




# PU2CANFD 系列产品用户手册

产品资料 and 软件我们一般更新在以下链接:

		
<p><b>PU2CANFD-X2</b> USB 转双路 CANFD, 兼容 PCAN-USB PRO FD</p>	<p><b>PU2CANFD-C</b> USB 转 CANFD, 兼容 PCAN-USB FD</p>	<p><b>PU2CANFD-MPCIE</b> Minipcie 接口 USB 转双路 CANFD, 兼容</p>

<a href="#">日期</a>	<a href="#">版本</a>	<a href="#">修订内容</a>
<a href="#">20230409</a>	<a href="#">V1</a>	<a href="#">发布</a>
<a href="#">20230410</a>	<a href="#">V1</a>	<a href="#">增加目录</a>

目录

<b>PU2CANFD 系列产品用户手册</b> .....	<b>1</b>
<b>1 产品介绍</b> .....	<b>4</b>
1.1 产品介绍 .....	4
1.2 产品特性 .....	4
1.3 系统配置要求 .....	5
1.4 资料清单 .....	5
<b>2 硬件描述</b> .....	<b>5</b>
2.1 PU2CANFD-X2 .....	5
2.2 PU2CANFD-C .....	9
2.3 PU2CANFDX2-MPCIE .....	12
<b>3 硬件接线参考</b> .....	<b>19</b>
3.1 使能/禁用内部 120 欧姆终端电阻 .....	19
3.2 USB2CANFD-X2 接线示意图 .....	19
3.3 PU2CANFD-C 接线示意图 .....	20
3.4 PU2CANFDX2-MPCIE 接线示意图 .....	20
<b>4 Linux Socket-CAN 使用说明</b> .....	<b>21</b>
4.1 安装 Linux 工具 .....	22
4.2 使用 CAN-UTILS 程序 .....	23
4.3 使用 C 程序 .....	26
4.4 使用 Python3 程序 .....	31
4.5 错误帧说明（英文） .....	33
附录 4--4.6 Linux 编译 PCAN 字符驱动 .....	35
<b>系统环境要求</b> .....	<b>37</b>
<b>5 PCAN-View 使用说明</b> .....	<b>42</b>
5.1 安装驱动 .....	43
5.2 连接框说明 .....	43
5.3 接收报文 .....	44
5.4 发送报文 .....	49
5.5 总线负载选项 .....	50
5.6 Status Bar（状态栏） .....	51
5.7 简单通讯案例 .....	51
5.8 ID 设置选项卡 .....	52
<b>6 PCAN-Explorer 使用说明</b> .....	<b>53</b>
6.1 PCAN-Explorer5 基本使用方法 .....	53
6.2 PCAN-Explorer5 入门手册 .....	53
6.3 J1939 基本使用方法 .....	54
6.4 Symbol Editor 基本使用方法 .....	54
6.5 Plotter 插件基本使用方法 .....	54
6.6 Instrument Panel 插件基本使用方法 .....	54
<b>7 免费软件编程接口介绍</b> .....	<b>55</b>
7.1 PCAN-Basic API .....	55
7.2 PCAN-CCPAPI 与 PCAN-XCPAPI .....	56

7.3 PCAN-ISO TPAPI .....	56
7.4 PCAN-UDS API .....	57
7.5 PCAN-OBD-2 API .....	57
7.6 PCAN-PassThru API .....	58
<b>附录 A-ID 设置使能/禁用终端电阻 .....</b>	<b>58</b>
A1 针对设备在 Windows 系统下使用的 ID 设置 .....	59
A2 针对设备在 Linux 系统下使用的 ID 设置 .....	60
A3 SW CAN 模式设置 .....	60
(仅 USB2CANFD-X2 有 SWCAN 功能) .....	60

# 1 产品介绍

产品名称	CAN 版本	CAN 最大比特率	电气隔离	每秒最大帧数	兼容	时间戳	产品描述
<b>PU2CANFD-X2</b> <b>USB2CANFD-X2</b>	CAN2.0A/B CANFD 1.0	12Mbit/s	2500V	15000fps/s	PCAN-USB PRO FD IPEH-004061 (No LIN)	1us	USB 转双路 CANFD
<b>PU2CANFDX2</b> <b>-MPCIE</b>	CAN2.0A/B CANFD 1.0	12Mbit/s	2500V	15000fps/s	PCAN-miniPCIe FD IPEH-004046	1us	USB 双路 CANFD minipcie 接口
<b>PU2CANFD-C</b>	CAN2.0A/B CANFD 1.0	12Mbit/s	2500V	15000fps/s	PCAN-USB FD IPEH-004022	1us	USB 转单路高品质 CAN 分析仪

## 1.1 产品介绍

### 新的 CAN FD 标准

CAN FD 标准 (CAN with Flexible Data Rate) 主要特征是更高数据传输带宽。每个 CAN FD 帧最大 64 个数据位(代替目前的 8 位)可用最快 12Mbit/s 比特率进行传输。CAN FD 向下兼容 CAN 2.0 A/B 标准, 因此 CAN FD 节点可用于现有 CAN 网络。但是, 在这种情况下, CAN FD 扩展不可用

### ISO 和非 ISO CAN FD

从最初博世发布 CAN FD 版本之后, 现在协议进行了改进, 是 ISO 11898-1 标准; 修订后的 CAN FD 版本与原协议不兼容。考虑到这种情况, 所以两种 CANFD 版本都提供了支持; 可以通过软件直接切换。

## 1.2 产品特性

- USB2.0 高速模式(兼容 USB1.1、USB2.0、USB3.0)
- 支持 ISO (11898-2) 和 NON-ISO, 软件切换;
- CAN FD 数据域 (64 位最大)比特率从 25kbit/s 至 12Mbit/s
- 普通 CAN 比特率范围 25 kbit/s 至 1 Mbit/s;
- 时间戳分辨率最低可达实测 1us;
- 每路 CANFD 通道高达 2500 V 电气隔离;
- 120 欧姆终端电阻可以通过软件激活, 同时配备两个 DB9 端子板 120 欧姆终端电阻, 可通过跳线帽激活;

- 总线负载测量包括错误帧和过载帧通，支持错误帧检测和总线状态监测
- 过 USB 供电,工作温度范围-40—85°C (-40 to 185 °F)  
支持 linux 系统内核已集成驱动 (Socket CAN)，或者安装 PCAN 自定义字符驱动。  
支持 MacOS (MacCAN)，savvyCAN
- 二次开发：PCAN-Basic API (应用程序接口)。

PCAN-Basic API(应用程序接口)是用于 PCAN 硬件接口系列的二次开发的应用程序接口。它允许开发简单的 CAN 应用，以实现和我们的 PCAN-PC 硬件通信。API 包括实际的设备驱动和一个提供 API 函数接口的 DLL (动态链接库)。PCAN-Basic 为开发者提供了各种环境下的多种函数，包括 C#, C++/CLR, Delphi,VB.NET, Java, 和 Python 2.6, 在开发包中都有这些环境下的例程，更多资料请参考：<http://www.peak-system.com>

## 1.3 系统配置要求

- WINDOWS11,10,8.1,7 (32/64 位系统)
- Linux (32/64 位系统)
- USB 接口

## 1.4 资料清单

- Windows PCANVIEW 软件
- Linux PCAN 字符驱动说明 (使用 Linux PCANVIEW)
- Linux Socket-CAN 使用 CAN-UTILS,C 程序 (含源码),Python 程序 (含源码)
- PCAN-Basic API (应用程序接口)。  
PCAN-Basic 为开发者提供了各种环境下的多种函数，包括 C#, C++/CLR, Delphi,VB.NET, Java, 和 Python 2.6
- MacCAN,SwavyCAN 程序及官方使用说明
- PDF 版本用户手册

# 2 硬件描述

## 2.1 PU2CANFD-X2

(USB2CANFD-X2)



### 2.1.1 产品规格

USB2CANFD-X2 设备是一款即插即用使用高速 USB2.0 转换为双路 CANFD 的通讯设备，让您轻松接入 CAN 网络。每路 CANFD 通道（数据和电压）高达 2500 V 磁耦隔离保证 PC 和 CAN 端之间，保护您的 PC 和设备面熟复杂工作环境电气干扰和静电侵害。相比 PCAN，USB2CANFD-X2 增加了 120 欧姆终端电阻软件使能功能，双路 CANFD 最高比特率可以达到 12Mbit/s

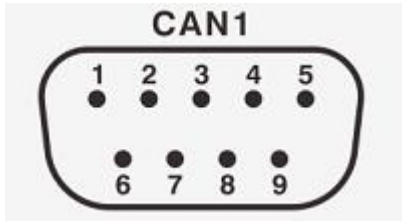
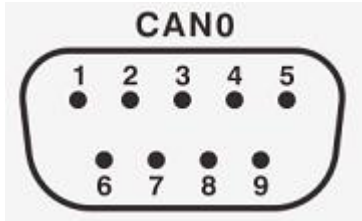
<b>Connector</b>	
CANFD	Dual Channel 9PIN D-SUB Connectors
USB	USB plug type A (Computer) USB plug type B (Devicie)
<b>CAN Features</b>	
Protocols	CAN 2.0A (standard format) CAN 2.0B (extended format) CAN FD ISO 11898-1:2015 CAN FD non-ISO
CAN bit rates	25 kbit/s up to 1 Mbit/s
CANFD bit rates	25 kbit/s up to 12Mbit/s

USB ESD	IEC 61000-4-2 Level 4 +25 kV (contact discharge) +30 kV (air-gap discharge)
Galvanic isolation	Signal & Power Separately Isolated by 2500 Volts against USB IEC 61000-4-2 Contact IEC 61000-4-2 Air ISO 10605 150 pF / 2 k2 Contact ISO 10605 330 pF 1 2 k2 Contact
Micro controller	180MHZ Cortex-M4 MCU
Timestamp resolution	1 $\mu$ s
Built In 120 $\Omega$ Termination Resistor	Enable/Disable Through Software
Software	Windows PCAN-VIEW Linux PCAN-VIEW (Instruction) Linux SOCKET-CAN: <ul style="list-style-type: none"> <li>● CAN Utils (Instruction) ,</li> <li>● C (Source Code Instruction) ,</li> <li>● Python (Source Code Instruction)</li> <li>● savvyCAN</li> <li>● Busmaster</li> </ul>
PCAN BASIC API Windows 10, 8.1, 7 (32/64-bit) Windows CE 6.x (x86/ARMv4) Linux (32/64-bit)	C#, C++/CLR, Delphi, VB.NET, Java, Python 2.6
Third Party Software	LabView, CodeSys, Matlab, BUSMASTER, EasyMotion Studio, CANmoon, XX-SCAN, PCAN-Explorer5
<b>Others</b>	
Temperature	-40°~ 85°
PCBA Size (L * W * H)	84x80x28 mm
Weight	190g

USB2CANFD-X2 提供 Windows CAN 监视器 PCAN - view 和 Linux PCAN-View 软件，提供基于 Linux 的 CAN-UTILS，C 程序及源码，Python 程序及源码。  
 设备驱动程序支持不同的操作系统 X86/ARM 的 Windows/Linux，因此程序可以很容易地访问连接的 CAN 总线  
 支持多种第三方软件：LabView, CodeSys, Matlab, BUSMASTER, EasyMotion Studio, CANmoon, XX-SCAN, PCAN-Explorer5.

2.2.3 供货清单

## 2.1.2 引脚描述

CAN1		引脚说明	
		1	NC
		2	CANL bus line (dominant low)
		3	CAN_GND
		4	NC
		5	NC
		6	NC
		7	CANH bus line (dominant high)
		8	NC
		9	NC
CAN0		引脚说明	
		1	NC
		2	CANL bus line (dominant low)
		3	CAN_GND
		4	NC
		5	NC
		6	NC
		7	CANH bus line (dominant high)
		8	NC
		9	NC

注意：GND 不连接不影响通信；如果带有屏蔽层，建议连接到 GND 脚。

## 2.1.3 LED 指示灯





当将设备插入电脑时，所有灯都会闪烁后关闭，Power 指示灯（Link）常亮。

（备注：如果使能了终端电阻，TERM 对应的通道指示灯也会常亮）

对应区域	描述
电源指示灯（LINK）	当将设备插入到电脑后，LINK 指示灯常亮
CAN1 LED 指示灯	TX LED 闪烁,发送数据 RX LED 闪烁, 接收数据 TERM LED 变绿,终端电阻使能
CAN0 LED 指示灯	TX LED 闪烁,发送数据 RX LED 闪烁, 接收数据 TERM LED 变绿,终端电阻使能

#### 2.1.4 装箱清单

- 带铝壳的 USB2CANFD-X2 设备
- 两个 DB9 凤凰端口转接板板载 120 欧姆终端电阻（可通过跳线帽设置）
- 1.2 米高品质 USB 线（用于连接设备和电脑）
- 软件下载链接

## 2.2 PU2CANFD-C

The PU2CANFD-C is a plug and play high speed USB2.0 to CANFD converter comes with 1.2m high quality Cable enables the connection of one channel CANFD networks to a computer via USB. CANFD is isolated protection against USB 2500V.

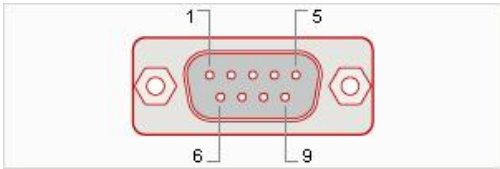


### 2.2.1 产品规格

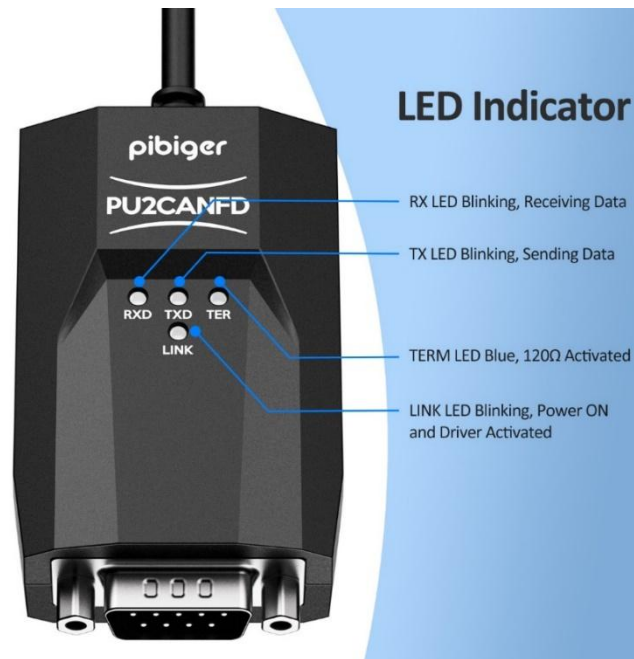
<b>Connector</b>	
CANFD	9PIN D-SUB Connectors
USB Cable	1.2M High Quality USB Cable with type A
<b>CAN Features</b>	
Protocols	CAN 2.0A (standard format) CAN 2.0B (extended format) CAN FD ISO 11898-1:2015 CAN FD non-ISO
CAN bit rates	25 kbit/s up to 1 Mbit/s
CANFD bit rates	25 kbit/s up to 12Mbit/s
USB ESD	IEC 61000-4-2 Level 4 +25 kV (contact discharge) +30 kV (air-gap discharge)
Galvanic isolation	Signal & Power Separately Isolated by 2500 Volts against USB IEC 61000-4-2 Contact

	IEC 61000-4-2 Air ISO 10605 150 pF / 2 k2 Contact ISO 10605 330 pF 1 2 k2 Contact
Micro controller	180MHZ Cortex-M4 MCU
Timestamp resolution	1 $\mu$ s
Built In 120 $\Omega$ Termination Resistor	Disable/Enable Through Software
Software	<ul style="list-style-type: none"> <li>● Windows PCAN-VIEW AND API</li> <li>● Linux PCAN-VIEW AND API</li> <li>● Linux SOCKET-CAN: CAN Utils/C Code/Python Code</li> <li>● Mac CAN And API</li> </ul>
PCAN BASIC API Windows Windows CE 6.x Linux (32/64-bit) Mac Os	C#, C++/CLR, Delphi,VB.NET, Java, Phyton 2.6
Third Party Software	LabView, CodeSys, Matlab, BUSMASTER, EasyMotion Studio, CANmoon, XX-SCAN, PCAN-Explorer5 Comes with Products: <ul style="list-style-type: none"> <li>● savvyCAN</li> <li>● Busmaster</li> </ul>
<b>Others</b>	
Temperature	-40°~ 85°

## 2.2.2 引脚描述

	<table border="1"> <tr><td>1</td><td>NC</td></tr> <tr><td>2</td><td>CAN-L</td></tr> <tr><td>3</td><td>GND</td></tr> <tr><td>4</td><td>NC</td></tr> <tr><td>5</td><td>NC</td></tr> <tr><td>6</td><td>NC</td></tr> <tr><td>7</td><td>CAN-H</td></tr> <tr><td>8</td><td>NC</td></tr> <tr><td>9</td><td>NC</td></tr> </table>	1	NC	2	CAN-L	3	GND	4	NC	5	NC	6	NC	7	CAN-H	8	NC	9	NC
1	NC																		
2	CAN-L																		
3	GND																		
4	NC																		
5	NC																		
6	NC																		
7	CAN-H																		
8	NC																		
9	NC																		

## 2.2.3 LED 指示灯



Led Name	LED Status	Description
RXD	Blinking	Receiving Data
TXD	Blinking	Sending Data
TER	Blue Led On	Build In 120Ω Term Resistor Enable;
LINK	Blinking	Power Up and Driver Installed.
	Blue Led On	Power Up Driver Not Installed.

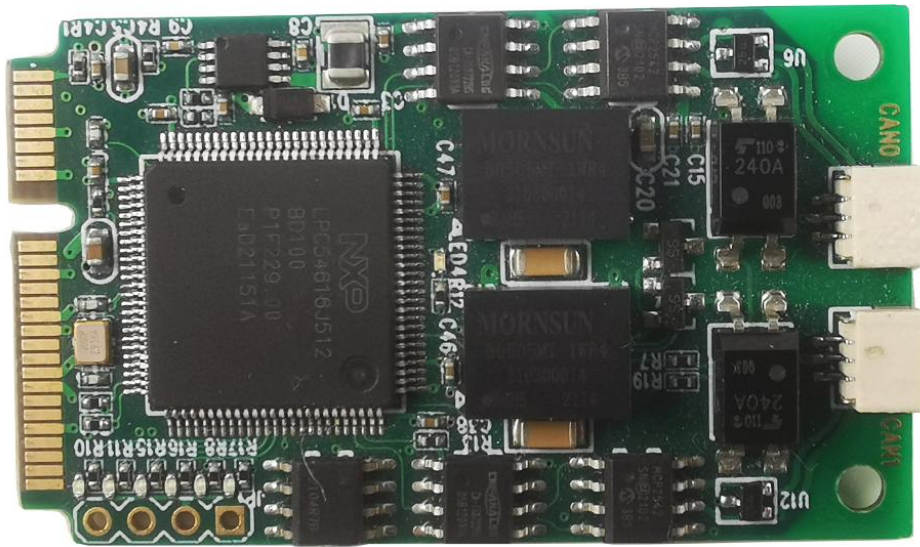
## 2.2.4 装箱单

- 1.2 米高品质 USB 转 CANFD 通讯线
- 1 个 DB9 凤凰端口转接板板载 120 欧姆终端电阻（可通过跳线帽设置）
- 软件下载链接

## 2.3 PU2CANFDX2-MPCIE

The PU2CANFD-MPCIE is a plug and play high speed USB2.0 to CANFD CAN Card. CAN FD each channel is separately isolated against USB with a maximum of 2500V.

**High Speed USB2.0 To CAN FD Converter**  
**Data Field Up to 12M Max**  
**Compatible With PCAN Software**

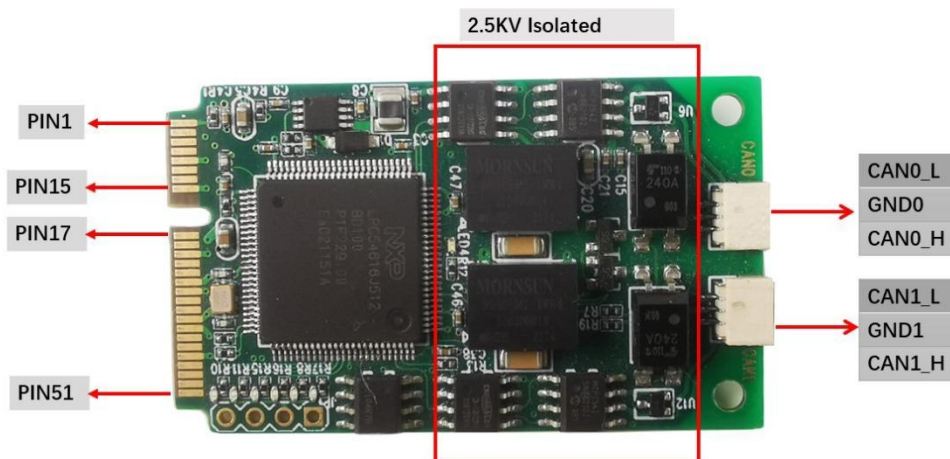


### 2.3.1 产品规格

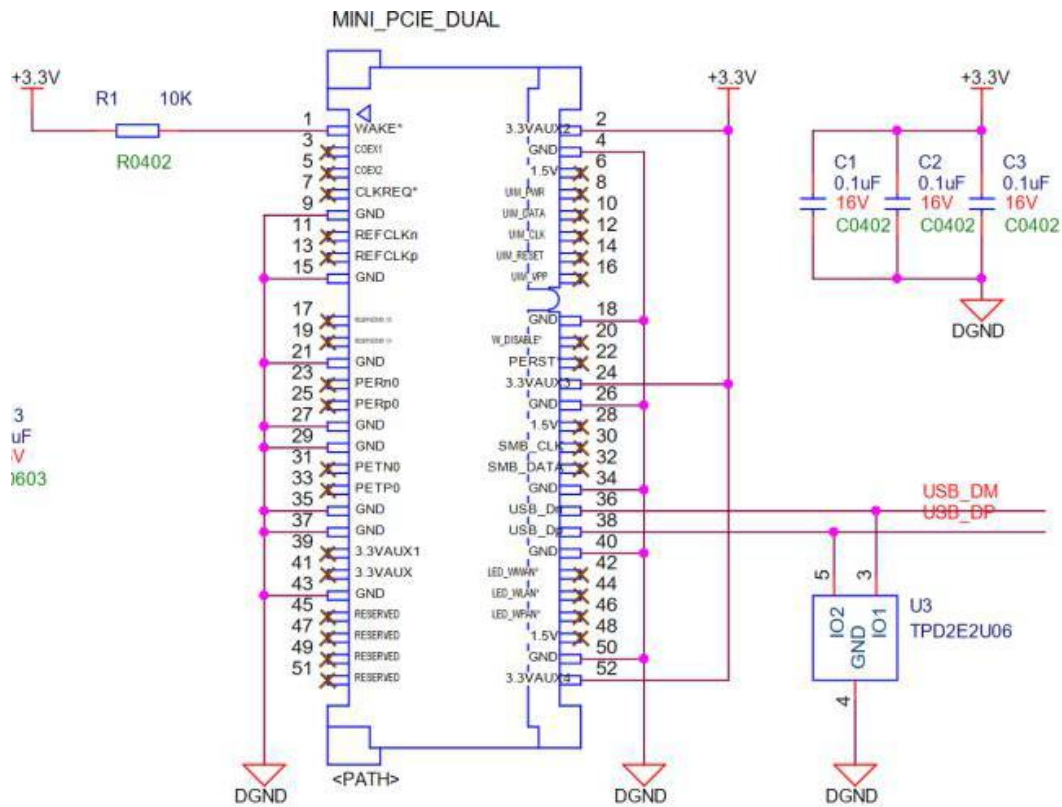
<b>Connector</b>	
Dual Channel CANFD	2X3PIN 1mm Connector
Mini PCIE USB	USB BUS
<b>CAN Features</b>	
Protocols	CAN 2.0A (standard format) CAN 2.0B (extended format) CAN FD ISO 11898-1:2015 CAN FD non-ISO
CAN bit rates	25 kbit/s up to 1 Mbit/s
CANFD bit rates	25 kbit/s up to 12Mbit/s
USB ESD	IEC 61000-4-2 Level 4 +25 kV (contact discharge) +30 kV (air-gap discharge)
Galvanic isolation	Signal &Power Separately Isolated by 2500 Volts against USB IEC 61000-4-2 Contact IEC 61000-4-2 Air ISO 10605 150 pF / 2 k2 Contact ISO 10605 330 pF 1 2 k2 Contact
Micro controller	180MHZ Cortex-M4 MCU
Timestamp resolution	1 $\mu$ s
Built In 120 $\Omega$ Termination Resistor	Activated/Deactivated Through Software

Software	Windows PCAN-VIEW Linux PCAN-VIEW (Instruction) Linux SOCKET-CAN: <ul style="list-style-type: none"> <li>● CAN Utils (Instruction) ,</li> <li>● C (Source Code Instruction) ,</li> <li>● Python (Source Code Instruction)</li> </ul>
PCAN BASIC API Windows 10, 8.1, 7 (32/64-bit) Windows CE 6.x (x86/ARMv4) Linux (32/64-bit)	C#, C++/CLR, Delphi,VB.NET, Java, Python 2.6
Third Party Software	savvyCAN Busmaster
<b>Others</b>	
Temperature	-40°~ 85°
PCBA Size (L * W * H)	84x80x28 mm
Weight	191g

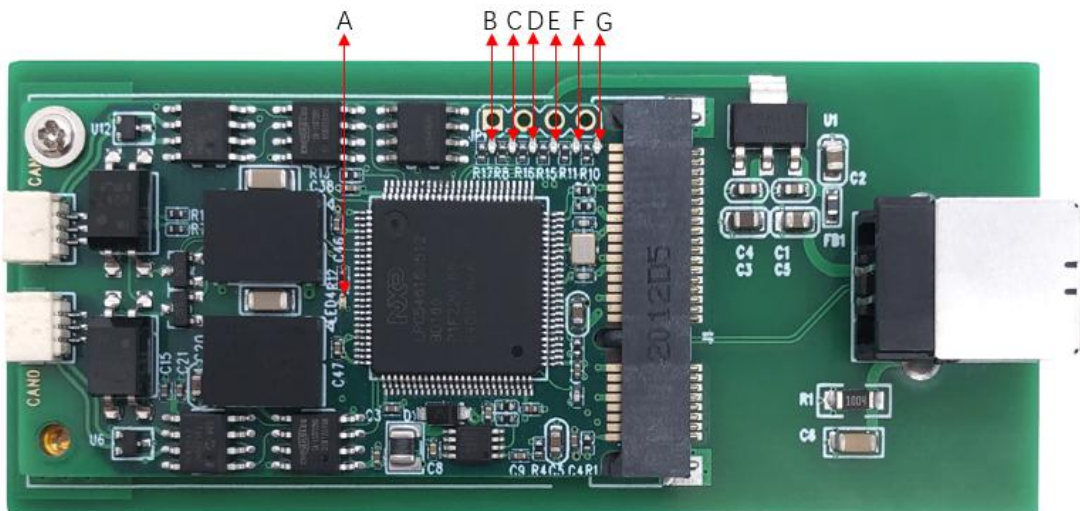
### 2.3.2 引脚描述



**High Speed USB2.0 To CAN FD Converter**  
**Data Field Up to 12M Max**  
**Compatible With PCAN Software**



### 2.3.3 LED 指示灯



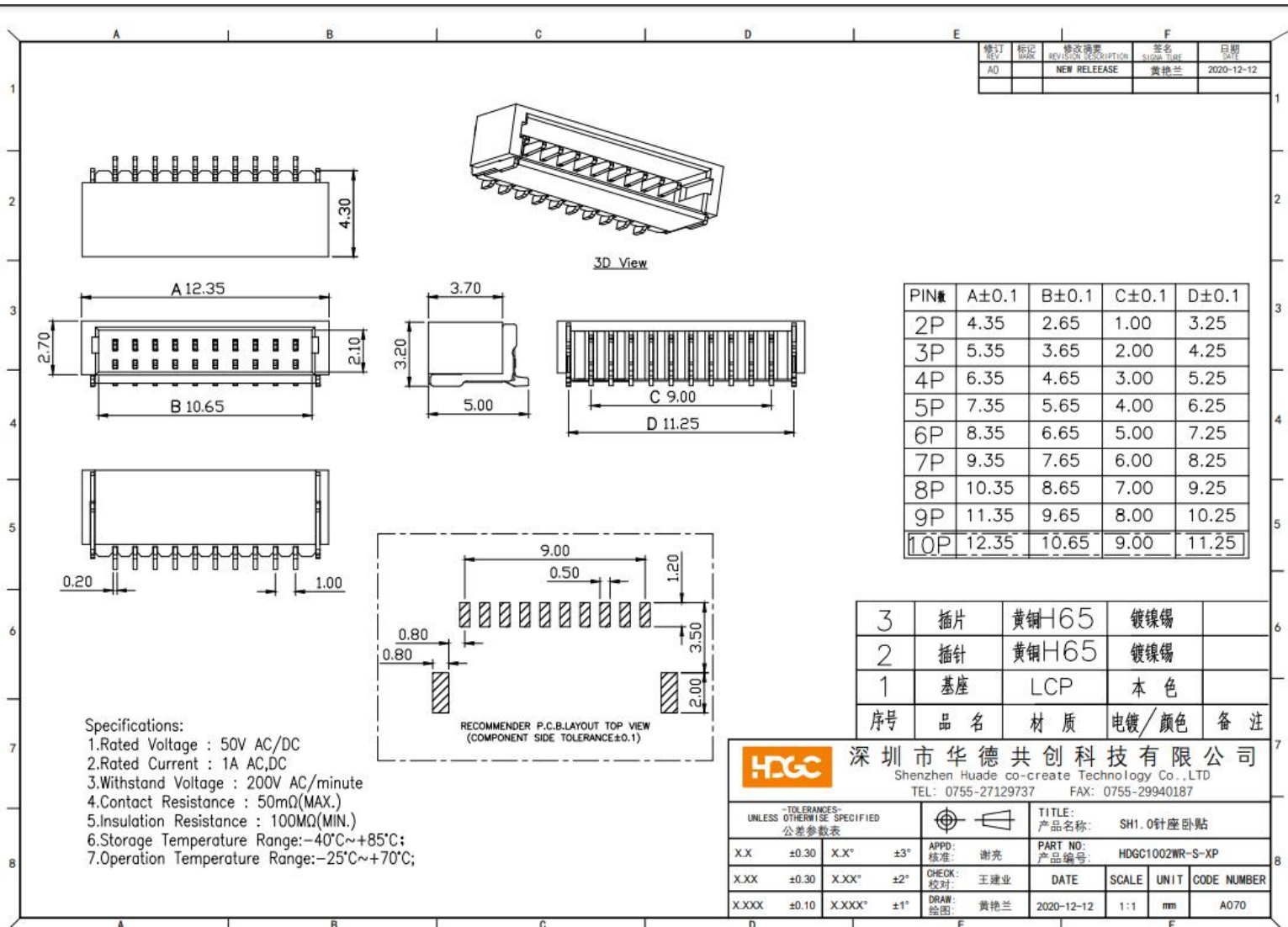
	Description	Indication
A	Power&Driver LED	Blinking, Driver install and Power Up
B	CAN0 120Ω Term	On, CAN0 Term Enable; Off, CAN0 Term Disable

C	CAN1 120Ω Term	On, CAN1 Term Enable; Off, CAN1 Term Disable
D	CAN0 Receive	Blinking, CAN0 Receiving Data;
E	CAN0 Send	Blinking, CAN0 Sending Data;
F	CAN1 Receive	Blinking, CAN1 Receiving Data;
G	CAN1 Send	Blinking, CAN0 Receiving Data;

## 2.3.4 设计参考

### 2.3.4.1 板端 3PIN 连接器

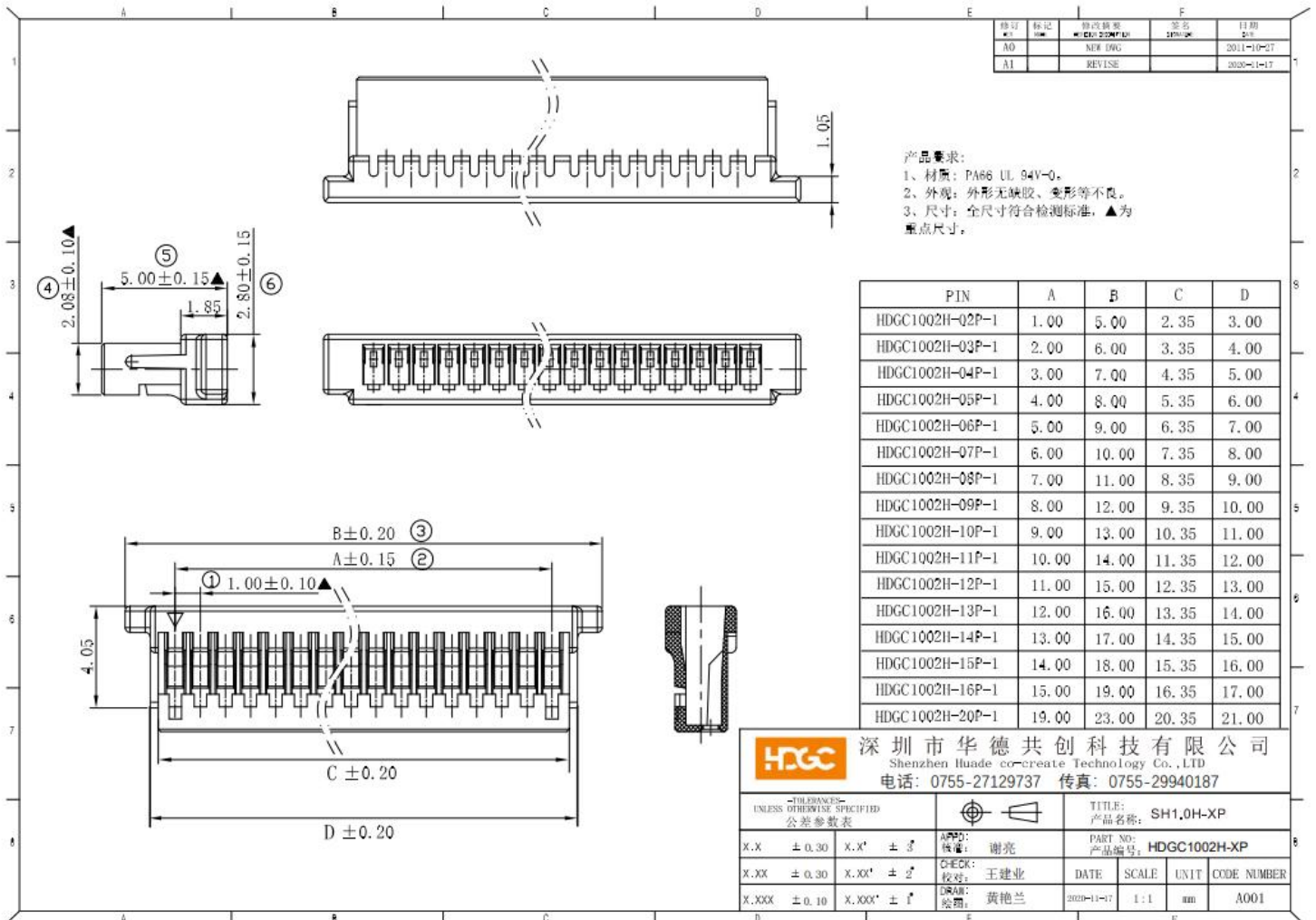
供应商型号: HDGC1002WR-S-3P





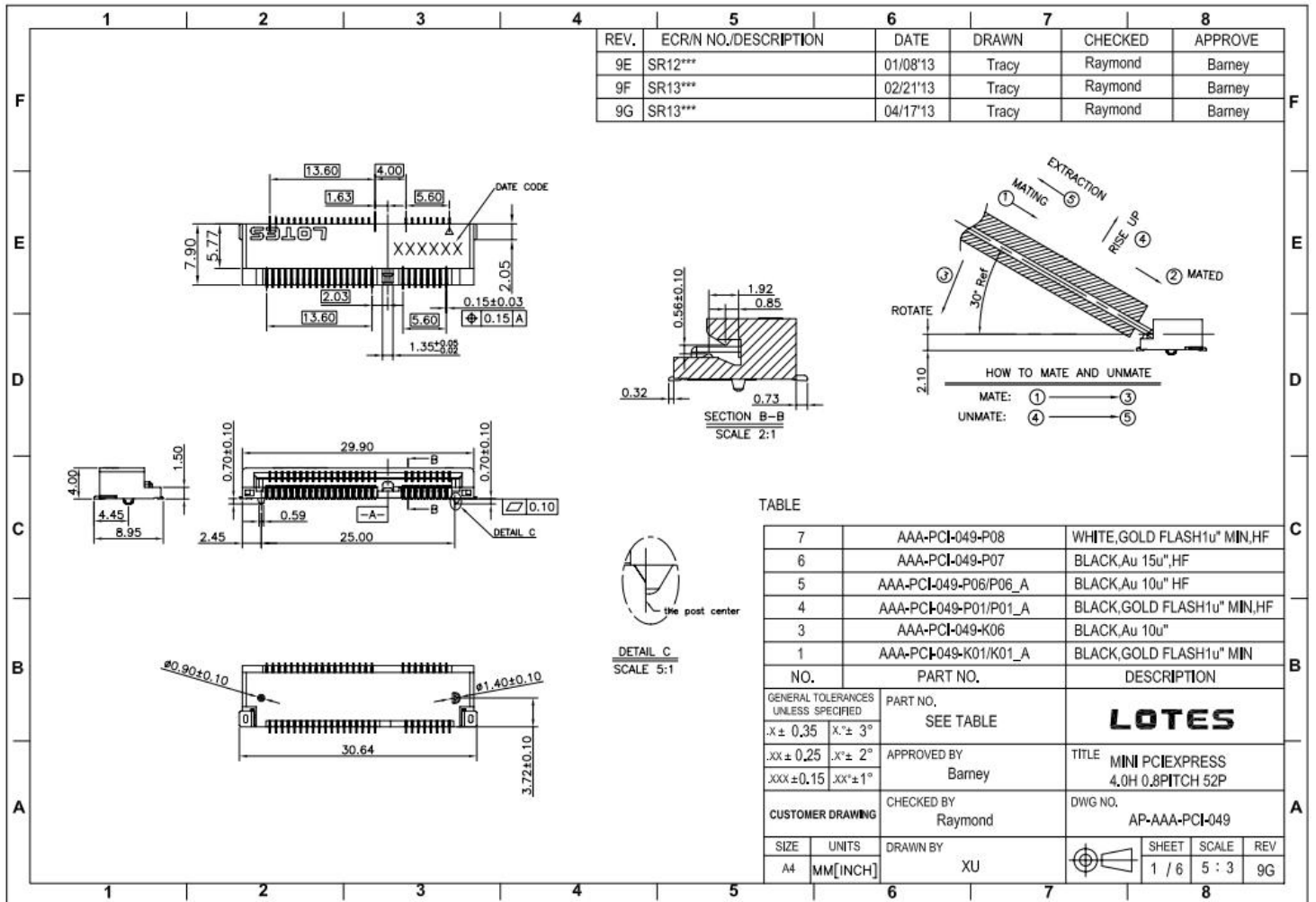
### 2.3.4.2 连接端 3PIN 连接器

供应商: HDGC1002H-5P



### 2.3.4.3 minipcie 连接器

供应商: Lotes, AAA-PCI-049-K01



### 2.3.4 装箱单

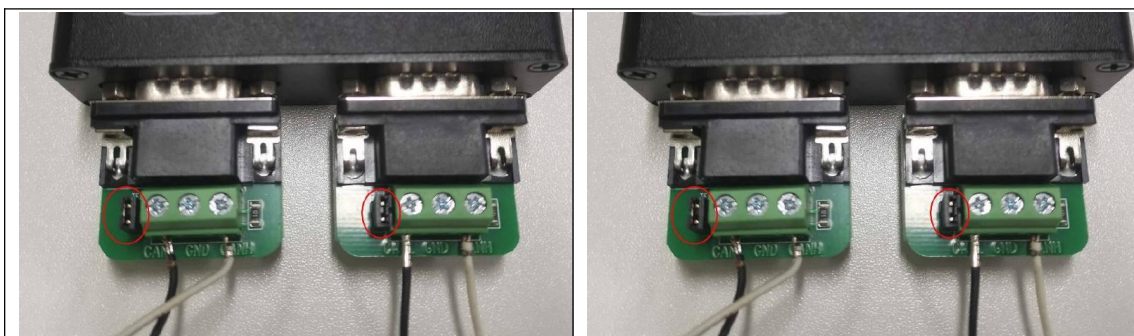
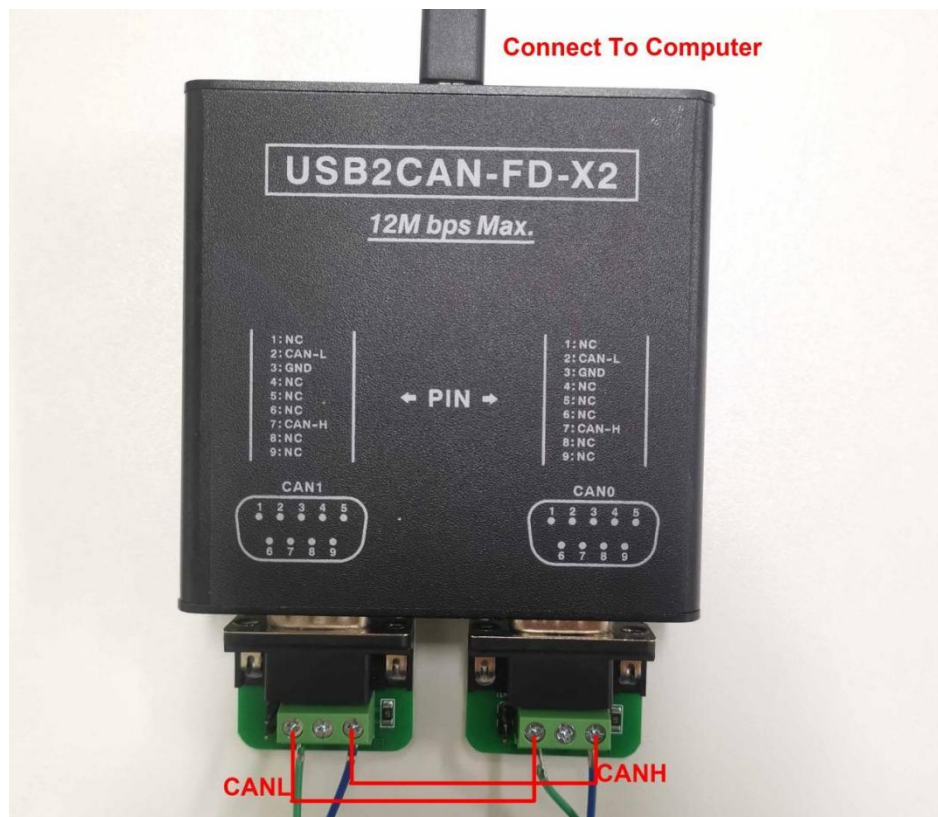
- MINIPCIE CANFD CAN 卡 X1
- MINIPCIE CANFD 底板 X1
- 1.2 米高品质线材 X1
- 2x3PIN 端子，另外一端为线，测试使用
- 软件下载链接

## 3 硬件接线参考

### 3.1 使能/禁用内部 120 欧姆终端电阻

产品默认使能了内部内部 120 欧姆终端电阻，无须再外接终端电阻。如果需要禁用内部 120 欧姆终端电阻，请参考说明书最后的附录说明。

### 3.2 USB2CANFD-X2 接线示意图



注意：如内部 120 欧姆终端电阻已经使能，需要移除圆圈中的终端电阻。

注意：如内部 120 欧姆终端电阻没有使能，需要增加圆圈中的终端电阻。

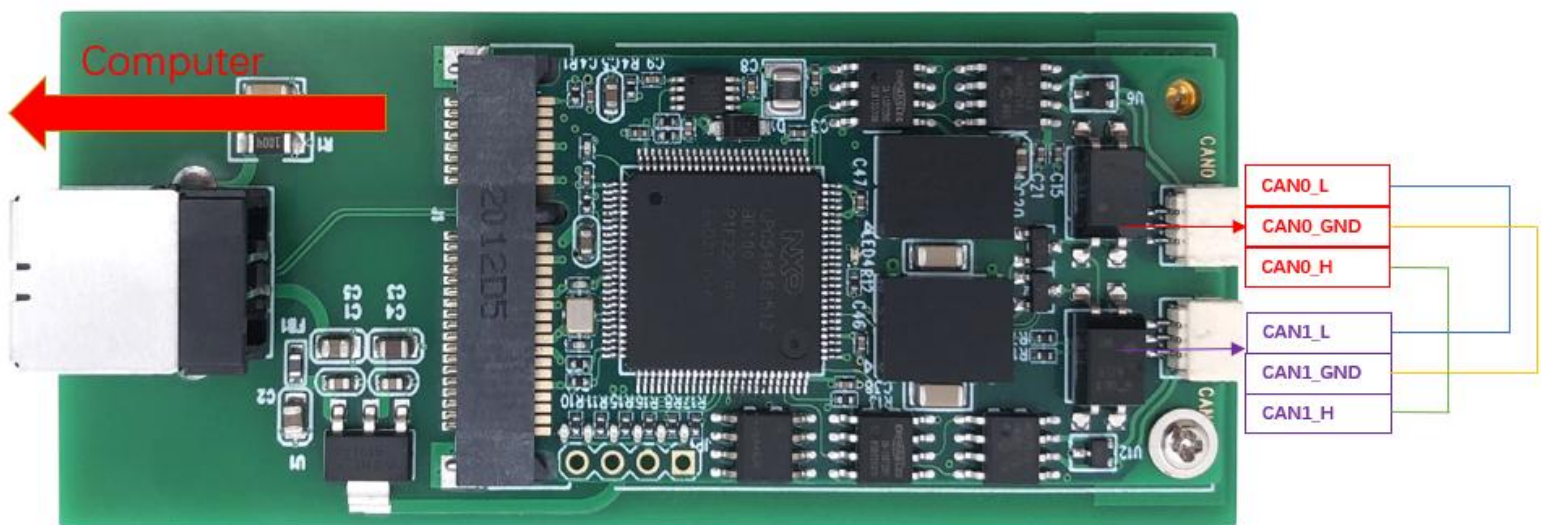
### 3.3 PU2CANFD-C 接线示意图

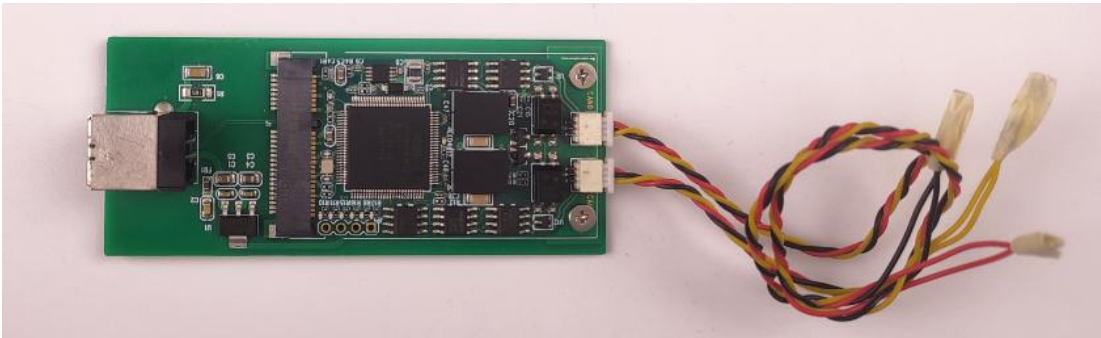


内部 120 欧姆终端电阻已经使能情况下，必须移除红色圈内的电阻。

### 3.4 PU2CANFDX2-MPCIE 接线示意图

内部 120 欧姆终端电阻已经使能



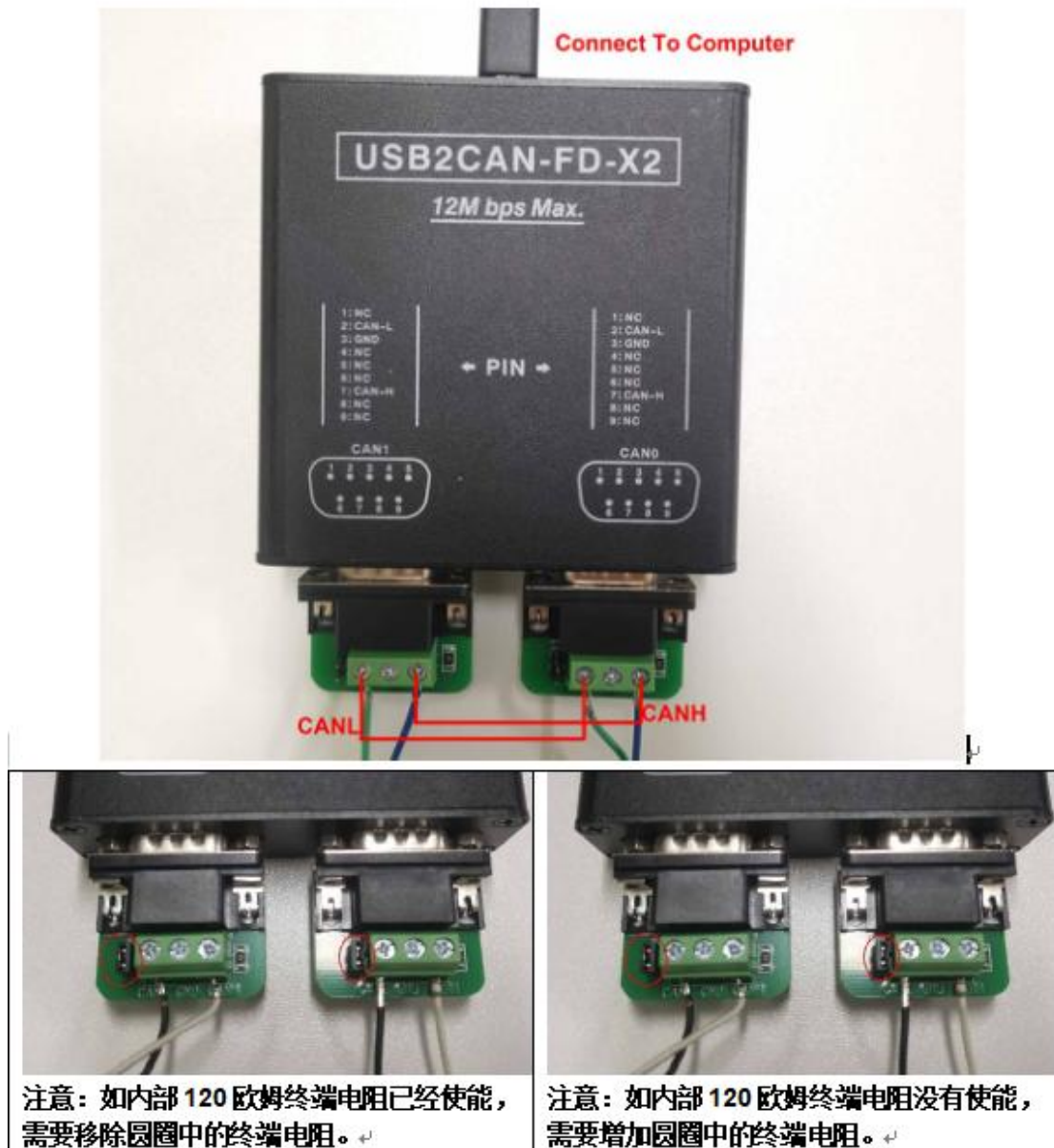


## 4 Linux Socket-CAN 使用说明

Linux 系统下 CAN 设备可被自动识别为 **(SocketCAN) netdev**

- 一般 Linux 内核已经包含 CAN 接口的驱动程序并以 SocketCAN 框架作为网络设备(又名 **netdev**)访问 CAN 接口。
- 也可以自行编译 PCAN 字符驱动 **chardev** 安装使用 **PCAN-BASIC**，请参考章节 4.6 本章节主要以 **SocketCAN** 为主。

本章节硬件以 **USB2CANFD-X2** 为例，硬件接线参考章节 3.2，LED 指示灯参考章节 2.1.3



## 4.1 安装 Linux 工具

### Step1, 安装 Net-Tools

先检查 "can0" 和 "can1" 设备在系统中是否可用

```
$ifconfig -a
```

如果找不到命令 ifconfig, 使用以下命令安装

```
$sudo apt-get install net-tools
```

设备被识别成功后如下图:

```
virtual-machine:~$ ifconfig -a
can0: flags=128<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

can1: flags=128<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
```

键入命令 `dmesg` 以检查更多设备信息

```
[ 4334.851804] usb 1-1: new high-speed USB device number 6 using ehci-pci
[ 4335.120375] usb 1-1: New USB device found, idVendor=0c72, idProduct=0011, bcdDevice= 0.00
[ 4335.120380] usb 1-1: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[ 4335.120384] usb 1-1: Product: PCAN-USB Pro FD
[ 4335.120385] usb 1-1: Manufacturer: PEAK-System Technik GmbH
[ 4335.137143] peak_usb 1-1:1.0: PEAK-System PCAN-USB Pro FD v2 fw v3.2.0 (2 channels)
[ 4335.143995] peak_usb 1-1:1.0 can0: attached to PCAN-USB Pro FD channel 0 (device 0)
[ 4335.151388] peak_usb 1-1:1.0 can1: attached to PCAN-USB Pro FD channel 1 (device 0)
```

## Step2, 安装 can-utils

输入以下命令安装 can-utils

```
$sudo apt-get install can-utils
```

## 4.2 使用 CAN-UTILS 程序

CAN-UTILS 工具是测试 USB2CANFD-X2 模块通信是否正常，是一个简单的使用说明。有关更多详细信息，请参阅 can-utils 用户手册和源代码。

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

### Step1, 设置波特率

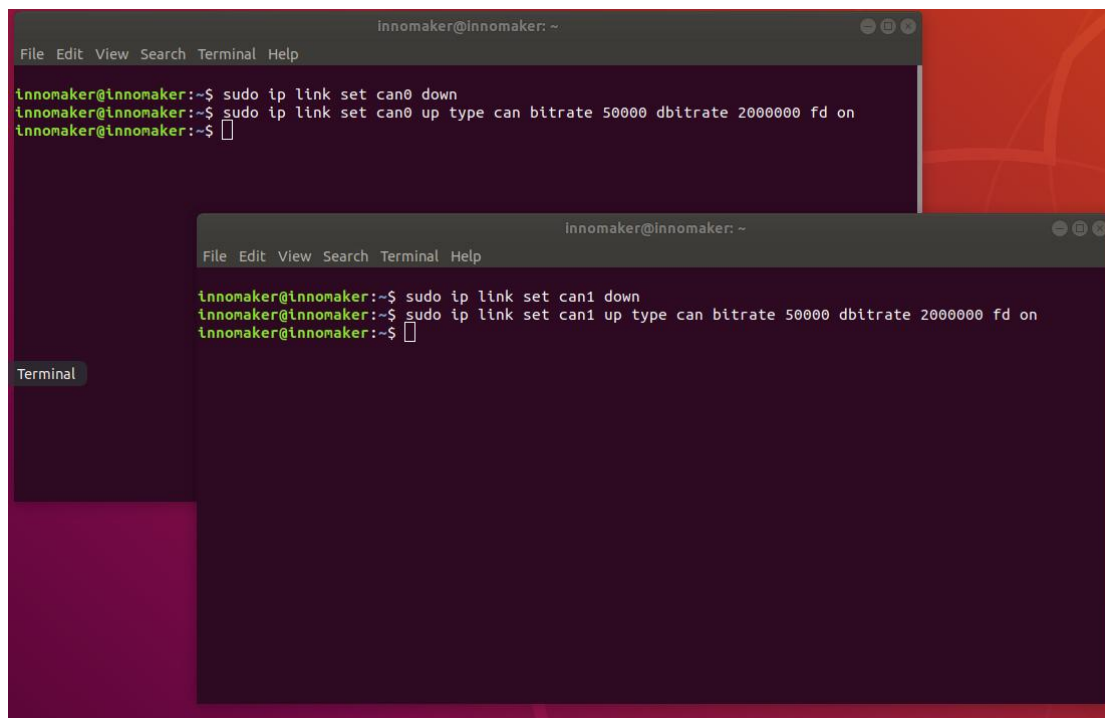
打开两个 Terminal 命令窗口，分别给 CAN0 和 CAN1 设置通讯波特率

```
sudo ip link set can0 down
```

```
sudo ip link set can0 up type can bitrate 50000 dbitrate 2000000 fd on
```

```
sudo ip link set can1 down
```

```
sudo ip link set can1 up type can bitrate 50000 dbitrate 2000000 fd on
```



```
innomaker@innomaker: ~  
File Edit View Search Terminal Help  
innomaker@innomaker:~$ sudo ip link set can0 down  
innomaker@innomaker:~$ sudo ip link set can0 up type can bitrate 50000 dbitrate 2000000 fd on  
innomaker@innomaker:~$  
  
innomaker@innomaker: ~  
File Edit View Search Terminal Help  
innomaker@innomaker:~$ sudo ip link set can1 down  
innomaker@innomaker:~$ sudo ip link set can1 up type can bitrate 50000 dbitrate 2000000 fd on  
innomaker@innomaker:~$
```

## Step2, 接收和发送

<1>然后设置 CAN0 为接收

`candump can0`

<2>设置 CAN1 为发送

`cansend can1 500#1E.10.10`

收发效果如下图



```
innomaker@innomaker: ~  
File Edit View Search Terminal Help  
innomaker@innomaker:~$ sudo ip link set can0 down  
innomaker@innomaker:~$ sudo ip link set can0 up type can bitrate 50000 dbitrate 2000000 fd on  
innomaker@innomaker:~$ candump can0  
can0 500 [3] 1E 10 10  
can0 500 [3] 1E 10 10  
can0 500 [3] 1E 10 10  
can0 500 [3] 1E 10 10  
innomaker@innomaker: ~  
File Edit View Search Terminal Help  
innomaker@innomaker:~$ sudo ip link set can1 down  
innomaker@innomaker:~$ sudo ip link set can1 up type can bitrate 50000 dbitrate 2000000 fd on  
innomaker@innomaker:~$ cansend can1 500#1E.10.10  
innomaker@innomaker:~$  
innomaker@innomaker:~$ cansend can1 500#1E.10.10  
innomaker@innomaker:~$ cansend can1 500#1E.10.10  
innomaker@innomaker:~$ cansend can1 500#1E.10.10  
innomaker@innomaker:~$
```

### Step3, CAN-UTILS 指令介绍

#### ● candump

candump 可以实时显示接收到的 can 消息。要实时显示设备 can0 上的所有 can 消息，在命令终端中输入以下命令：

```
candump can0
```

candump 还可以使用掩码和标识符对接收到的 can 信息进行过滤。有两种过滤器类型：

- [can\_id]:[can\_mask] : 当[received\_can\_id] & [can\_mask] == [can\_id] & [mask] 被显示
- [can\_id]~[can\_mask] : 当[received\_can\_id] & [can\_mask] != [can\_id] & [mask] 被显示

例如：

仅显示 can0 上收到的 ID 为 0x123 的消息：

```
candump vcan0,0x123:0x7FF
```

仅显示 can0 上收到的 ID 为 0x123 或 0x456 的消息：

```
candump vcan0,0x123:0x7FF,0x456:0x7FF
```

## ● cansend

cansend 可以将单个 CAN 帧发送到总线上。您将必须指定设备，标识符和要发送的数据字节。

例如：

```
cansend can0 123#1122334455667788
```

此条指令将在接口 can0 上发送一条消息，其标识符为 0x123，数据字节为[0x11、0x22、0x33、0x44、0x55、0x66、0x77、0x88]。请注意，此工具始终假定值以十六进制给出。

## ● Cangen

cangen 可以生成随机的 CAN 数据，这对于测试很有用。有关更多的用法信息，请在命令终端中输入：

```
cangen --help
```

## ● Cansniffer

cansniffer 可以显示总线上接收到的 CAN 消息，而且可以过滤掉数据不变的帧。这对于逆向工程 CAN 总线系统非常有用。有关更多信息，请在命令终端中输入：

```
cansniffer --help
```

## 4.3 使用 C 程序

**Step1, 切换到源码程序目录：**

```
PU2CANFD-C\For_Linux_SocketCAN\c
```

```
├─ can_receive.c
├─ can_send.c
├─ can0_receive_fd
├─ can1_send_fd
```

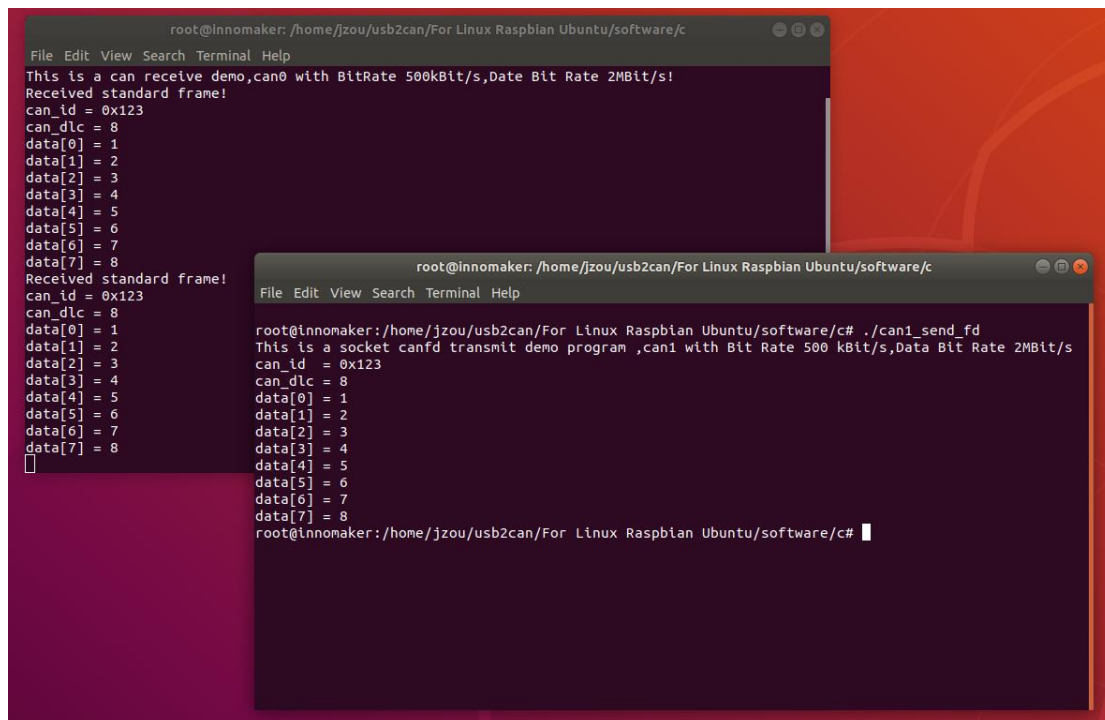
**Step2, 设置 CAN0 为接收端**

```
sudo ./can0_receive_fd
```

### Step3, 设置 CAN1 为发送端

```
sudo ./can1_send_fd
```

具体 C 程序请参照文件: can\_send.c, can\_receive.c



```
root@innomaker: /home/jzou/usb2can/For Linux Raspbian Ubuntu/software/c
File Edit View Search Terminal Help
This is a can receive demo,can0 with BitRate 500kBit/s,Data Bit Rate 2MBit/s!
Received standard frame!
can_id = 0x123
can_dlc = 8
data[0] = 1
data[1] = 2
data[2] = 3
data[3] = 4
data[4] = 5
data[5] = 6
data[6] = 7
data[7] = 8
Received standard frame!
can_id = 0x123
can_dlc = 8
data[0] = 1
data[1] = 2
data[2] = 3
data[3] = 4
data[4] = 5
data[5] = 6
data[6] = 7
data[7] = 8
█

root@innomaker: /home/jzou/usb2can/For Linux Raspbian Ubuntu/software/c
File Edit View Search Terminal Help
root@innomaker: /home/jzou/usb2can/For Linux Raspbian Ubuntu/software/c# ./can1_send_fd
This is a socket canfd transmit demo program ,can1 with Bit Rate 500 kBit/s,Data Bi Rate 2MBit/s
can_id = 0x123
can_dlc = 8
data[0] = 1
data[1] = 2
data[2] = 3
data[3] = 4
data[4] = 5
data[5] = 6
data[6] = 7
data[7] = 8
root@innomaker: /home/jzou/usb2can/For Linux Raspbian Ubuntu/software/c# █
```

### 附录 1-C 程序源码及说明（英文）

For Sender's codes

(1): Create the socket, If an error occurs then the return result is -1.

```
/*Create socket*/
s = socket(PF_CAN, SOCK_RAW, CAN_RAW);
if (s < 0) {
    perror("Create socket PF_CAN failed");
    return 1;
}
```

(2): Locate the interface to "can0" or other name you wish to use. The name will show when you execute "./ifconfig -a".

```
/*Specify can0 device*/  
strcpy(ifr.ifr_name, "can0");  
ret = ioctl(s, SIOCGIFINDEX, &ifr);  
if (ret < 0) {  
    perror("ioctl interface index failed!");  
    return 1;  
}
```

(3): Bind the socket to "can0".

```
/*Bind the socket to can0*/  
addr.can_family = PF_CAN;  
addr.can_ifindex = ifr.ifr_ifindex;  
ret = bind(s, (struct sockaddr *)&addr, sizeof(addr));  
if (ret < 0) {  
    perror("bind failed");  
    return 1;  
}
```

(4): Disable sender's filtering rules, this program only send message do not receive packets.

```
/*Disable filtering rules, this program only send message do not receive packets */  
setsockopt(s, SOL_CAN_RAW, CAN_RAW_FILTER, NULL, 0);
```

(5): Assembly data to send.

```
/*assembly message data! */  
frame.can_id = 0x123;  
frame.can_dlc = 8;  
frame.data[0] = 1;  
frame.data[1] = 2;  
frame.data[2] = 3;  
frame.data[3] = 4;  
frame.data[4] = 5;  
frame.data[5] = 6;  
frame.data[6] = 7;  
frame.data[7] = 8;  
//if(frame.can_id&CAN_EFF_FLAG==0)  
if(!(frame.can_id&CAN_EFF_FLAG))  
    printf("Transmit standard frame!\n");  
else  
    printf("Transmit extended frame!\n");
```

(6): Send message to the can bus. You can use the return value of write() to check whether all data has been sent successfully .

```
/*Send message out */  
nbytes = write(s, &frame, sizeof(frame));  
if(nbytes != sizeof(frame)) {  
    printf("Send frame incompletely!\r\n");  
    system("sudo ifconfig can0 down");  
}
```

(7): Close can0 device and disable socket.

```
/*Close can0 device and destroy socket!*/  
close(s);
```

For Receiver's codes

(1)step 1 and (2) is same as Sender's code.

(3):It's different from Sender's.

```
/*Bind the socket to can0*/  
addr.can_family = PF_CAN;  
addr.can_ifindex = ifr.ifr_ifindex;  
ret = bind(s, (struct sockaddr *)&addr, sizeof(addr));  
if (ret < 0) {  
    perror("bind failed");  
    return 1;  
}
```

(4): Define receive filter rules,we can set more than one filters rule.

```
/*Define receive filter rules,we can set more than one filter rule!*/  
struct can_filter rfilter[2];  
rfilter[0].can_id = 0x123;//Standard frame id !  
rfilter[0].can_mask = CAN_SFF_MASK;  
rfilter[1].can_id = 0x12345678;//extend frame id!  
rfilter[1].can_mask = CAN_EFF_MASK;
```

(5): Read data back from can bus.

```
nbytes = read(s, &frame, sizeof(frame));
```

## 附录 2-参考 C 程序二次开发

应用程序首先通过初始化一个套接字（与 TCP/IP 通信中的情况非常类似），然后将该套接字绑定到一个接口（或所有接口，如果应用程序需要），来设置对 CAN 接口的访问。一旦绑定，套接字就可以进行读取，写入等操作，像 UDP 套接字一样使用。

二次开发前需要安装 can\_dev 模块并配置 CAN 总线波特率，然后启用。

例如：

```
$ modprobe can_dev  
$ modprobe can  
$ modprobe can_raw  
$ sudo ip link set can0 type can bitrate 500000  
$ sudo ip link set up can0
```

还有一个用于测试目的的虚拟 CAN 驱动程序，可以使用以下命令在 Linux 中加载和创建：

```
$ modprobe can  
$ modprobe can_raw
```

```
$ modprobe vcan
$ sudo ip link add dev vcan0 type vcan
$ sudo ip link set up vcan0
$ ip link show vcan0
3: vcan0: <NOARP,UP,LOWER_UP> mtu 16 qdisc noqueue state UNKNOWN link/can
```

以下代码段是 SocketCAN API 的工作示例，该 API 使用原始接口发送数据包：

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <net/if.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/ioctl.h>
#include <linux/can.h>
#include <linux/can/raw.h>
int main(void)
{
    int s;
    int nbytes;
    struct sockaddr_can addr;
    struct can_frame frame;
    struct ifreq ifr;
    const char *ifname = "vcan0";
    if((s = socket(PF_CAN, SOCK_RAW, CAN_RAW)) < 0) {
        perror("Error while opening socket");
        return -1;
    }
    strcpy(ifr.ifr_name, ifname);
    ioctl(s, SIOCGIFINDEX, &ifr);
    addr.can_family = AF_CAN;
    addr.can_ifindex = ifr.ifr_ifindex;
    printf("%s at index %d\n", ifname, ifr.ifr_ifindex);
    if(bind(s, (struct sockaddr *)&addr, sizeof(addr)) < 0) {
        perror("Error in socket bind");
        return -2;
    }
    frame.can_id = 0x123;
    frame.can_dlc = 2;
    frame.data[0] = 0x11;
    frame.data[1] = 0x22;
    nbytes = write(s, &frame, sizeof(struct can_frame));
}
```

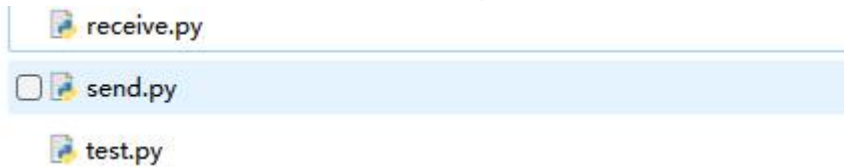
```
printf("Wrote %d bytes\n", nbytes);  
return 0;  
}
```

## 4.4 使用 Python3 程序

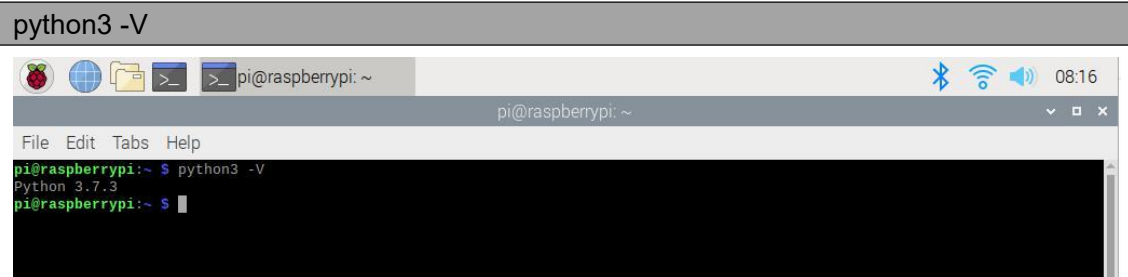
本章节介绍使用 python3 例程实现 CANFD 通讯，需要安装 python-can 及确保 python 版本不低于 3。

### Step1, 切换到程序源码目录

PU2CANFD-C\For\_Linux\_SocketCAN\python3



### Step2, 确认 Python 版本



如果 python 版本低于 3，按以下指令安装 python3

```
$sudo apt-get install python3-pip
```

### Step3, 安装 Python-can

安装 python-can

```
$sudo pip3 install python-can
```

### Step4, 设置 CAN0 为接收端

```
$sudo python3 receive.py
```

## Step5, 设置 CAN1 为发送端

```
$sudo python3 send.py
```

## 附录 3-Python 程序源码及说明 (英文)

Import

`import os`

The OS module in Python provides a way of using operating system dependent functionality. The functions that the OS module provides allows you to interface with the underlying operating system that Python is running on – be that Windows, Mac or Linux. We usually use `os.system()` function to execute a shell command to set CAN.

`import can`

The python-can library provides Controller Area Network support for Python, providing common abstractions to different hardware devices, and a suite of utilities for sending and receiving messages on a CAN bus.

For more information about python-can, please to below link:

<https://python-can.readthedocs.io/en/stable/index.html>

`ifconfig`

If you are use Ubuntu system, It may can't use the 'ifconfig' command. Please install the net tools.

`sudo apt install net-tools`

Simple common functions

(1) Set bitrate and start up CAN device.

`os.system('sudo ip link set can0 type can bitrate 1000000')`

`os.system('sudo ifconfig can0 up')`

(2) Bind the socket to 'can0'.

`can0 = can.interface.Bus(channel = 'can0', bustype = 'socketcan_ctypes')`

(3) Assembly data to send.

`msg = can.Message(arbitration_id=0x123, data=[0, 1, 2, 3], extended_id=False)`

(4) Send data.



```
can0.send(msg)
```

(5) Receive data.

```
msg = can0.recv(30.0)
```

(6) Close CAN device

```
os.system('sudo ifconfig can0 down')
```

## 4.5 错误帧说明（英文）

Now with previous demo's code to show you how to program socket can in Raspbian with C and Python . The socket can is an implementation of CAN protocols(Controller Area Network) for Linux. CAN is a networking technology which has widespread use in automation, embedded devices, and automotive fields. While there have been other CAN implementations for Linux based on character devices, Socket CAN uses the Berkeley socket API, the Linux network stack and implements the CAN device drivers as network interfaces. The CAN socket API has been designed as similar as possible to the TCP/IP protocols to allow programmers, familiar with network programming, to easily learn how to use CAN sockets. For more Socket CAN detail please refer to below link:

<https://www.kernel.org/doc/Documentation/networking/can.txt>

[https://elinux.org/CAN\\_Bus](https://elinux.org/CAN_Bus)

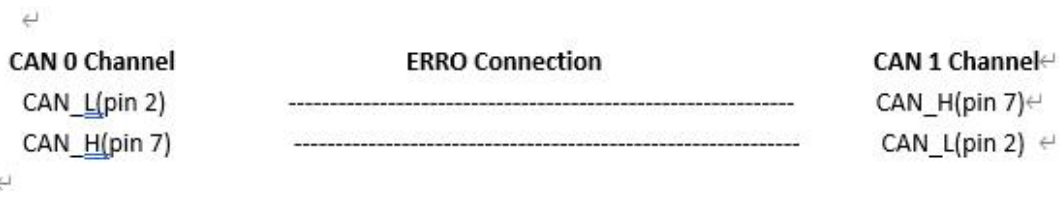
You may receive some error frame marked in red when you use the USB2CANX2-FD module. They will tell you what problem does the USB2CANX2-FD module meet on your CAN Bus.

Some people would say why didn't they meet the error frame with other tool or USB to CAN module before. The truth is that most of the tool filter out the error frame to avoid controversy and support. They just show nothing when there are some error on the CAN Bus. We want to show the all raw data to help you to analyze your CAN BUS. Some error can be ignored, but some error maybe the hidden danger for your CAN BUS.

For the error frame ID description, please refer to below link:

<https://github.com/linux-can/can-utils/blob/master/include/linux/can/error.h>

Now we take a simple case to show you how to analyze the error frame ID. I made the incorrect connection between the USB2CAN module and the CAN Bus, to see what happens.



SeqID	SystemTime	Channel	Dirac...	FrameId	Frame...	Frame...	Length	FrameData
4	2020/6/29 14:44:08	0	Recv	0x20000024	Data ...	Stand...	8	0x 00 00 00 00 00 00 00 00
5	2020/6/29 14:44:08	0	Recv	0x20000024	Data ...	Stand...	8	0x 00 00 00 00 00 00 00 00
6	2020/6/29 14:44:08	0	Recv	0x20000024	Data ...	Stand...	8	0x 00 00 00 00 00 00 00 00
7	2020/6/29 14:44:08	0	Recv	0x20000024	Data ...	Stand...	8	0x 00 00 00 00 00 00 00 00
8	2020/6/29 14:44:08	0	Recv	0x20000024	Data ...	Stand...	8	0x 00 00 00 00 00 00 00 00
9	2020/6/29 14:44:08	0	Recv	0x20000024	Data ...	Stand...	8	0x 00 00 00 00 00 00 00 00
10	2020/6/29 14:44:08	0	Recv	0x20000024	Data ...	Stand...	8	0x 00 00 00 00 00 00 00 00
11	2020/6/29 14:44:08	0	Recv	0x20000024	Data ...	Stand...	8	0x 00 00 00 00 00 00 00 00
12	2020/6/29 14:44:08	0	Recv	0x20000024	Data ...	Stand...	8	0x 00 0C 00 00 00 00 00 00
13	2020/6/29 14:44:08	0	Recv	0x20000024	Data ...	Stand...	8	0x 00 0C 00 00 00 00 00 00
14	2020/6/29 14:44:08	0	Recv	0x20000024	Data ...	Stand...	8	0x 00 0C 00 00 00 00 00 00
15	2020/6/29 14:44:08	0	Recv	0x20000024	Data ...	Stand...	8	0x 00 0C 00 00 00 00 00 00
16	2020/6/29 14:44:08	0	Recv	0x20000024	Data ...	Stand...	8	0x 00 30 00 00 00 00 00 00

As Above, We received error frame Id: 0x20000024 and 2 set of 8 byte Frame Data:  
data[0]=0x00, data[1]=0x0C,data[3] to data[7] are all 0x00 .  
data[0]=0x00, data[1]=0x30,data[3] to data[7] are all 0x00 .

According the above error frame ID description link:

```

/* error class (mask) in can_id */
#define CAN_ERR_TX_TIMEOUT 0x00000001U /* TX timeout (by netdevice driver) */
#define CAN_ERR_LOSTARB 0x00000002U /* lost arbitration / data[0] */
#define CAN_ERR_CRTL 0x00000004U /* controller problems / data[1] */
#define CAN_ERR_PROT 0x00000008U /* protocol violations / data[2..3] */
#define CAN_ERR_TRX 0x00000010U /* transceiver status / data[4] */
#define CAN_ERR_ACK 0x00000020U /* received no ACK on transmission */
#define CAN_ERR_BUSOFF 0x00000040U /* bus off */
#define CAN_ERR_BUSERROR 0x00000080U /* bus error (may flood!) */
#define CAN_ERR_RESTARTED 0x00000100U /* controller restarted */

```

$$\begin{aligned}
 & \text{This Error frame ID} = 0x200000000 \mid 0x00000020 \mid 0x00000004 \\
 & \qquad \qquad \qquad = \qquad \qquad \qquad 0x200000000 \qquad \qquad \qquad \mid \\
 & \text{CAN\_ERR\_ACK} \mid \text{CAN\_ERR\_CRTL}
 \end{aligned}$$

So the USB2CANX2-FD meet two problem 'received no ACK on transmission' and 'controller problems'.

For problem 'received no ACK on transmission' may case by the not CAN-BUS or other module on the CAN BUS are only listen mode(No ACK).

For problem 'controller problems', refer to the data[1] description:

```
/* error status of CAN-controller / data[1] */  
#define CAN_ERR_CTRL_UNSPEC      0x00 /* unspecified */  
#define CAN_ERR_CTRL_RX_OVERFLOW 0x01 /* RX buffer overflow */  
#define CAN_ERR_CTRL_TX_OVERFLOW 0x02 /* TX buffer overflow */  
#define CAN_ERR_CTRL_RX_WARNING 0x04 /* reached warning level for RX errors */  
#define CAN_ERR_CTRL_TX_WARNING 0x08 /* reached warning level for TX errors */  
#define CAN_ERR_CTRL_RX_PASSIVE 0x10 /* reached error passive status RX */  
#define CAN_ERR_CTRL_TX_PASSIVE 0x20 /* reached error passive status TX */  
                                     /* (at least one error counter exceeds */  
                                     /* the protocol-defined level of 127) */  
#define CAN_ERR_CTRL_ACTIVE     0x40 /* recovered to error active state */
```

data[1] = 0x0C = 0x04|0x08 = CAN\_ERR\_CTRL\_RX\_WARNING|CAN\_ERR\_CTRL\_TX\_WARNING

It means the USB2CAN module can't send/receive data properly and reached warning level.

data[1] = 0x30 = 0x10|0x20 = CAN\_ERR\_CTRL\_RX\_PASSIVE | CAN\_ERR\_CTRL\_TX\_PASSIVE

It means the USB2CAN module can't send/receive data too much, USB2CAN module into error status.

Summing up the above, the error frame tell us, USB2CAN module can't get ACK from CAN BUS and can't send data to the CAN Bus. So the CAN Bus may not inexistence or the connection error.

## 附录 4--4.6 Linux 编译 PCAN 字符驱动

Linux 系统下 CAN 设备可被自动识别为 (SocketCAN) netdev。如需使用 PCAN-basic API 或者 PCAN-VIEW Linux 版本，可以通过自行编译安装字符驱动程序(chardev)。

如何检查: 我的 Linux 环境是否已经包含 PCAN 驱动? 在命令行终端中输入

```
grep PEAK_ /boot/config-`uname -r`
```

执行这条命令列出了所有的 PEAK 驱动程序(y =包含在内核中, m =单独的模块)。

如何检查: 我的 PCAN CAN 设备是否已经初始化? 在命令行终端中输入

```
lsmod | grep ^peak
```

如果连接并初始化了 PCAN 的 CAN 接口, 则将至少有一行以 peak\_usb 开头的输出

我们的例程基于 **Ubuntu 18.04 64 位系统**，对于其他系统，  
 请参考以下详细链接

<https://www.peak-system.com/fileadmin/media/linux/index.htm>

	amd64	i386	arm64	armhf	ppc64el
<b>Ubuntu:</b>					
Trusty 14.04 LTS	■	■	■	■	
Xenial 16.04 LTS	■	■	■	■	■
Bionic 18.04 LTS	■	■	■	■	■
Cosmic 18.10	■	■	■	■	
Disco 19.04	■	■	■	■	
Eoan 19.10	■	■	■	■	
Focal 20.04 LTS	■	■	■	■	■
Groovy 20.10	■	■	■	■	■
Hirsute 21.04	■	■	■	■	■
OpenSUSE Tumbleweed	see Xenial				
<b>Debian:</b>					
Wheezy 7.11	■	■	■	■	
Jessie 8.11	■	■	■	■	
Stretch 9.9	■	■	■	■	
Buster 10	■	■	■	■	■
Bullseye 11	■	■	■	■	■

#### 4.6.1 安装 Linux 字符驱动

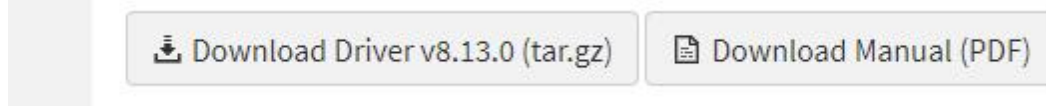
##### Step1, 开发环境必须的安装包

```
sudo apt-get install gcc
sudo apt-get install g++
sudo apt-get install libpopt-dev
```

## Step2,从以下链接下载驱动包

我们使用的驱动包是 V8.13.0，如下图

<https://www.peak-system.com/fileadmin/media/linux/version-history.html>



## Step3, 解压、编译、安装驱动

```
tar -xzf peak-linux-driver-8.13.0.tar.gz
cd peak-linux-driver-8.13.0/
make clean
make
sudo make install
```

注意事项：make 的过程中可能会遇到很多问题，请根据 log 进行下一步操作，如遇到无法编译通过情况，需要告知我们使用的 Linux 版本。

### 4.6.2 PCAN-View Linux 版本安装

Linux 下的 PCAN-VIEW 是一款用于监控显示 CAN 和 CANFD 通讯的简单软件，可以用来发送和接收 CAN 和 CANFD 数据，PCANVIEW 基于 NCurses library。

## 系统环境要求

按 4.3.1 章节安装好字符驱动，详细请参考 [Driver Package for Proprietary Purposes](#)

## 通过 repository 安装 PCANVIEW

Installing software through repository needs first to register the repository only once. Next to the first installation of the software, there is nothing you have to do, except installing available updates when prompted by your system.

**Step1**，下载并安装 peak-system.list 文件

```
wget -q http://www.peak-system.com/debian/dists/`lsb_release  
-cs`/peak-system.list -O- | sudo tee /etc/apt/sources.list.d/peak-system.list
```

安装成功如下图显示:

```
innomaker@innomaker:~/Downloads/peak-linux-driver-8.13.0$ wget -q http://w  
ww.peak-system.com/debian/dists/`lsb_release -cs`/peak-system.list -O- | su  
do tee /etc/apt/sources.list.d/peak-system.list  
  
deb http://www.peak-system.com/debian bionic non-free  
#deb-src http://www.peak-system.com/debian bionic non-free
```

备注: 如果 `lsb_release` 工具没有在你的系统版本中安装, 需要根据你使用的系统分支版本, 使用版本号替代上述命令中的 ``lsb_release -cs`` 内容, 如使用的是 `wheezy`, 请使用以下命令。

```
wget -q http://www.peak-system.com/debian/dists/wheezy/peak-system.list -O- | sudo tee  
/etc/apt/sources.list.d/peak-system.list
```

一般情况下不需要。

## Step2, 下载并安装 public key

```
wget -q http://www.peak-system.com/debian/peak-system-public-key.asc -O- |  
sudo apt-key add -
```

安装成功如下图显示:

```
innomaker@innomaker:~/Downloads/peak-linux-driver-8.13.0$ wget -q http://ww  
w.peak-system.com/debian/peak-system-public-key.asc -O- | sudo apt-key add  
-  
OK
```

## Step3, 安装 Pcanview-ncurses

```
sudo apt-get update
```

```
sudo apt-get install pcanview-ncurses
```

安装成功如下图显示

```
innomaker@innomaker:~/Downloads/peak-linux-driver-8.13.0$ sudo apt-get install pcanview-ncurses
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libegl1-mesa libwayland-egl1-mesa
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  Terminalview-ncurses
0 upgraded, 1 newly installed, 0 to remove and 8 not upgraded.
Need to get 80.7 kB of archives.
After this operation, 169 kB of additional disk space will be used.
Get:1 http://www.peak-system.com/debian bionic/non-free amd64 pcanview-ncurses amd64 0.9.1-0 [80.7 kB]
Fetched 80.7 kB in 1s (66.4 kB/s)
Selecting previously unselected package pcanview-ncurses:amd64.
(Reading database ... 170478 files and directories currently installed.)
Preparing to unpack .../pcanview-ncurses_0.9.1-0_amd64.deb ...
Unpacking pcanview-ncurses:amd64 (0.9.1-0) ...
Setting up pcanview-ncurses:amd64 (0.9.1-0) ...
```

### 4.6.3 Linux PCANVIEW 收发数据

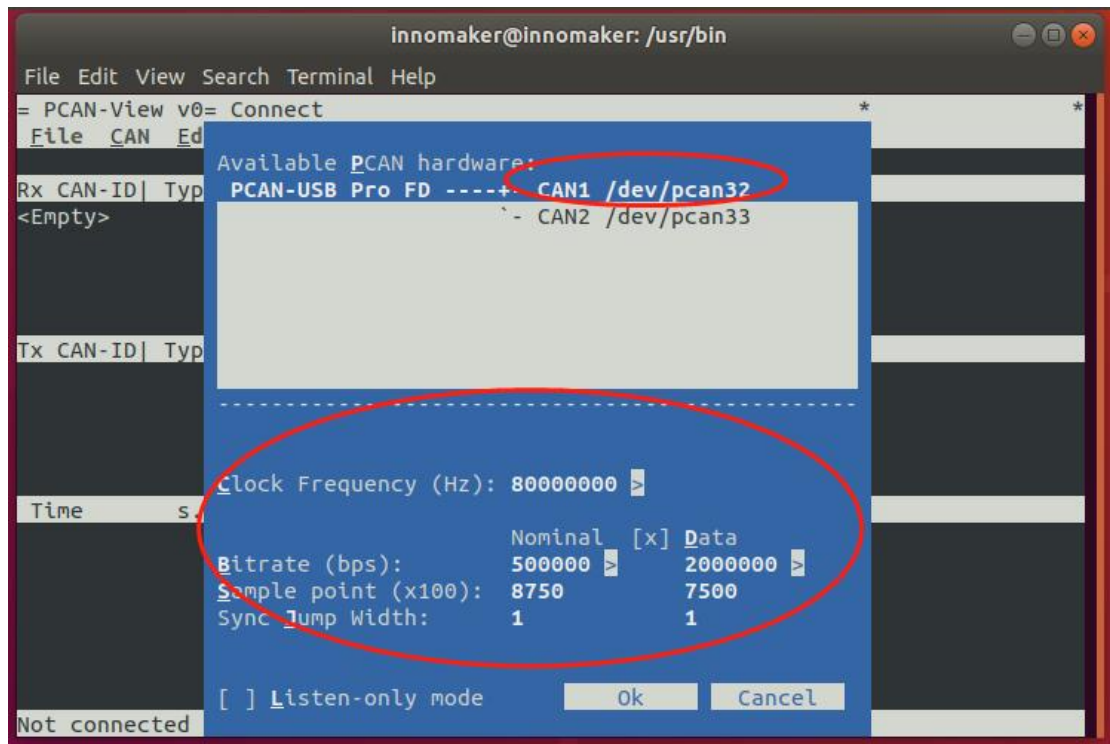
#### Step1 设置通讯参数

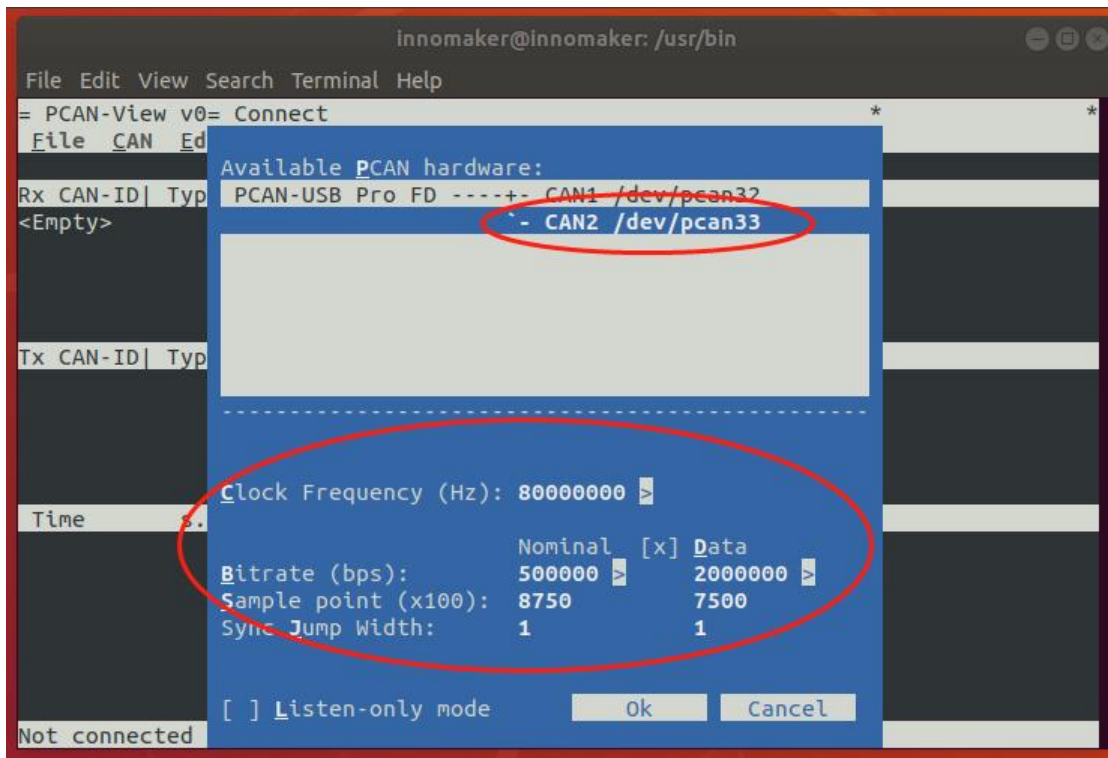
打开两个 Terminal 窗口，分别执行下面指令。1 个用于 CAN1 通讯，1 各用于 CAN2 通讯，

```
cd /usr/bin
```

```
./pcanview
```

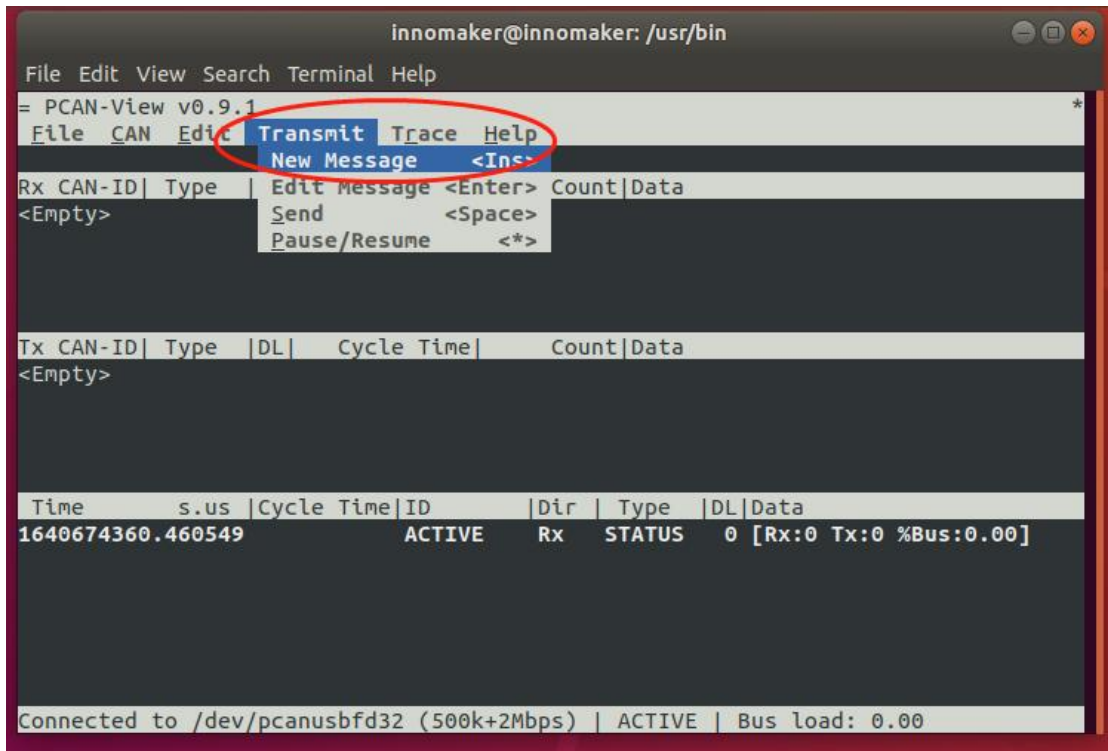
如下图为 CAN1 和 CAN2 选择相同的 Normianl/Data 波特率。





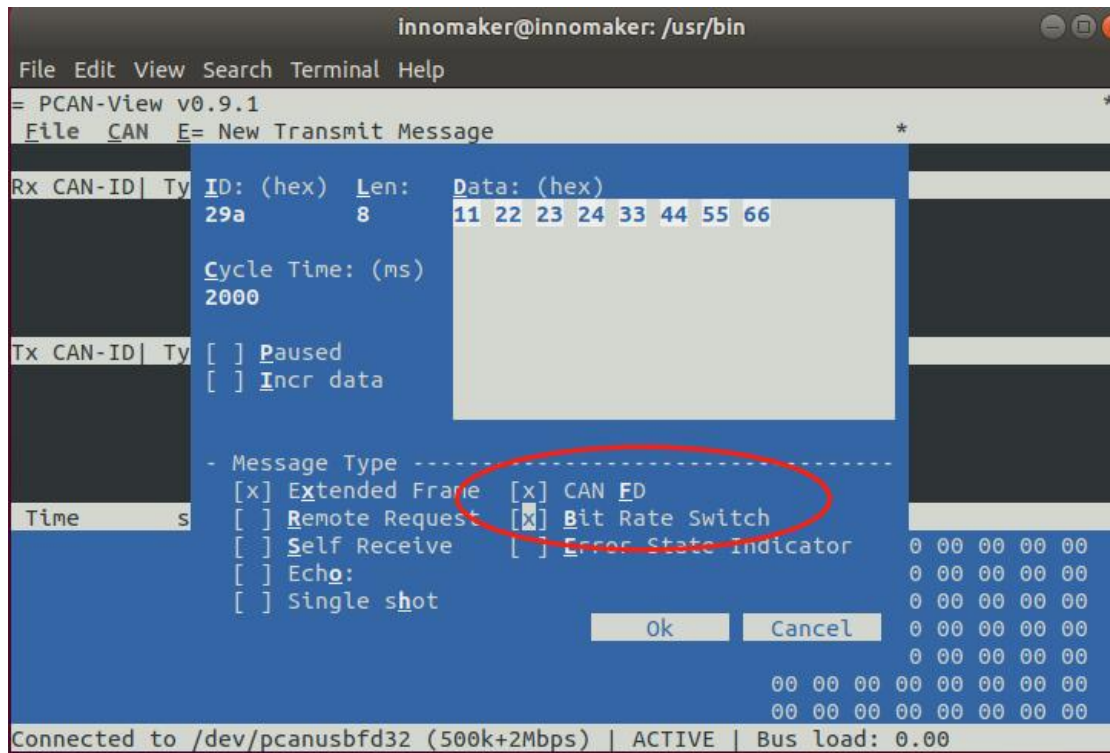
## Step2 发送和接收数据

参照下图，选择 **Transmit**——**New Message** 打开发送数据窗口





弹出对话框如下图所示，切记要选择 Bit rate switch



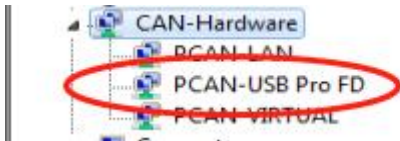
通讯效果如下图所示：



- 驱动目录为：PU2CANFD-C\For\_Windows\Driver
- 软件目录为：PU2CANFD-C\For\_Windows\Peak\_Soft
- 本章节以 USB2CANFD-X2 为例，接线请参考 3.2，LED 指示灯请参考 2.1.3

## 5.1 安装驱动

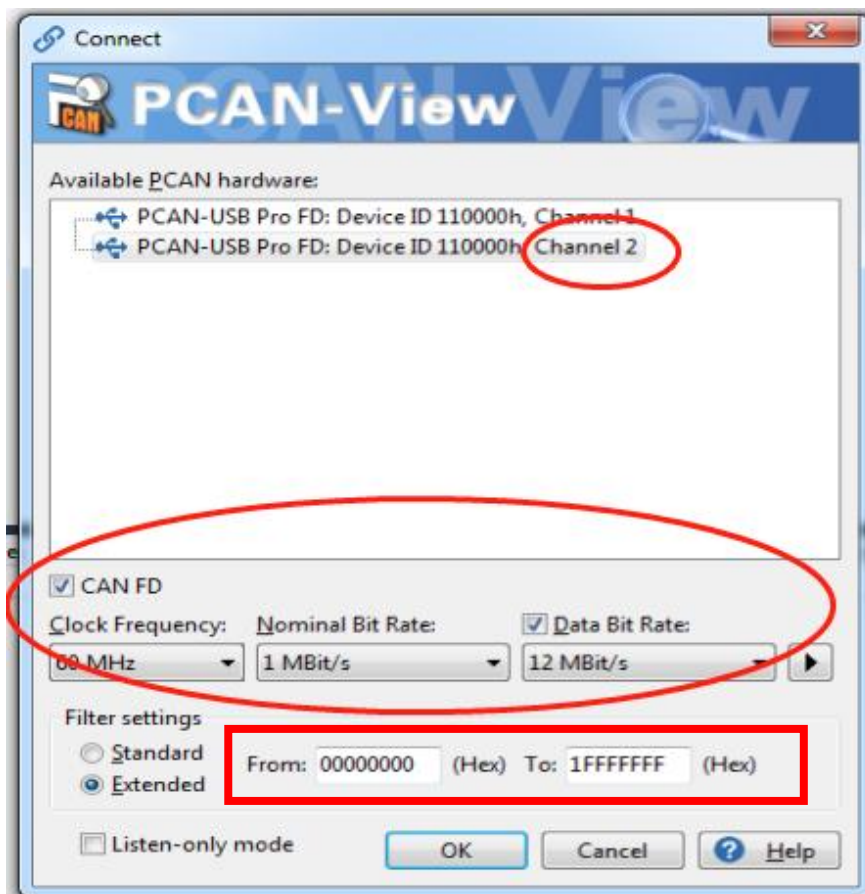
- 连接 USBCANFD-X2 设备到电脑；
- 双击 PeakOemDrv.exe，按照默认设置或根据您的需要配置安装选项然后进行安装。
- 驱动安装成功后 USB2CANFD-X2 设备在设备管理器识别成如下图所示设备。



- 驱动安装完成后设备对应 LINK LED 指示灯开始闪烁；

## 5.2 连接框说明

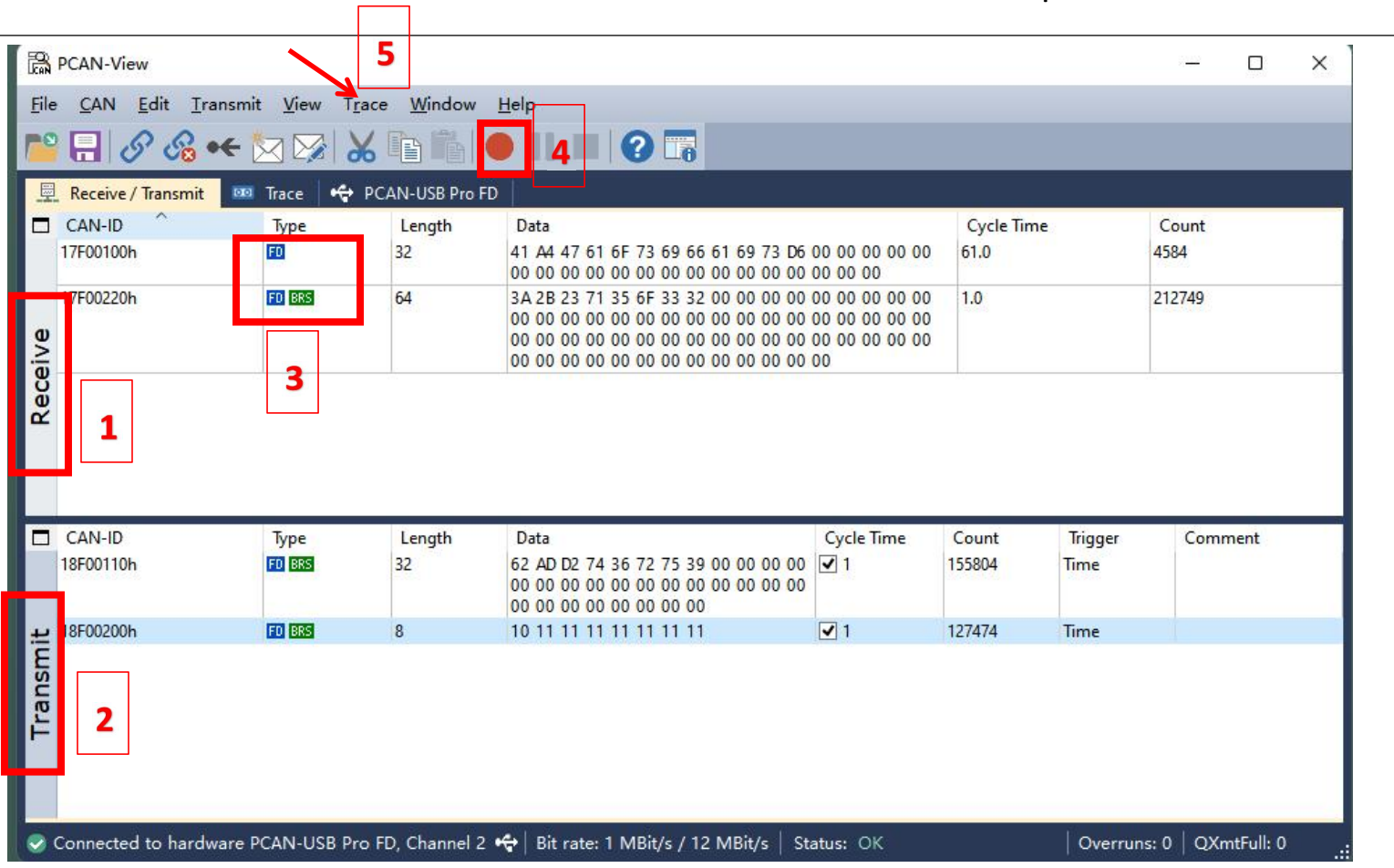
打开 PCAN-VIEW 软件后，出现如下图所示的连接对话框窗口：



- 选项 1, Available PCAN hardware:
  - 可以从窗口中选择要连接的设备接口
- 选项 2, CANFD:
  - 勾选后支持 CANFD
- 选项 3, Clock Frequency:
  - 设置 CAN 时钟频率 (Clock Frequency), 后面的选项波特率(Data Bit Rate) 基于所选的 CAN 时钟频率;
- 选项 4, Nominal Bit Rate:
  - 下拉选择仲裁比特率 (Nominal Bit Rate), 仲裁阶段最大比特率位 1Mbit/s。
- 选项 5, Data Bit Rate:
  - 勾选方框, 选择通讯比特率 (Data Bit Rate) 最大 12M;
- 选项 6, Filter Setting:
  - 可以选择 11 位 ID 的标准帧 (Standard);
  - 可以选择 29 位 ID 的扩展帧 (Extended);
  - 设置过滤器, 接收某个 ID 范围内的报文 (From----to)。
- 选项 7, Listen-only mode:
  - .如果您不主动参与 CAN 流量, 只想观察, 请激活仅收听模式。这也避免了未知 CAN 环境的意外中断 (例如, 由于不同的比特率)。
- 选项 8, 点击 OK 确认设置。

## 5.3 接收报文

如果设备已经连接到 CAN 总线上, 并且总线是有数据传输的话, 在接收区域是可以接收到数据的。如下图中所示。

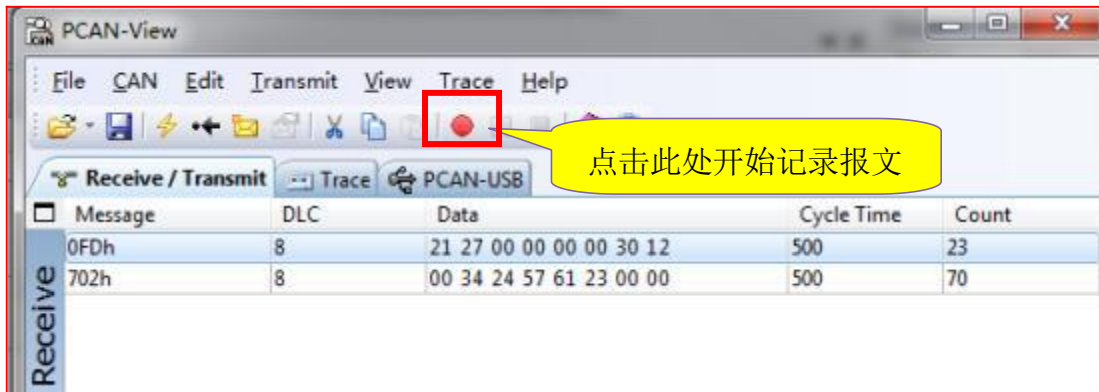


- 1, 报文接收区域
- 2, 报文发送区域
- 3, 报文类型
  - “FD”标识该帧为 CAN FD 格式
  - “BRS”标识该帧启用的数据段可变速率,
  - “ESI”标识该帧的错误标识位为高(指示发送节点处于错误被动状态)
- 4, 记录报文按钮
- 5, 记录报文相关的功能选项

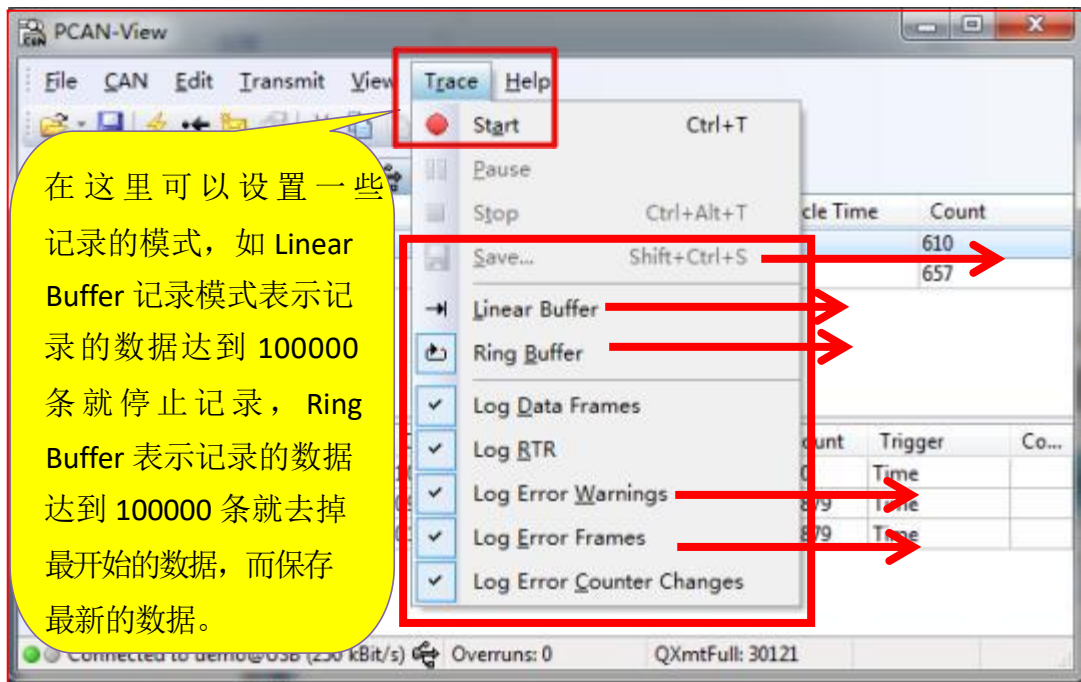
### 5.3.1 记录报文

- 在 Trace 选项卡上,PCAN-View 的数据跟踪器(数据记录器)用于记录 CAN 总线上的通信。
- 在此过程中,消息被缓存到 PC 的工作内存中。然后,可以将它们保存到一个文件中。跟踪程序以线性或环形缓冲模式运行。
- 线性缓冲区模式在缓冲区满时立即停止跟踪器。一旦缓冲区满了,循环缓冲区模式就会用新的消息覆盖旧的消息。
- 记录多达 10 万条报文,包括发送、接收及错误报文,并可保存为 trc 格式的文件,可用记事本打开。并会显示当前的记录状态:记录的总时间、接收报文数量,发送报文数

量，错误数量，缓存占有量（百分比），缓存模式（线性、环形）。

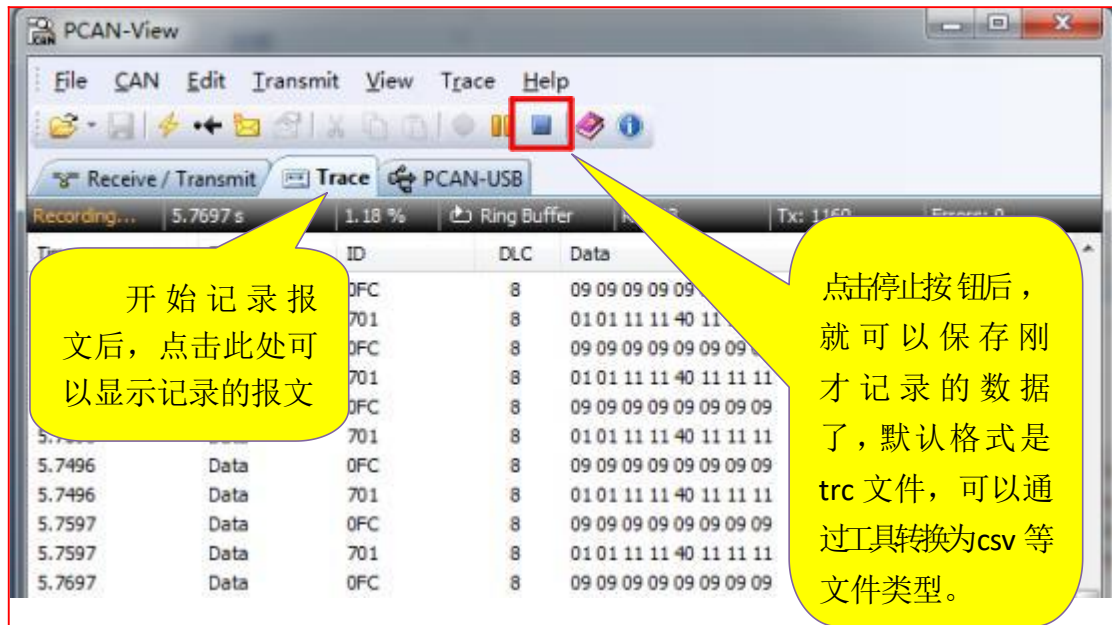


### 5.3.2 显示错误帧和错误计数器



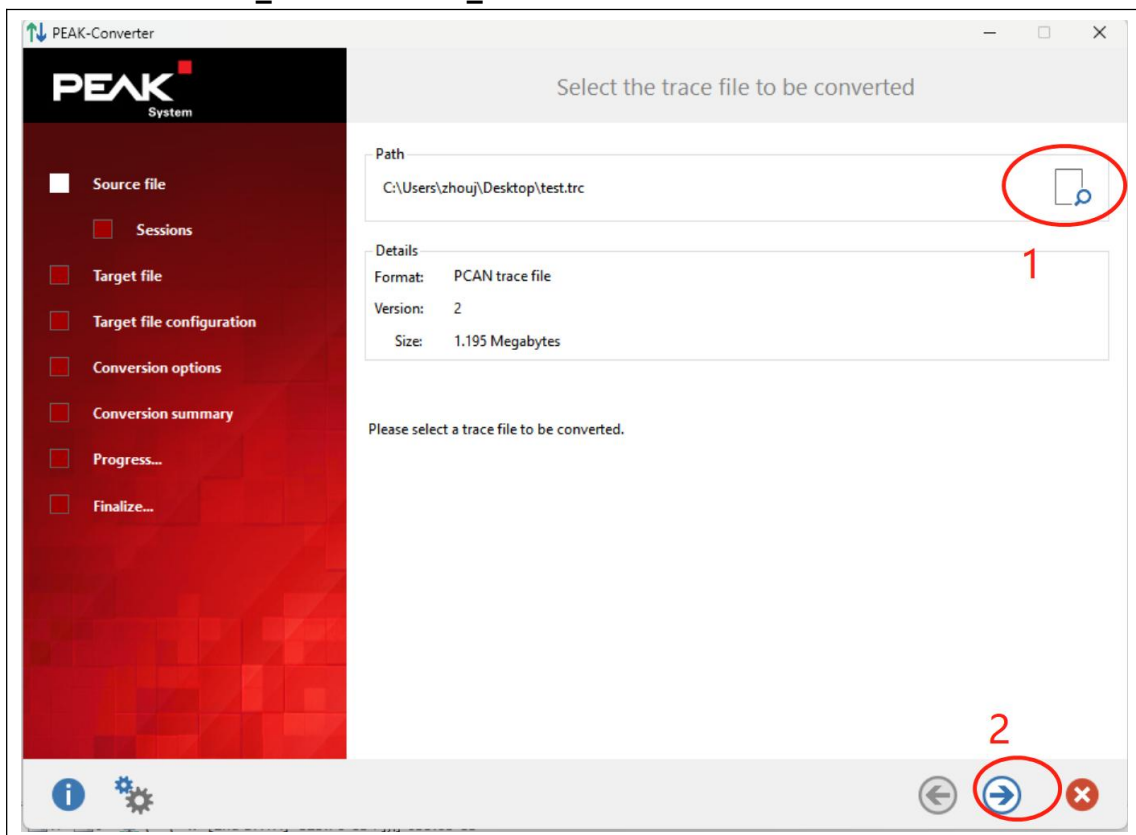
- **Save:** 保存记录的报文
- **Linea Buffer** 记录模式: 表示记录的数据达到 100000 条就停止记录;
- **Ring Buffer:**表示记录的数据达到 100000 条就去掉最开始的数据，而保存最新的数据;
- **Log Error Warnings:** 记录错误帧
- **Log Error Counter Changes:** 记录错误计数器

### 5.3.3 存储报文/转换报文格式



保存的数据的格式是.trc 的文件格式，可以用记事本打开，也可以用 PCAN 的专用转换 PCAN Converter 工具将其转换为 ASC 或者 CSV 格式的文件。

#### PU2CANFD-C\For\_Windows\Peak\_Soft



- 1, 选择需要转换的报文;
- 2, 点击下一步选择需要转换的格式

Select the target file and the location

Type

Format: TRC Description: PCAN trace file

Location

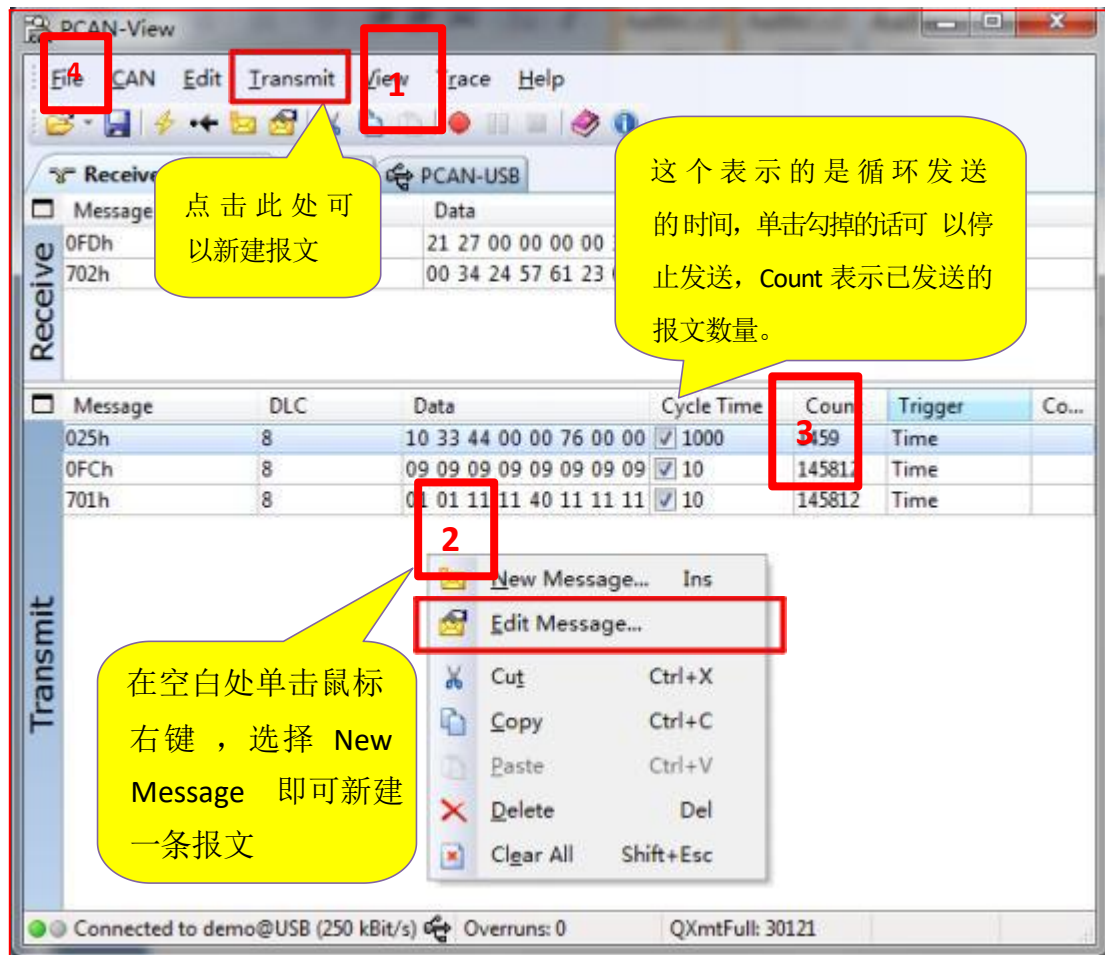
Folder: BTRC

File name: Converted\_test .trc



## 5.4 发送报文

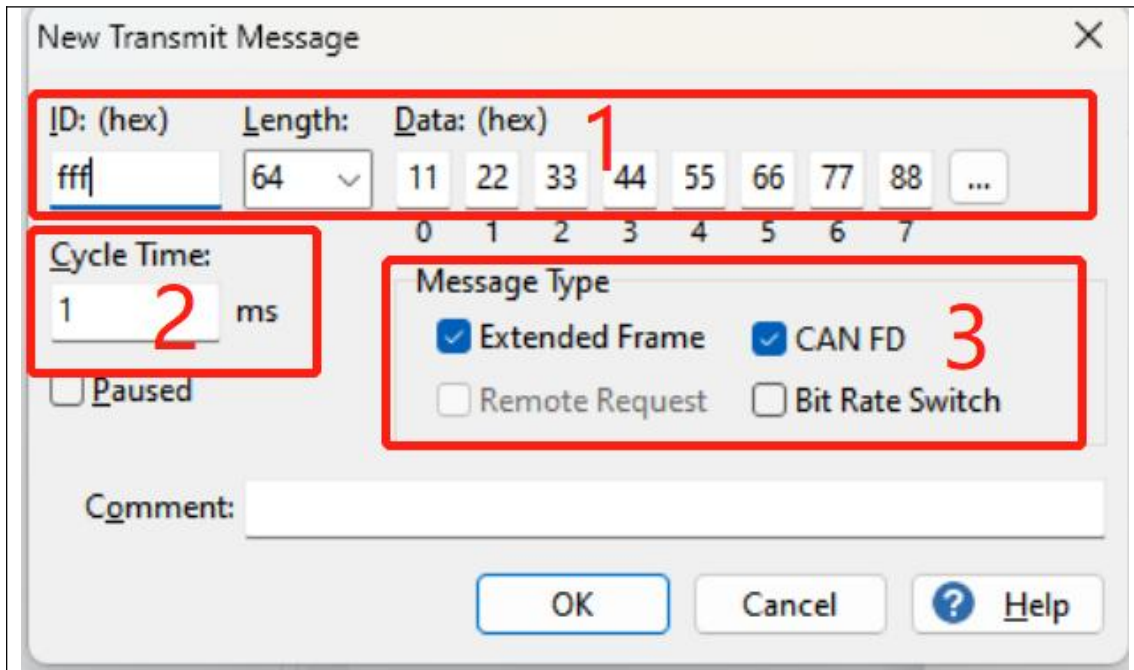
### 5.4.1 报文发送框说明



- 1, 新建报文按钮
- 2, 或者右键点击新建报文;
- 3, 循环发送时间勾选可以直接发送。
- 4, 在发送报文的界面上, 点击软件左上角的保存按钮, 可以将当前 Transmit 框中的 ID 保存为 xml 文件(发送列表), 下次打开可以继续使用。

### 5.4.1 新报文编辑窗口说明

选择菜单命令 **Transmit > New Message** 弹出“New Transmit Message”对话框



选择菜单命令 **Transmit > New Message** 弹出“New Transmit Message”对话框。

选项 1，

- 输入 ID、数据长度和 CAN 消息数据。

选项 2，

- 在“Cycle Time”字段中输入一个值，以选择手动或周期性的消息传输。请输入一个大于 0 的值，以便定期发送。输入值 0 以手动传输。

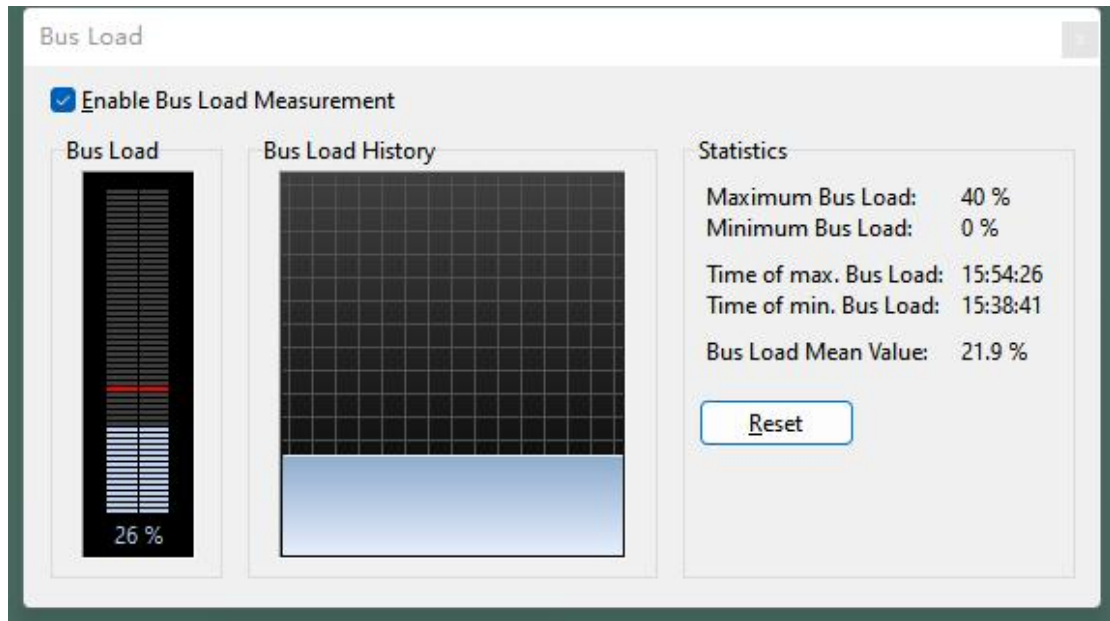
选项 3，

- 若需要发送 CANFD 格式，需要勾选“CAN FD”选项，数据长度最多扩展至 64 字节
- 勾选“CAN FD”选项后，可以选择勾选“Bit Rate Switch”，用于启用数据段切换高速波特率。
- 点击 OK 确认条目。创建的传输消息出现在接收/发送选项卡。

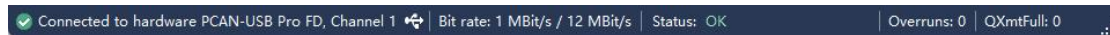
通过菜单命令 **Transmit > Send** 手动触发选定的发送消息 (或者空格键)。另外，也可以对定期传输的 CAN 报文进行手动传输。

## 5.5 总线负载选项

- 在总线负载选项卡上，显示 CAN 通道的当前总线负载，时间过程和统计信息。CAN 总线负载反映了传输容量的利用率。
- 图形化显示当前和历史总线负载，也可以显示这段时间以来的最大总线负载，最小总线负载及其出现的时间，平均总线负载



## 5.6 Status Bar (状态栏)



如上图，状态栏显示有关当前 CAN 连接、错误帧计数器（Overruns、QXmtFull）的信息，并显示错误消息。您可以在帮助中找到有关使用 PCAN-View 的更多信息，您可以通过“帮助”菜单或使用 F1 键在程序中调用该帮助。

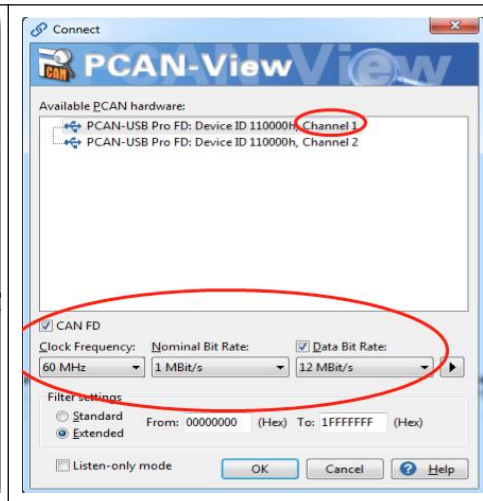
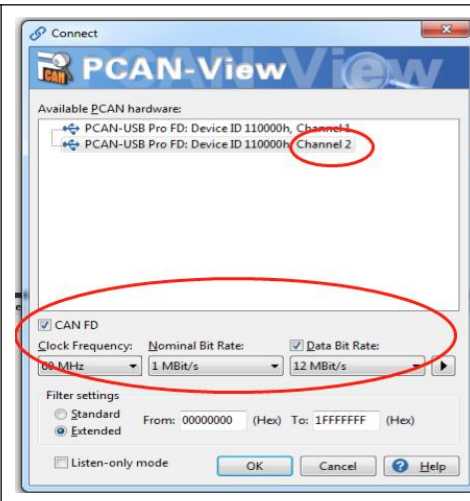
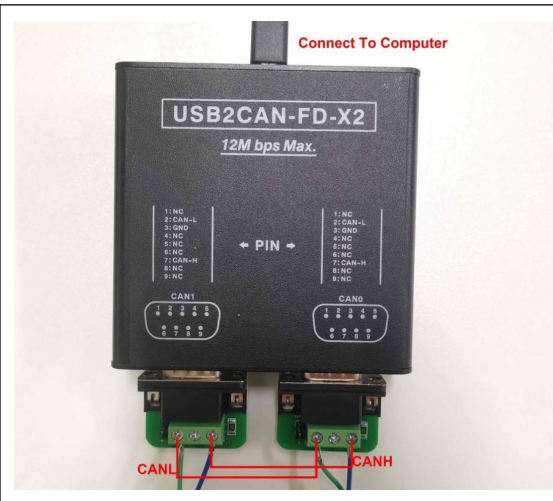
## 5.7 简单通讯案例

本次案例需要同时打开两个 PCAN-VIEW 窗口，并且按下图设置 Channel1, Channel2 参数。需要设置值如下：

- CANFD: 勾选
- Clock Frequency 必须同时被设置为: 60MHz,
- Nominal Bit Rate 设置为: 1MBit/s
- Data Bit Rate 设置为: 12MBit/s

接线、设置

# High Speed USB2.0 To CAN FD Converter Data Field Up to 12M Max Compatible With PCAN Software



通讯效果

Timestamp: 1µs

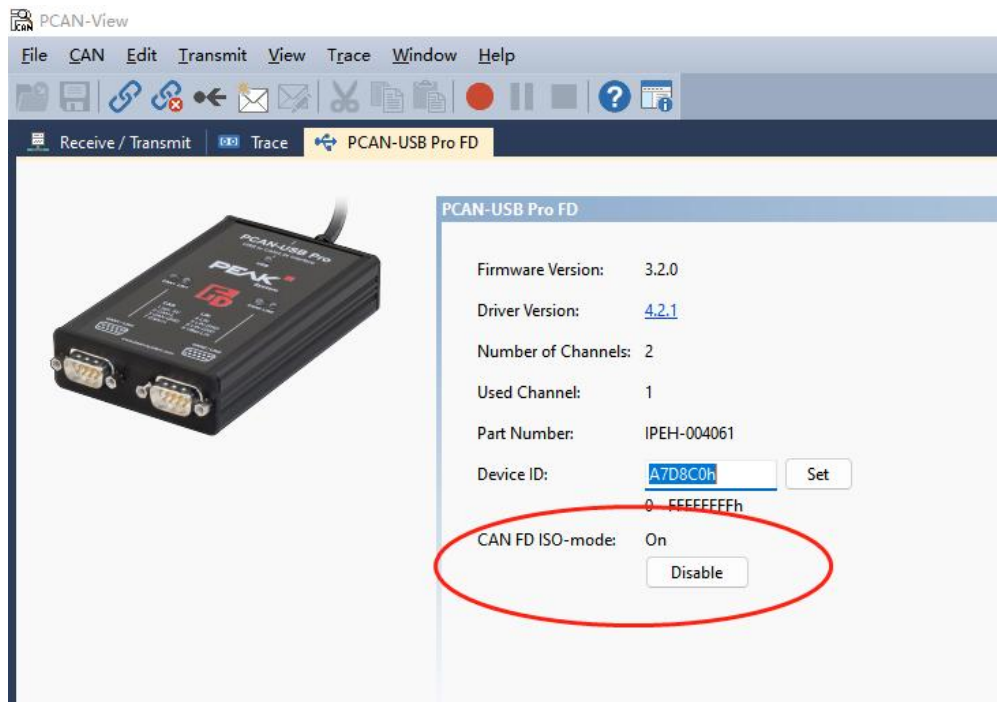
The receiving result is completely consistent with the sending result

15 groups at the same time, 15000frames/s per group  
Each group accumulates 100 million frames

**PASSED**

CAN-ID	Type	Length	Data	Cycle Time	Count	Trigger	Comment
100h	R	8	11 00 00 00 00 00 77	1.0	11504683	Time	
101h	R	8	11 00 00 00 00 00 88	1.0	11504373	Time	
102h	R	8	11 00 00 00 00 00 99	1.0	11504063	Time	
103h	R	8	00 11 00 00 00 00 66 00	1.0	11503753	Time	
104h	R	8	00 11 00 00 00 00 55 00	1.0	11503443	Time	
105h	R	8	12 00 00 00 00 00 00 00	1.0	11503133	Time	
106h	R	8	13 00 00 00 00 00 00 00	1.0	11502823	Time	
107h	R	8	14 00 00 00 00 00 00 00	1.0	11502513	Time	
108h	R	8	15 00 00 00 00 00 00 00	1.0	11502203	Time	
109h	R	8	16 00 00 00 00 00 00 00	1.0	11501893	Time	
10Ah	R	8	17 00 00 00 00 00 00 00	1.0	11501583	Time	
10Bh	R	8	18 00 00 00 00 00 00 00	1.0	11501273	Time	
10Ch	R	8	19 00 00 00 00 00 00 00	1.0	11500963	Time	
10Dh	R	8	20 00 00 00 00 00 00 00	1.1	11500653	Time	
10Eh	R	8	21 00 00 00 00 00 00 00	1.0	11500343	Time	
10Fh	R	8	22 00 00 00 00 00 00 00	1.0	11500033	Time	

## 5.8 ID 设置选项卡



- PCAN-USB FD Pro 选项卡包含有关硬件和驱动程序的一些详细信息。此外，还可以将设备 ID 分配给适配器。因此，在同时在计算机上操作多个 PCAN-USB Pro FD 适配器时，可以唯一地识别它。
- 要识别 PCAN-USB Pro FD 适配器，请首先转到用于选择 PCAN-View 硬件的对话框。在"可用的 PCAN 硬件和 PCAN 网络"列表中，您可以右键单击每个 USB 适配器并执行命令"识别"。因此，相应适配器的 LED 会很快闪烁。
- **CAN FD ISO-mode** ISO 11898 标准中定义的与原始协议不兼容。通过支持两个协议版本的 CAN FD 接口来考虑这一点。如果需要，用户可以使用启用/禁用按钮 ("非 ISO" 和"ISO") 切换到环境中使用的 CAN FD 协议。参考上图中的红色圆圈设置

## 6 PCAN-Explorer 使用说明

PCAN-Explorer 属于付费软件，需要自行采购密匙并且安装使用

### 6.1 PCAN-Explorer5 基本使用方法



PCAN-Explorer5  
基本使用方法new.

### 6.2 PCAN-Explorer5 入门手册



PCAN-Explorer5  
入门手册.pdf

## 6.3 J1939 基本使用方法



J1939基本使用方  
法new.pdf

## 6.4 Symbol Editor 基本使用方法



Symbol  
Editor基本使用方

## 6.5 Plotter 插件基本使用方法



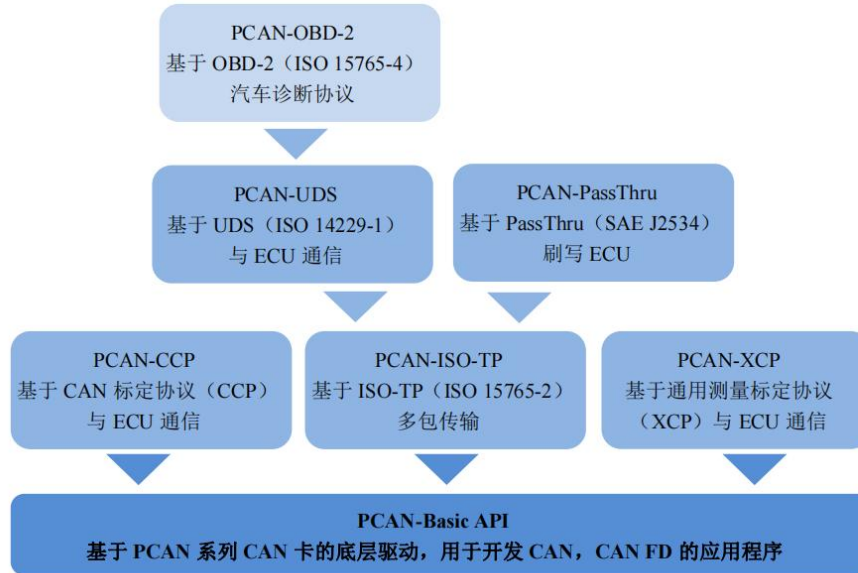
Plotter插件基本  
使用方法new.pdf

## 6.6 Instrument Panel 插件基本使用方法



Instrument  
Panel插件基本使

## 7 免费软件编程接口介绍



从上图可以看出，基于 PCAN-USB 产品，我们主要提供了基础的 CAN 通信开发包 PCAN-Basic; 用于 ECU 标定的 CCP 和 XCP 开发包; 用于诊断方面的 ISP-TP, UDS, OBD-2 开发包。以上 API 都是免费提供的。

更多资料请参考：<https://www.peak-system.com>

下面简要介绍一个各个开发包的主要功能：

### 7.1 PCAN-Basic API

PCAN-Basic API（应用程序接口）是用于 PCAN 硬件接口系列的二次开发的应用程序接口。它允许开发简单的 CAN 应用，以实现和我们的 PCAN-PC 硬件通信。API 包括实际的设备驱动和一个提供 API 函数接口的 DLL（动态链接库）。

PCAN-Basic 为开发者提供了各种环境下的多种函数，包括 C#, C++/CLR, Delphi, VB.NET, Java, 和 Python 2.6，在开发包中都有这些环境下的例程。

关于 LabView，我们没有免费的 LabView 驱动 VI，客户可向我们购买，或者自己根据 DLL 来编写 LabView 驱动。PCAN-Basic API 还可用于 WinCE 6.x，目前可使用的编程语言包括 C++, C#, and VB.NET。

特性：

- 支持 Windows8/7/Vista/XP (32/64 位)和 WinCE 6.x 操作系统（注：ISA，并口和 PC 卡 CAN 接口只支持 32 位系统）

- 多个 PEAK 公司的和你自己的应用程序可以在一个物理 CAN 通道上面同时运行
- 单个 DLL 可支持所有的硬件类型
- 为每个硬件单元可使用多达 8 个通道（取决于所采用的 PEAK CAN 接口）
- PCAN PC 硬件通道间可进行简单的切换
- 每个 CAN 通道有 32,768 消息的内部驱动缓冲
- 接收报文的时间精度可达 1  $\mu$ s（取决于所采用的 PCAN 接口）
- 可访问一些硬件参数，比如只听模式
- 通过 Windows Events 通知已经接收到消息
- 一个扩展系统可用于调试操作
- 语言支持包括德语，英语，法语，西班牙语和意大利语
- 输出语言取决于操作系统
- 可自定义调试信息

## 7.2 PCAN-CCPAPI 与 PCAN-XCPAPI

PCAN-CCP API 是 Windows®应用程序（主站）和电子控制单元（从站 ECU）之间通讯的编程接口。API 基于 ASAM 规定的 CAN 标定协议（CCP），主要用于汽车电子开发。

通用测量和标定协议（XCP）是 CCP 更深层次的开发协议，但是两者不兼容。XCP 支持多个传输介质（CAN，以太网，USB，Flexray）。我们相应的编程接口叫作 PCAN-XCP API，它采用 CAN 总线作为传输介质，类似于 PCAN-CCP API。

以上两种 API 都使用编程接口 PCAN-Basic 访问电脑上的 CAN 硬件。PCAN-Basic 已经包含在 PEAK-System 公司的每一个 CAN 接口中。都是免费的。

特点：

- Windows DLLs for 32-bit 和 64-bit 应用程序
- 使用我们的 CAN 接口可通过 CAN 进行物理通讯
- 使用 PCAN-Basic API 可访问电脑上的 CAN 硬件
- Thread-safe API（线程安全的 API）
- 一个 API 功能用于 CCP/XCP 标准上的每个命令
- 附加命令用于通讯管理

## 7.3 PCAN-ISO TPAPI

ISO-TP (ISO 15765-2) 是一项国际标准，用于通过 CAN 传输数据包。在 CAN (OSI 层 1 和 2) 上面，该协议覆盖 OSI 层 3（网络层）和 4（传输层）。它每个数据包能够传输最大 4095 字节的 CAN 报文。数据字节使用 CAN 多帧方式分段传输。

PCAN-ISO-TP API 的执行基于 10 个功能函数基础的标准功能性。它们被分类为分配、配



置、地址映射配置、信息、和通讯。

PCAN-ISO-TP 使用 PCAN-Basic 编程接口访问电脑上的 CAN 硬件。PCAN-Basic 和每个 PCAN 系列 CAN 接口一起提供。

特点:

- ISO-TP 协议(ISO 15765-2)的执行用于通过 CAN 执行传输最多 4095 字节的数据包
- Windows DLLs 用于开发 32-bit 和 64-bit 应用程序
- 用 PCAN 系列 CAN 接口通过 CAN 总线进行物理通讯
- 用 PCAN-Basic API 访问电脑上的 CAN 硬件

## 7.4 PCAN-UDS API

UDS (ISO 14229-1) 标准用于统一的诊断服务和定义控制器 (ECU) 的通讯。

Windows 软件用各种服务测试控制器。这个过程在客户服务器上完成, 程序原则上代替客户端 (也叫作测试者)。UDS 使用 ISO-TP 标准作为传输协议, 因此 UDS 可传输最大 4095 字节的数据块。除了交换维护信息之外, 例如, 还能够传输固件。

PCAN-UDS API 执行基于 8 个功能函数基础的标准功能性。它们被分类为测试仪分配、配置、信息、Utilities、服务、和通讯。

特点:

- UDS 协议 (ISO 14229-1) 的执行用于控制器通讯
- Windows DLLs 用于开发 32-bit 和 64-bit 应用程序
- 用 PCAN 系列 CAN 接口通过 CAN 总线进行物理通讯
- 用 PCAN-Basic API 访问电脑上的 CAN 硬件
- 用 PCAN-ISO-TP API (ISO 15765-2)
- 通过 CAN 总线传输最多 4095 字节的数据包

## 7.5 PCAN-OBD-2 API

对于车载诊断, OBD-2 标准定义了特定车辆参数的交换标准。客户端会向车辆上的控制器 (ECU) 发出请求: 哪一个或几个 ECU 正在应答。作为 OBD-2 的一部分, ISO 15765-4 标准描述 CAN 总线作为传输选项。

PCAN-OBD-2 API 执行基于 15 个功能函数基础的标准功能性。它们被分类为测试仪分配、配置、地址映射配置、服务、和通讯。

依照 ISO 15765-4, OBD-2 基于 UDS。以此类推, PCAN-OBD-2 使用 PCAN-UDS 编程接口用于诊断数据的交换。

特点:

- OBD-2 协议 (ISO 15765-4) 的执行作为车载诊断标准
- Windows DLLs 用于开发 32-bit 和 64-bit 应用程序
- 用 PCAN 系列 CAN 接口通过 CAN 总线进行物理通讯
- 用 PCAN-Basic 编程接口访问电脑上的 CAN 硬件
- 用 PCAN-ISO-TP API (ISO 15765-2) 通过 CAN 总线传输最多 4095 字节的数据包
- 使用 PCAN-UDS API (ISO 14229-1) 用于控制器 (ECU) 通讯

## 7.6 PCAN-PassThru API

对控制器 (ECU) 编程, 有无数应用程序来自于各个厂家, 它们被用于开发和诊断车辆 电子系统。在这些应用程序和控制器 (ECU) 之间的通讯接口由国际标准 SAE J2534 (Pass-Thru) 来定义。因此, 选择连接到控制器的硬件时可以考虑它的厂家。

PCAN-PassThru 可使用基于我们的 CAN 适配器开发 SAE J2534 应用程序。

SAE J2534 标准定义的相关功能都集成在 Windows DLLs (32 和 64 位系统) 中; 基于此可用于开发自己的 Pass-Thru 应用程序。

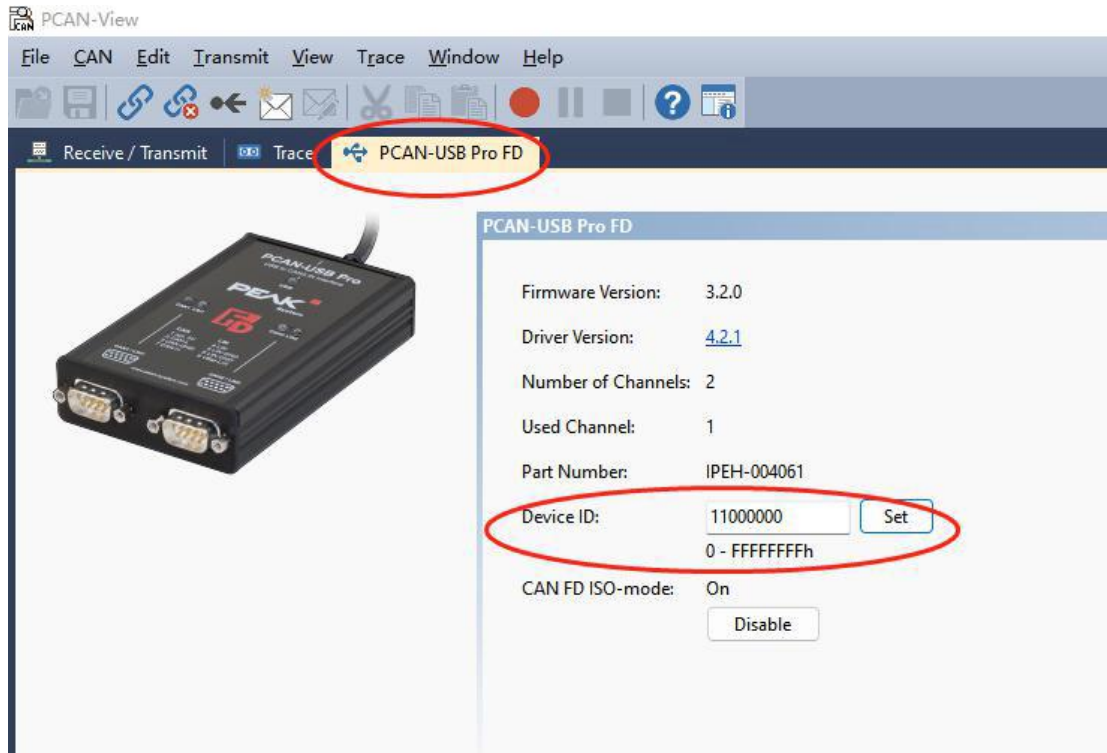
特点:

- 基于国际标准 SAE J2534 (PassThru)
- Windows DLLs 用于开发 SAE J2534 应用程序 (32-bit 和 64-bit) 线程安全 API
- 用 PCAN 系列 CAN 接口通过 CAN 总线进行物理通讯
- 用 PCAN-Basic 编程接口在电脑上访问 CAN 硬件
- 用 PCAN-ISO-TP API (ISO 15765-2)
- 通过 CAN 总线传输最多 4095 字节的数据包

## 附录 A-ID 设置使能/禁用终端电阻

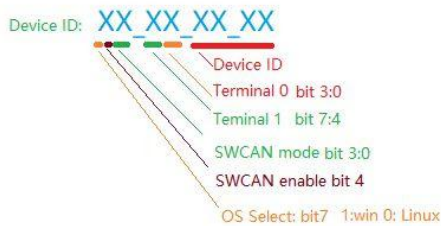
PU2CANFD 系列产品默认出厂使能 120 欧姆终端电阻, 可以通过 PCANVIEW 软件设置 ID 使能/禁用内置 120 欧姆终端电阻。

参考以下图片进入设置界面



**ID Setting Description is as below:**

Device serial number have 4 bytes: Byte3|Byte2|Byte1|Byte0  
**Device serial number byte 2 used to control 2 channels of terminal resistor.**  
**BYTE2 bit0—3 control channel0--- 1: resistor on 0 :terminal resistor off**  
**BYTE2 bit7—4 control channel1--- 1: resistor on 0 :terminal resistor off**  
**Note: 设置完成后, 需要重新拔插设备才能生效**



**A1 针对设备在 Windows 系统下使用的 ID 设置**

ID	Description
80FF0000h	CAN0 TERM Enable CAN1 TERM Enable

80000000h	CAN0 TERM Disable CAN1 TERM Disable
800F0000h	CAN0 TERM Enable CAN1 TERM Disable
80F00000h	CAN0 TERM Disable CAN1 TERM Enable

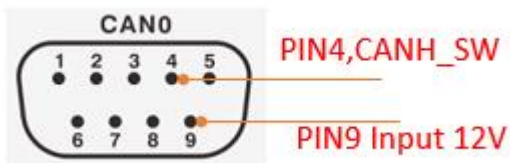
## A2 针对设备在 Linux 系统下使用的 ID 设置

ID	Description
08FF0000h	CAN0 TERM Enable CAN1 TERM Enable
08000000h	CAN0 TERM Disable CAN1 TERM Disable
080F0000h	CAN0 TERM Enable CAN1 TERM Disable
08F00000h	CAN0 TERM Disable CAN1 TERM Enable

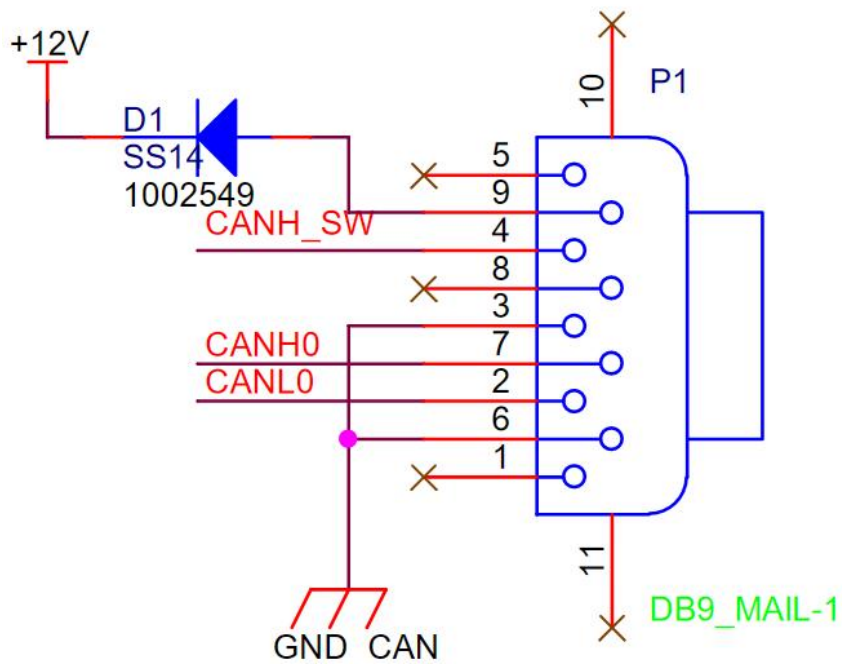
## A3 SW CAN 模式设置

(仅 USB2CANFD-X2 有 SWCAN 功能)

USB2CANFD-X2 CAN0 支持 SWCAN,使能 SWCAN 功能，硬件需要给 CAN0 的 PIN9 供电 12V 并且配合 ID 设置使能，则 PIN4 实现 CANH\_SW 功能



硬件示意图



## ID 设置

ID	SWCAN MODE:	Description
F0000000h	Mode0	0 : sleep
F1000000h	Mode1	1: high speed mode (83.33kbit/s)
F2000000h	Mode2	2: high volage wake up
F3000000h	Mode3	3: normal mode (33.33kbit/s)