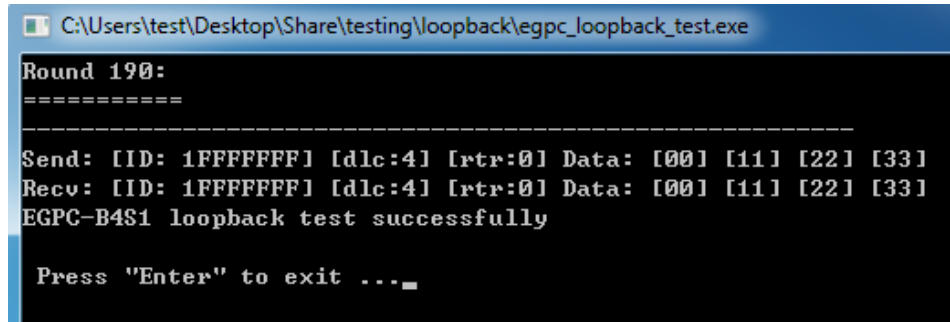
comport: COM3; BaudRate: 1000000 !
=====
comport: COM4; BaudRate: 1000000 !
=====
comport: COM5; BaudRate: 1000000 !
=====
comport: COM6; BaudRate: 1000000 !
=====

===== Scan connecting =====

COM3 <-> COM4
COM4 <-> COM3
COM5 <-> COM6
COM6 <-> COM5
Start to test loopback: COM3 COM4
Start to test loopback: COM5 COM6
-
```

When the program is running, for example, CAN1 sends a frame to CAN2, after CAN2 receives the frame CAN2 will check if the frame is correct or not. Then turn to CAN2 sends and CAN1 receives.

If the received CAN port doesn't receive the frame or the received frame is incorrect, the program will terminate and show the result is failed.



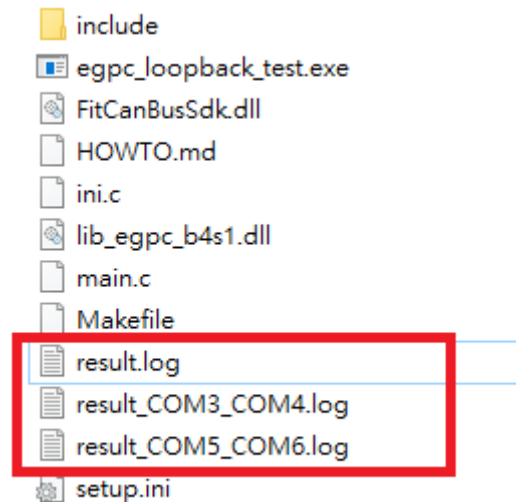
```

C:\Users\test\Desktop\Share\testing\loopback\egpc_loopback_test.exe
Round 190:
=====
Send: [ID: 1FFFFFFF] [dlc:4] [rtr:0] Data: [00] [11] [22] [33]
Recv: [ID: 1FFFFFFF] [dlc:4] [rtr:0] Data: [00] [11] [22] [33]
EGPC-B4S1 loopback test successfully

Press "Enter" to exit ...

```

Test results will be saved to the following logs.



```
===== 2020/06/01 14:57:20 =====
comport: COM3, COM4
baudrate = 1000000
interval = 0 [ms]
test_time = 60 [sec]
EGPC-B4S1 loopback test successfully

===== 2020/06/01 15:00:49 =====
comport: COM3, COM4
baudrate = 1000000
interval = 0 [ms]
test_time = 60 [sec]
EGPC-B4S1 loopback test successfully

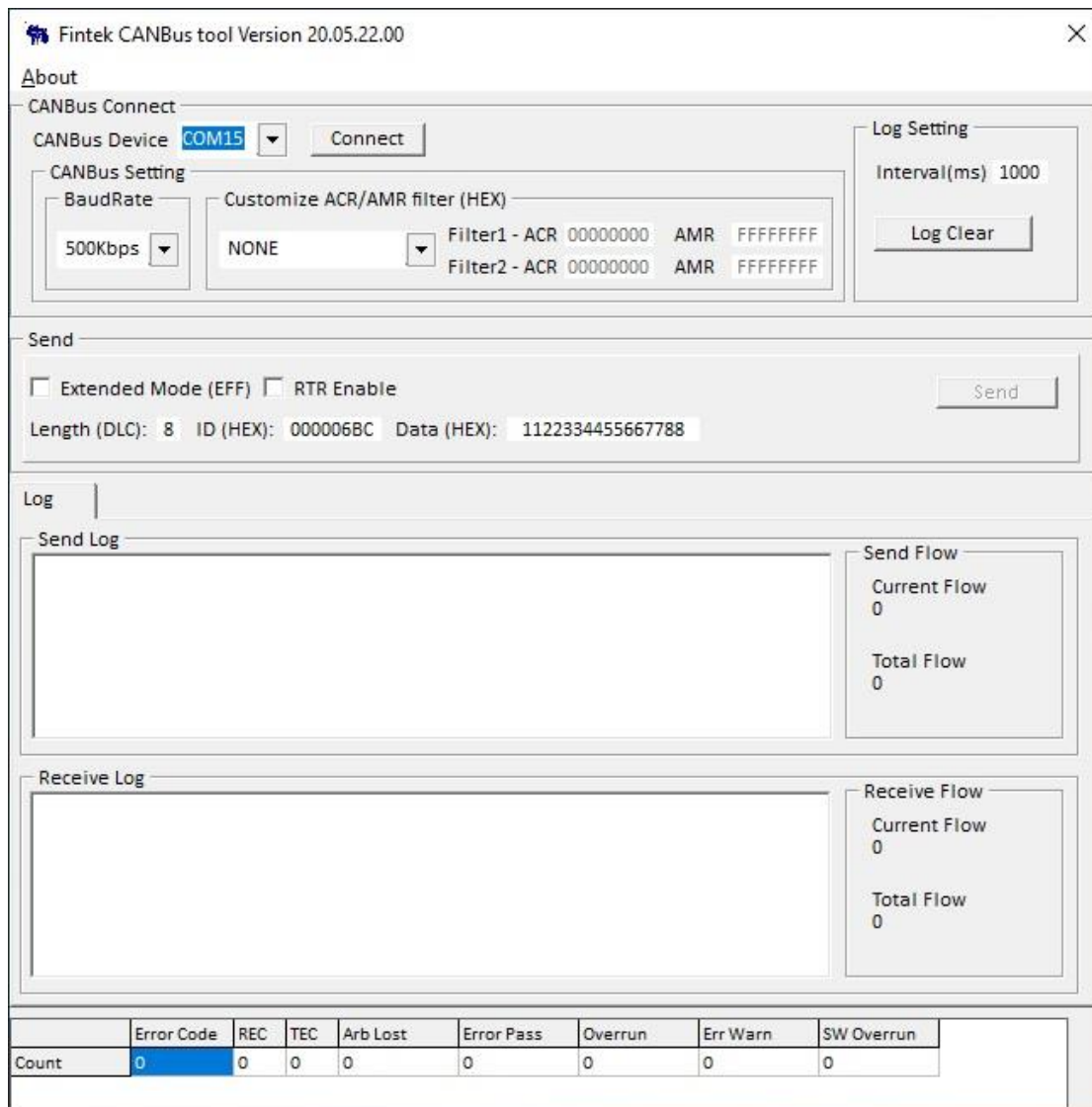
===== 2020/06/01 15:04:09 =====
comport: COM3, COM4
baudrate = 1000000
interval = 0 [ms]
test_time = 60 [sec]
EGPC-B4S1 loopback test successfully

===== 2020/06/01 15:07:29 =====
comport: COM3, COM4
baudrate = 1000000
interval = 0 [ms]
test_time = 60 [sec]
EGPC-B4S1 loopback test successfully

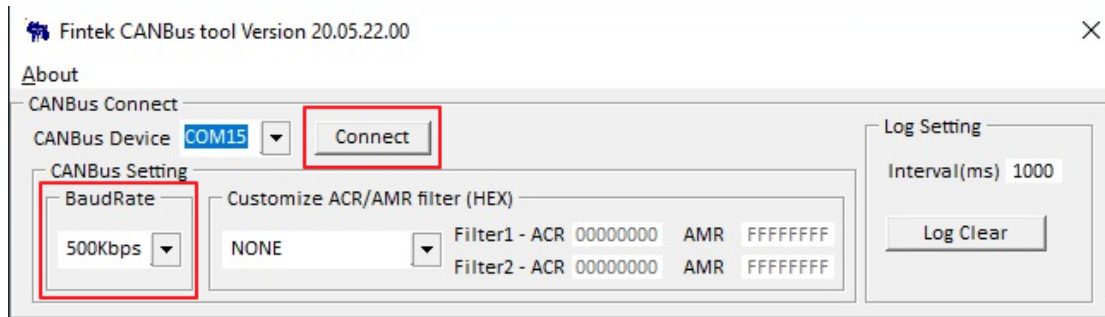
===== 2020/06/01 15:10:49 =====
comport: COM3, COM4
baudrate = 1000000
interval = 0 [ms]
test_time = 60 [sec]
```

2.3. GUI CAN bus Tool

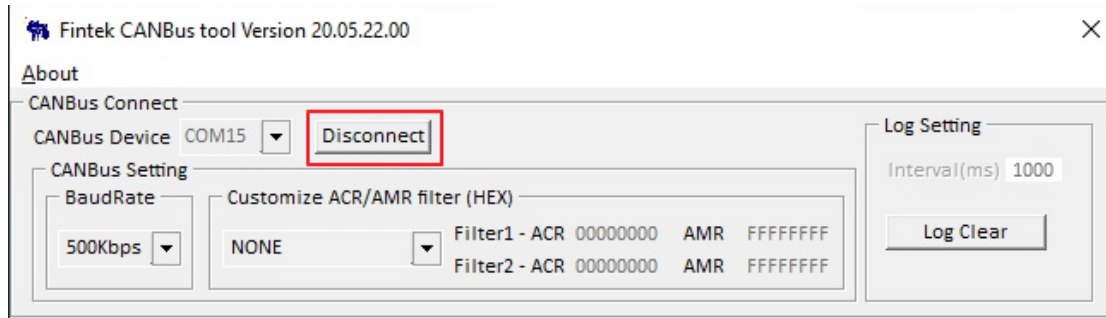
We provide GUI CAN bus tool in Windows to verify the module.



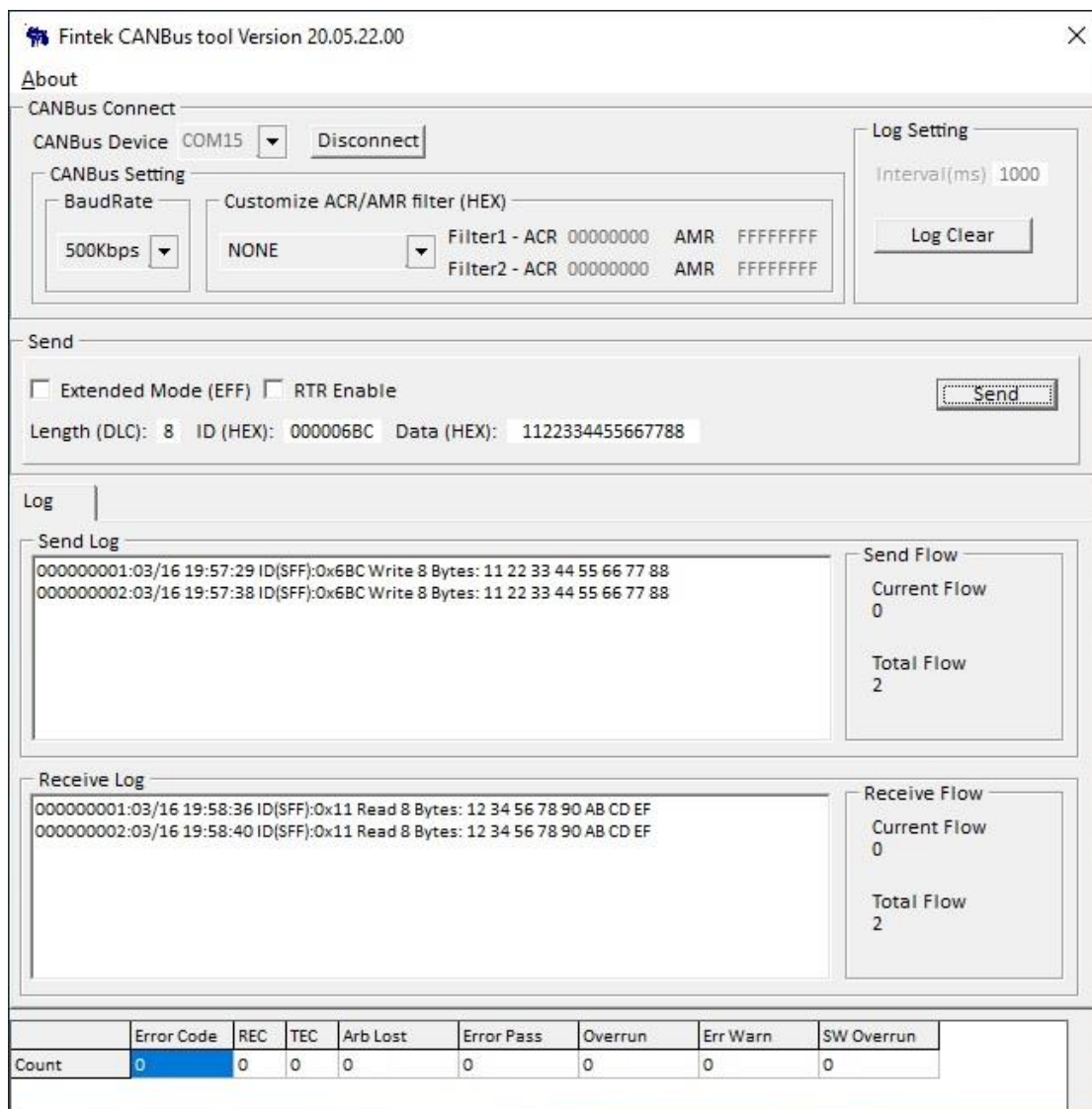
Before testing, please set BaudRate, and then click "Connect" to start CAN.



After successfully starting CAN, the button will change from "Connect" to "Disconnect".



Using the CAN bus tool, you can monitor whether CAN frames are received and sent from the CAN bus network.



2.4. Software API

EGPC API is based on a dynamic library (DLL) in Windows to control ExPC Fintek PCIe CAN series product.

Please refer to the following table for ExPC APIs:

| ID | Function Name |
|------|------------------|
| 4.1 | EGPCOpen |
| 4.2 | EGPCOpenInactive |
| 4.3 | EGPCInitCAN |
| 4.4 | EGPCClose |
| 4.5 | EGPCSetBaudRate |
| 4.6 | EGPCGetBaudRate |
| 4.7 | EGPCSetFilter |
| 4.8 | EGPCClearFilter |
| 4.9 | EGPCReceive |
| 4.10 | EGPCSend |
| 4.11 | EGPCReset |
| 4.12 | EGPCShowVer |
| 4.13 | EGPCWarnStr |

2.4.1. EGPCOpen

long EGPCOpen(char* sComPortNumber)

Function: Open CAN bus virtual COM Port.

Parameters:

sComPortNumber: COM port Number

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. Refer to section **Error Codes** to find more information.

2.4.2. EGPCOpenInactive

long EGPCOpenInactive(char* sComPortNumber)

Function: Open CAN bus virtual COM Port with inactive mode

Parameters:

sComPortNumber: COM port Number

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. Refer to section **Error Codes** to find more information.

2.4.3. EGPCInitCAN

long EGPCInitCAN (char* sComPortNumber, int status)

Function: Set CAN port to active/inactive.

Parameters:

sComPortNumber: COM port Number

status: CAN status value.

inactive: 0 or STS_INACTIVE

active: 1 or STS_ACTIVE

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. Refer to section **Error Codes** to find more information.

2.4.4. EGPCClose

long EGPCClose(char* sComPortNumber)

Function: Close CAN bus virtual COM Port.

Parameters:

sComPortNumber: COM port Number

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. Refer to section **Error Codes** to find more information.

2.4.5. EGPCSetBaudRate

long EGPCSetBaudRate(char* sComPortNumber, DWORD BaudRate)

Function: Set CAN bus port Baud Rate.

Parameters:

sComPortNumber: COM port Number

BaudRate:

1M : 1000000 or EGPC_BAUDRATE_1M

800K : 800000 or EGPC_BAUDRATE_800K

500K : 500000 or EGPC_BAUDRATE_500K

250K : 250000 or EGPC_BAUDRATE_250K

100K : 100000 or EGPC_BAUDRATE_100K

50K : 50000 or EGPC_BAUDRATE_50K

20K : 20000 or EGPC_BAUDRATE_20K

10K : 10000 or EGPC_BAUDRATE_10K

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. Refer to section **Error Codes** to find more information.

2.4.6. EGPCGetBaudRate

long EGPCGetBaudRate(char* sComPortNumber, DWORD* BaudRate)

Function: Get CAN bus port Baud Rate.

Parameters:

sComPortNumber: COM port Number

***BaudRate:**

1M:1000000

800K:800000

500K:500000

250K:250000

100K:100000

50K:50000

20K:20000

10K:10000

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. Refer to section **Error Codes** to find more information.

2.4.7. EGPCSetFilter

long EGPCSetFilter(char* sComPortNumber, DWORD pattern, DWORD mask)

Function: Set CAN bus port Filter. Default: pattern:0; mask:0, all frame will be received

Parameters:

sComPortNumber: COM port Number

pattern: Specified CAN ID pattern to filter

mask: Specified the mask for filter

Return Value:

If the function succeeds, the return value is filter total count - 1 (0-14, MAX: 15). If the function fails, the return value is error code generated by the API. Refer to section **Error Codes** to find more information.

2.4.8. EGPCClearFilter

long EGPCClearFilter(char* sComPortNumber)

Function: Clear CAN bus port Filter

Parameters:

sComPortNumber: COM port Number

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. Refer to section **Error Codes** to find more information.

2.4.9. EGPCReceive

long EGPCReceive(char* sComPortNumber, CanFrameInforProc callback)

Function: Receive CAN bus frame from virtual COM Port.

Parameters:

sComPortNumber: COM port Number

callback:

```
typedef void(CALLBACK *CanFrameInforProc)(long error, CanFrameInfor* msg);
```

```
struct CanFrameInfor {
    int type;
    bool rtr;
    DWORD id;
    BYTE data_len;
    BYTE data[CAN_BUS_MAX_DATA_SIZE];
};

enum CanFrameInfor type {
    CP_29Bit,
    CP_11Bit
};

enum CanFrameInfor rtr {
    DISABLE_RTR,
    ENABLE_RTR
};
```

Return Value:

If the function succeeds, the return value is zero. If the function fails, the callback parameter “error” is nonzero and the error code generated by it. Refer to section **Error Codes** to find more information.

2.4.10. EGPCSend

long EGPCSend(char* sComPortNumber, CanFrameInfor* msg)

Function: Send CAN bus frame via virtual COM Port.

Parameters:

sComPortNumber: COM port number.

msg: The send frame, include frame format, frame id, frame length and frame data.

```
struct CanFrameInfor {
    int type;
    bool rtr;
    DWORD id;
    BYTE data_len;
    BYTE data[CAN_BUS_MAX_DATA_SIZE];
};

enum CanFrameInfor type {
    CP_29Bit,
    CP_11Bit
};

enum CanFrameInfor rtr {
    DISABLE_RTR,
    ENABLE_RTR
};
```

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. Refer to section **Error Codes** to find more information.

2.4.11. EGPCReset

```
long EGPCReset(char* sComPortNumber)
```

Function: Reset CAN bus port.

Parameters:

sComPortNumber: COM port Number

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. Refer to section **Error Codes** to find more information.

2.4.12. EGPCShowVer

```
viod EGPCShowVer(char* ver_info)
```

Function: get module version

Parameters:

ver_info: module version information.

2.4.13. EGPCWarnStr

const char* EGPCWarnStr (void)

Function: Show last warning message.

2.4.14. Error Codes

The below table lists errors that Fintek CAN bus functions API returns in response to calls.

| Error Code | Description |
|------------|---|
| 0x80000005 | CAN number ERROR |
| 0x80000007 | Thread ERROR |
| 0x8005FFFF | FATAL ERROR(bus off) |
| 0x80050FFF | CAN BUFFER FULL |
| 0x800500XX | ClearCommError ERROR |
| 0x80060002 | INVALID HANDLE VALUE |
| 0x80060003 | No response from device |
| 0x80060008 | CAN function called fail |
| 0x80060101 | CAN communication fail |
| 0x80060201 | Write fail |
| 0x80060202 | Read fail |
| other | CAN ERROR: bit [31:24]: Receive Error Counter bit [23:16]: Transmit Error Counter bit[15:8]: Error Code Capture bit[7]: Bus Error bit[6]: Arbitration Lost bit[5]: Error Passive bit[3]: Data Overrun bit[2]: Error Warning |

3. Linux OS

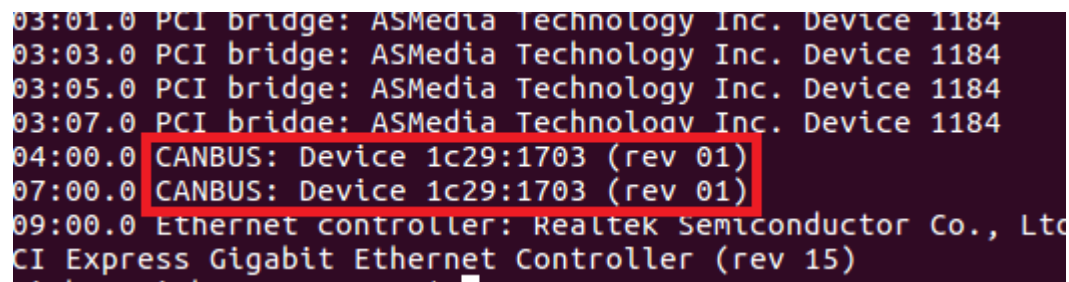
The following uses EGPC-B4S1 for driver installation and test as the example.

3.1. SocketCAN Driver Installation

SocketCAN driver of ExPC Fintek PCIe CAN bus Series supports Linux kernel 2.6.38 and above.

Please follow the steps to install SocketCAN driver.

Use “lspci” command to make sure the device is recognized by OS.



```
03:01.0 PCI bridge: ASMedia Technology Inc. Device 1184
03:03.0 PCI bridge: ASMedia Technology Inc. Device 1184
03:05.0 PCI bridge: ASMedia Technology Inc. Device 1184
03:07.0 PCI bridge: ASMedia Technology Inc. Device 1184
04:00.0 CANBUS: Device 1c29:1703 (rev 01)
07:00.0 CANBUS: Device 1c29:1703 (rev 01)
09:00.0 Ethernet controller: Realtek Semiconductor Co., Ltd
CI Express Gigabit Ethernet Controller (rev 15)
```

Follow the step below to install the SocketCAN driver.

- Change to root privilege.
 - \$ sudo su
- Prepare the kernel tree & compiler tools for your distribution.
 - \$ apt-get update
 - \$ apt-get install build-essential gcc
- Change directory to the SocketCAN folder, then build SocketCAN driver
 - \$ make clean; make; make install

```

root@yichen-MS-7971: /SocketCAN
root@yichen-MS-7971:/SocketCAN# make clean;make;make install
make[1]: Entering directory '/SocketCAN/driver'
make -C /lib/modules/4.4.0-148-generic/build M=/SocketCAN/driver clean
make[2]: Entering directory '/usr/src/linux-headers-4.4.0-148-generic'
  CLEAN    /SocketCAN/driver/.tmp_versions
  CLEAN    /SocketCAN/driver/Module.symvers
make[2]: Leaving directory '/usr/src/linux-headers-4.4.0-148-generic'
rm -rf *.~ *.o *.ko *.mod.c *.cmd *.o.d .tmp_versions Module.symvers modul
es.order Module.markers
rm -f /lib/modules/4.4.0-148-generic/kernel/drivers/char/f81601.ko
make[1]: Leaving directory '/SocketCAN/driver'
make[1]: Entering directory '/SocketCAN/driver'
make -C /lib/modules/4.4.0-148-generic/build M=/SocketCAN/driver modules
make[2]: Entering directory '/usr/src/linux-headers-4.4.0-148-generic'
  CC [M]    /SocketCAN/driver/f81601.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC        /SocketCAN/driver/f81601.mod.o
  LD [M]    /SocketCAN/driver/f81601.ko
make[2]: Leaving directory '/usr/src/linux-headers-4.4.0-148-generic'
#make -C /DataDisk/old/hpeter/allwinner/loboris/test/sunxi_jwrdegoede M=/Socket
CAN/driver modules
#make -C /DataDisk/hpeter/DMA-210UII/samsung_android_kernel_3.0 M=/SocketCAN/dri
ver modules
#make -C /DataDisk/freescale/android_x86-6.0-r2/out/target/product/x86/obj/kerne
l M=/SocketCAN/driver modules
#make -C /DataDisk/freescale/imx/build/tmp/work/imx6qsabresd-poky-linux-gnueabi/
linux-imx/4.9.11-r0/build modules M=/SocketCAN/driver
make[1]: Leaving directory '/SocketCAN/driver'
make[1]: Entering directory '/SocketCAN/driver'
# make -C /lib/modules/4.4.0-148-generic/build M=/SocketCAN/driver modules_insta
ll
# make -C /lib/modules/`uname -r`/build M=/SocketCAN/driver modules_install
cp f81601.ko /lib/modules/4.4.0-148-generic/kernel/drivers/char
depmod -a
make[1]: Leaving directory '/SocketCAN/driver'
root@yichen-MS-7971:/SocketCAN#

```

- Module probe

- \$ modprobe f81601

```

yichen@yichen-MS-7971: /SocketCAN
yichen@yichen-MS-7971:/SocketCAN$ modprobe f81601

```

- Use the following command to check driver version

- \$ dmesg | grep 'Fintek'

```

root@yichen-MS-7971: /SocketCAN/release
root@yichen-MS-7971:/SocketCAN/release# dmesg | grep 'Fintek'
[ 1.952371] f81601 0000:02:00:0: Fintek F81601 Driver version: v1.09

```

- Use the following command to check CAN BUS device is available (can0/... etc.)

- \$ ls /sys/class/net/ -al

```
yichen@yichen-MS-7971: /SocketCAN
yichen@yichen-MS-7971: /SocketCAN$ ls /sys/class/net/ -al
total 0
drwxr-xr-x  2 root root 0 2月 19 17:12 .
drwxr-xr-x 60 root root 0 2月 19 17:12 ..
lrwxrwxrwx  1 root root 0 2月 19 17:12 can0 -> ../../devices/pci0000:00/0000:00
:1b.2/0000:02:00.0/net/can0
lrwxrwxrwx  1 root root 0 2月 19 17:12 can1 -> ../../devices/pci0000:00/0000:00
:1b.2/0000:02:00.0/net/can1
lrwxrwxrwx  1 root root 0 2月 19 17:12 can2 -> ../../devices/pci0000:00/0000:00
:1c.3/0000:04:00.0/net/can2
lrwxrwxrwx  1 root root 0 2月 19 17:12 can3 -> ../../devices/pci0000:00/0000:00
:1c.3/0000:04:00.0/net/can3
lrwxrwxrwx  1 root root 0 2月 19 17:12 eth0 -> ../../devices/pci0000:00/0000:00
:1d.3/0000:06:00.0/net/eth0
lrwxrwxrwx  1 root root 0 2月 19 17:12 lo -> ../../devices/virtual/net/lo
yichen@yichen-MS-7971: /SocketCAN$
```

- Use the following command to start SocketCAN port

(reference script: Linux/SocketCAN/release/start_socketcan.sh)

Ip link set **(port name)** type can bitrate **(baud rate value)** sample-point **(value)**

- \$ ip link set can0 type can bitrate 1000000 sample-point 0.75

- \$ ip link set can0 up qlen 1000

- \$ ifconfig

```
root@yichen-MS-7971: /SocketCAN# ip link set can0 type can bitrate 1000000 sample-point 0.75
root@yichen-MS-7971: /SocketCAN# ip link set can0 up qlen 1000
root@yichen-MS-7971: /SocketCAN# ifconfig
can0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
UP RUNNING NOARP  MTU:16  Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
Interrupt:61
```

The following is the sample point of supported baud rate. Please modify the sample-point by different baud rate setting.

| Baud Rate | Sample-point |
|-----------|--------------|
| 1000000 | 0.75 |
| 800000 | 0.80 |
| 500000 | 0.875 |
| 250000 | 0.875 |
| 125000 | 0.875 |
| 100000 | 0.875 |
| 50000 | 0.875 |
| 20000 | 0.875 |
| 10000 | 0.875 |

- Repeat the “ip link” command to active all CAN bus ports.

```
can0    Link encap:UNSPEC HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
        UP RUNNING NOARP MTU:16 Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
        Interrupt:66

can1    Link encap:UNSPEC HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
        UP RUNNING NOARP MTU:16 Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
        Interrupt:66

can2    Link encap:UNSPEC HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
        UP RUNNING NOARP MTU:16 Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
        Interrupt:67

can3    Link encap:UNSPEC HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
        UP RUNNING NOARP MTU:16 Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
        Interrupt:67
```

- End can0 (reference script: Linux/SocketCAN/release/end_socketcan.sh)
- \$ ip link set can0 down

```
root@yichen-MS-7971: /SocketCAN
root@yichen-MS-7971:/SocketCAN# ip link set can0 down
root@yichen-MS-7971:/SocketCAN#
```

- Run the following command in the “release” folder to add/remove boot up script.

- \$ chmod +x add_2_boot.sh
- \$./add_2_boot.sh

```
yichen@yichen-MS-7971:~/svn/Inno/Trunk/EP/EGPC_B4S1/Linux/SocketCAN/release$ ./add_2_boot.sh
yichen@yichen-MS-7971:~/svn/Inno/Trunk/EP/EGPC_B4S1/Linux/SocketCAN/release$
```

- \$ chmod +x remove_boot.sh
- \$./remove_boot.sh

```
yichen@yichen-MS-7971:~/svn/Inno/Trunk/EP/EGPC_B4S1/Linux/SocketCAN/release$ ./remove_boot.sh
yichen@yichen-MS-7971:~/svn/Inno/Trunk/EP/EGPC_B4S1/Linux/SocketCAN/release$
```

NOTE: You also can use “start_socketcan.sh” and “end_socketcan.sh” to initial or remove EGPC SocketCAN driver, please provide the “execute” permission by using “chmod +x” then run the scripts.

3.2. CAN-utils

After SocketCAN setup is finished, you can use open source project “can-utils” to test by “cansend” and “candump”.

(<https://github.com/linux-can/can-utils>).

- Install CAN-utils
 - \$ apt-get install can-utils
- use can0 to send and can1 to receive.

```
yichen@yichen-MS-7971:~$ cansend can0 111#1122334455667788
yichen@yichen-MS-7971:~$ cansend can0 111#1122334455667788
yichen@yichen-MS-7971:~$ cansend can0 111#1122334455667788
yichen@yichen-MS-7971:~$ cansend can0 111#R1
yichen@yichen-MS-7971:~$ cansend can0 111#R2
yichen@yichen-MS-7971:~$ cansend can0 111#R3
yichen@yichen-MS-7971:~$
```

```
yichen@yichen-MS-7971:~$ candump can1
can1 111 [8] 11 22 33 44 55 66 77 88
can1 111 [8] 11 22 33 44 55 66 77 88
can1 111 [8] 11 22 33 44 55 66 77 88
can1 111 [1] remote request
can1 111 [2] remote request
can1 111 [3] remote request
```


3.3. Loop Back Test Program

We provide a SocketCAN loop back test program with source code in Linux to verify the module. Please connect CAN ports the same as the description in “[2.2. Loop Back Test Program](#)”.

The test program can be configured by edit “setup.ini”. In majority cases, just keep the default setting.

- Change to root privilege.

- \$ sudo su

-Change the directory to the “tool” folder and run “Make”.

```
ylchen@H5-7971:~/svn/Inno/Trunk/EP/EGPC_B451/Linux/tool$ make
make[1]: Entering directory '/home/ylchen/svn/Inno/Trunk/EP/EGPC_B451/Linux/tool/can-utils'
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE canwg.c -o canwg
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE isotpdump.c -o isotpdump
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE isotperrf.c -o isotperrf
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE isotprecv.c -o isotprecv
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE isotpsend.c -o isotpsend
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE isotpsniffer.c -o isotpsniffer
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE isotptun.c -o isotptun
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o jact.o jact.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o jcat.o jcat.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o jsr.o jsr.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o jspy.o jspy.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o jsr.o jsr.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o testj1939.o testj1939.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o slcan_attach.o slcan_attach.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o slcand.o slcand.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o asc2log.o asc2log.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o lib.o lib.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o bcmsniffer.o bcmsniffer.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o can-calc-bit-timing.o can-calc-bit-timing.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o canbusload.o canbusload.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o canfranelen.o canfranelen.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o candump.o candump.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o canfdtest.o canfdtest.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o cangen.o cangen.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o canlogserver.o canlogserver.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o canplayer.o canplayer.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o cansend.o cansend.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o loopback_send_script.o loopback_send_script.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o loopback_dump_script.o loopback_dump_script.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o loopback.o loopback.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o loopback_send.o loopback_send.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o loopback_dump.o loopback_dump.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o lnt.o lnt.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o cansniffer.o cansniffer.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o log2asc.o log2asc.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o log2long.o log2long.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o log2long.o log2long.c
cc -O2 -Wall -Wno-parentheses -Iinclude -DAP_CAN=PF_CAN -DPP_CAN=29 -DSO_RXQ_OVFL=40 -DSCH_TIMESTAMPING_OPT_STATS=54 -D_FILE_OFFSET_BITS=64 -D_GNU_SOURCE -c -o scanty.o scanty.c
make[1]: Leaving directory '/home/ylchen/svn/Inno/Trunk/EP/EGPC_B451/Linux/tool/can-utils'
```

-Run the test

- \$./loopback

```
yichen@yichen-MS-7971:~/svn/Inno/Trunk/EP/EGPC_B4S1/Linux/tool$ ./loopback
Config
-----
pair_1_device_1: can0
pair_1_device_2: can1
pair_2_device_1: can2
pair_2_device_2: can3
sleep_interval: 200
test_time: 1
data_length: 8
rtr: disable
id_type: 29Bit
-----

Config
Config
-----
send_CAN: can0
id_type: 29Bit
rtr: Disable
data_length: 8
sleep_interval: 200
test_time: 1
-----

send can frame: 1FFFFFFF#1122334455667788

-----
send_CAN: can2
id_type: 29Bit
rtr: Disable
data_length: 8
sleep_interval: 200
test_time: 1
-----

send can frame: 1FFFFFFF#1122334455667788

Elapsed time: 000.160294 (sec)
Elapsed time: 000.314360 (sec)
Elapsed time: 000.471596 (sec)
Elapsed time: 000.627987 (sec)
Elapsed time: 000.786241 (sec)
Elapsed time: 000.946617 (sec)
Elapsed time: 001.105472 (sec)
Elapsed time: 001.261737 (sec)
Elapsed time: 001.418679 (sec)
Elapsed time: 001.572617 (sec)
Interval: 001.609711 (sec)
Pair1:
tx_cnt:5129, rx_cnt:5129
Result: PASS!
Pair2:
tx_cnt:5113, rx_cnt:5113
Result: PASS!
yichen@yichen-MS-7971:~/svn/Inno/Trunk/EP/EGPC_B4S1/Linux/tool$
```

Contact us

Headquarters (Taiwan)

5F., No. 237, Sec. 1, Datong Rd., Xizhi Dist., New Taipei City 221, Taiwan

Tel: +886-2-77033000

Email: sales@innodisk.com

Branch Offices:

USA

usasales@innodisk.com

+1-510-770-9421

Europe

eusales@innodisk.com

+31-40-3045-400

Japan

jpsales@innodisk.com

+81-3-6667-0161

China

sales_cn@innodisk.com

+86-755-21673689

www.innodisk.com

© 2022 Innodisk Corporation.

All right reserved. Specifications are subject to change without prior notice.

April 27, 2023