

USB2CANFD-X2

高速 USB2.0 转换双路 CANFD 通讯设备

USB2CANFD-X2 用户手册

V1.0



版本	修改内容	联系方式
V1.0	初始版本	
V2.0		

# 1. 产品介绍

## 1.1 系列产品

产品名称	CAN 版本	通道	波特率 MAX	电气隔 离	最高帧率	接口类型
USB2CANFD-X2	CAN2.0A/B CANFD 1.0	2	12Mbit/s	2500V	15000fps	<ul style="list-style-type: none"><li>● USB Type A</li><li>● USB Type B</li><li>● 2*D-SUB 9PIN</li></ul>
USB2CANFDX2-MPCIE	CAN2.0A/B CANFD 1.0	2	12Mbit/s	2500V	15000fps	<ul style="list-style-type: none"><li>● USB Type A</li><li>● USB Type B</li><li>● 2*3P 连接器</li><li>● MINIPCIE</li></ul>
USB2CANFD-X2-Core	CAN2.0A/B CANFD 1.0	2	12Mbit/s	2500V	15000fps	<ul style="list-style-type: none"><li>● MINIPCIE</li></ul>

## 1.2 描述

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

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.

新的 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。

## 1.3 产品特点

- USB2.0 高速模式(兼容 USB1.1、USB2.0、USB3.0)
- 符合 CAN 2.0A(11 位 ID)、2.0B(29 位 ID)、CANFD1.0;
- 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。同时配备两个外接 DB9 端子配备 120 欧姆终端电阻, 可通过跳线帽激活;
- 支持 CAN 时钟设置;
- 总线负载测量包括错误帧和过载帧
- 为进入和外出的 CAN 报文诱导错误发生
- 通过 USB 供电,工作温度范围-40—85°C (-40 to 185 °F)
- 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>

## 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

## 供货清单

- 带铝壳的 USB2CANFD-X2 设备
- Windows PCANVIEW 软件, Linux PCANVIEW 软件 (安装使用说明), CAN-UTILS (安装使用说明), C 程序 (含源码), Python 程序 (含源码)
- 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)
- 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
- PDF 版本用户手册

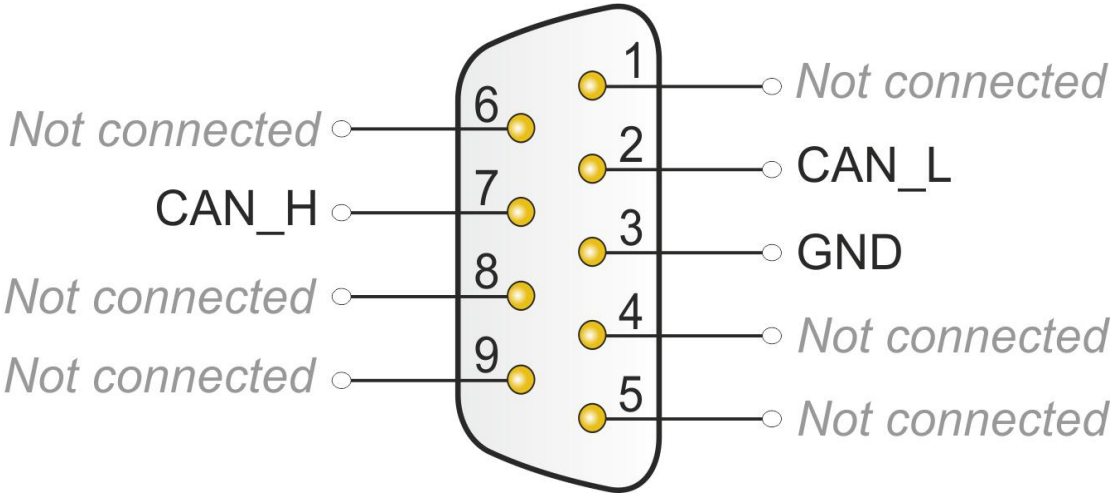
## 1.4 技术参数表

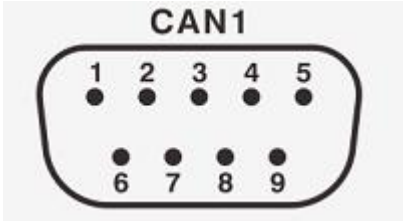
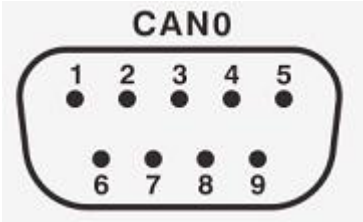
接口	
CANFD 接口	通过两路 9PIN DB9 接口引出
USB 接口	USB plug type A (电脑端) USB plug type B (设备端)
技术特性	
CAN 规格	CAN 2.0A (标准帧 11 位 ID, 8 位数据) CAN 2.0B (扩展帧 29 位 ID, 8 位数据) CAN FD ISO 11898-1:2015 CAN FD non-ISO
CAN 比特率	CAN 支持 25 kbit/s- 1 Mbit/s
CANFD 比特率	支持 25 kbit/s up to 12Mbit/s
电气隔离参数	双路 CANFD 通道每路单独配备高达 2500 伏特的数据和电压磁耦隔离
控制器	高达 180MHZ Cortex-M4 单片机控制器
时间戳分辨率	最低可达 1 $\mu$ s
终端电阻	内置 120 欧姆可通过软件配置的终端电阻
提供软件	Windows PCAN-VIEW Linux PCAN-VIEW (手册) Linux SOCKET-CAN: CAN Utils 工具, C 程序 (源码及说明), Python 程序 (源码及说明)
第三方软件	LabView, CodeSys, Matlab, BUSMASTER, EasyMotion Studio, CANmoon, XX-SCAN, PCAN-Explorer5
PCAN BASIC 应用程序开发接口 (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
其他	
工作温度	-40°~ 85°
尺寸	84x80x28 mm
重量	190 克

## 2 硬件连接及 LED 指示灯说明

两路高速 CANFD, 每路通过 DB9 连接器引出, CAN 的引脚分配符合 CiA 303-1 规范, 9 针 D-SUB 引脚分配如下图。

2.1 DB9 连接器引脚说明



<p><b>CAN1</b></p> 	<p><b>引脚说明</b></p> <table><tr><td>1</td><td>NC</td></tr><tr><td>2</td><td>CANL bus line (dominant low)</td></tr><tr><td>3</td><td>CAN_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>CANH bus line (dominant high)</td></tr><tr><td>8</td><td>NC</td></tr><tr><td>9</td><td>NC</td></tr></table>	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
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																		
<p><b>CAN0</b></p> 	<p><b>引脚说明</b></p> <table><tr><td>1</td><td>NC</td></tr><tr><td>2</td><td>CANL bus line (dominant low)</td></tr><tr><td>3</td><td>CAN_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>CANH bus line (dominant high)</td></tr><tr><td>8</td><td>NC</td></tr><tr><td>9</td><td>NC</td></tr></table>	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
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																		

# USB2CANFD-X2

高速 USB2.0 转换双路 CANFD 通讯设备

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。

## 2.2 120 欧姆终端电阻使能方式

### 通过硬件使能



我们随货提供了 2 片 DB9 转凤凰端子的转接板，附带 120 欧姆终端电阻，可以通过如图红色圈内跳线帽配置使能。（注意如果使用硬件使能，软件使能要关闭）



## 通过软件使能

设备支持软件使能 120 欧姆终端电阻。确保去掉转接板上的跳线帽。

### 2.2.1 设计原理

Device serial number byte 2 used to control 2 channels of terminal resister.  
BYTE3 bit0—3 control channel0--- 1: resister on 0 :terminal resister off  
BYTE3 bit7—4 control channel1--- 1: resister on 0 :terminal resister off

### 2.2.2 案例说明

请见设备背面指示灯，TERM 下对应两个灯变绿。



备注：一定要重新上电，即重新插拔 USB 才能生效。

设置参数对照表及原理

## 2.3 LED 指示灯说明



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）

## USB2CANFD-X2

高速 USB2.0 转换双路 CANFD 通讯设备

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

### 3 Windows PCANVIEW 使用说明

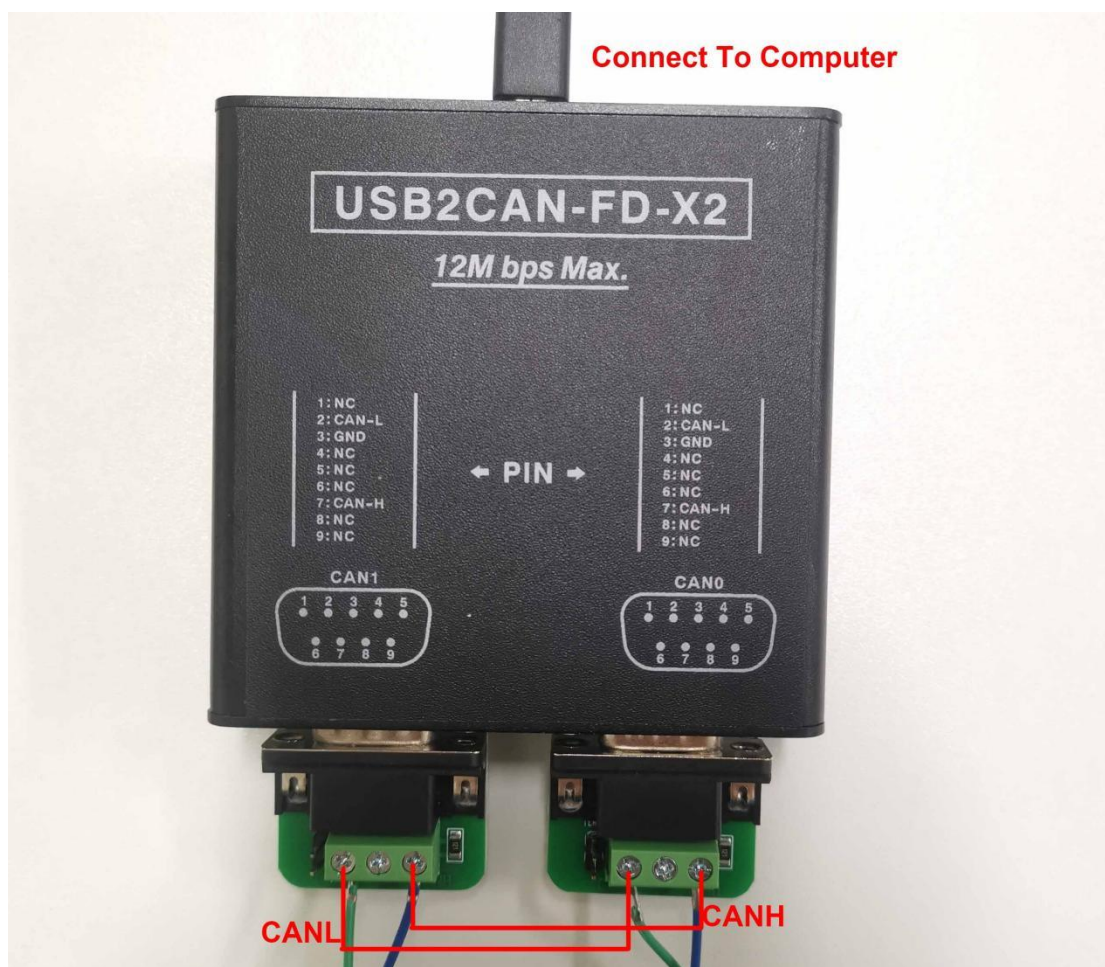
参照下图使用提供的线材连接设备，将设备连接器到电脑

本章涵盖了 USB2CANFD-X2 适配器的软件设置与计算机的连接。在适配器连接到计算机之前，先安装驱动程序。



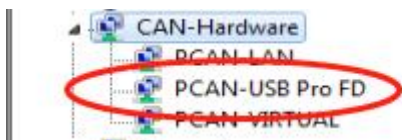
# USB2CANFD-X2

高速 USB2.0 转换双路 CANFD 通讯设备



## 3.1 驱动安装

- 连接 USB2CANFD-X2 设备到电脑后，将会自动识别安装驱动程序
- 如果无法自动安装，解压提供的 PEAK-System\_Driver-Setup.zip，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。



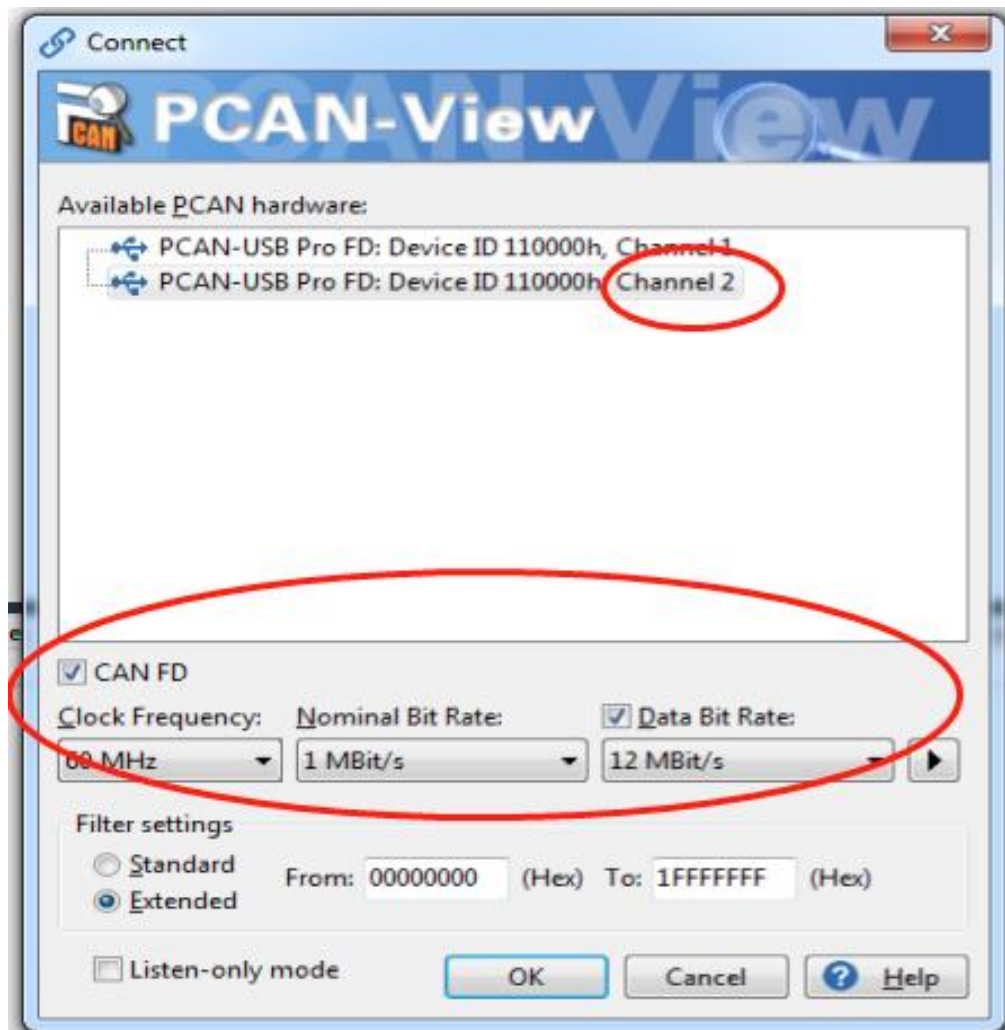
此时背后的 LINK LED 指示灯会闪烁。

## 3.2 使用 PCAN-VIEW 软件进行通讯

PCAN-View 软件是一个适用于 Windows 的 CAN 报文监视器，可同时接收，发送和记录 CAN 报文。支持 CAN FD，CAN 2.0A 和 2.0B 协议，最大波特率可达 12 Mbit/s。连接窗口会显示当前连接的硬件，并可设置波特率，过滤器等参数。

### 3.2.1 连接对话框界面说明

选项 1，打开 PCAN-VIEW 软件后，出现如下图所示的连接对话框窗口；



选项 2，如上图所示，可从窗口中选择要连接的设备接口。

选项 3，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，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，使能（方框√上）通讯比特率（Data Bit Rate）

选项 6，从下拉列表中为 CANFD 选择数据通讯比特率。

选项 7，在 Filter Setting 设置中，可以设置使用标准帧或者扩展帧，并且可以设置 CAN ID 的接收范围。可以选择 11 位 ID 的标准帧（Standard）并设置范围，或者选择 29 位 ID 的扩展帧（Extended）并设置接收范围

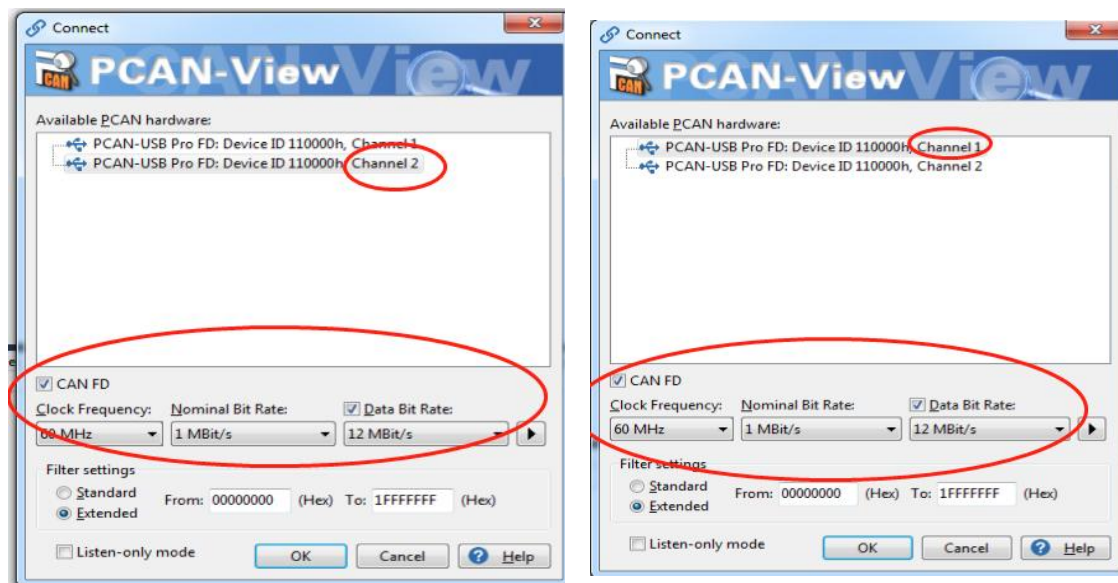
选项 8，步骤 8.如果您不主动参与 CAN 流量，只想观察，请激活仅收听模式。这也避免了未知 CAN 环境的意外中断（例如，由于不同的比特率）。

选项 9，点击 OK 确认设置。将会出现 PCAN-VIEW 的主要窗口。

### 3.2.2 软件使用案例说明

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 设置为：60MHz，
- Nominal Bit Rate 设置为：1MBit/s
- Data Bit Rate 设置为：12MBit/s



### 3.2.3 通过软件使能 120 欧姆终端电阻

如下图，参照章节 2.2，通过软件使能内置 120 欧姆终端电阻。

# USB2CANFD-X2

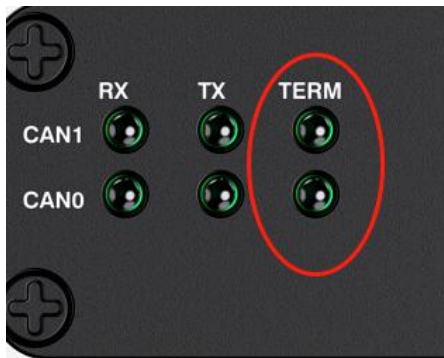
高速 USB2.0 转换双路 CANFD 通讯设备

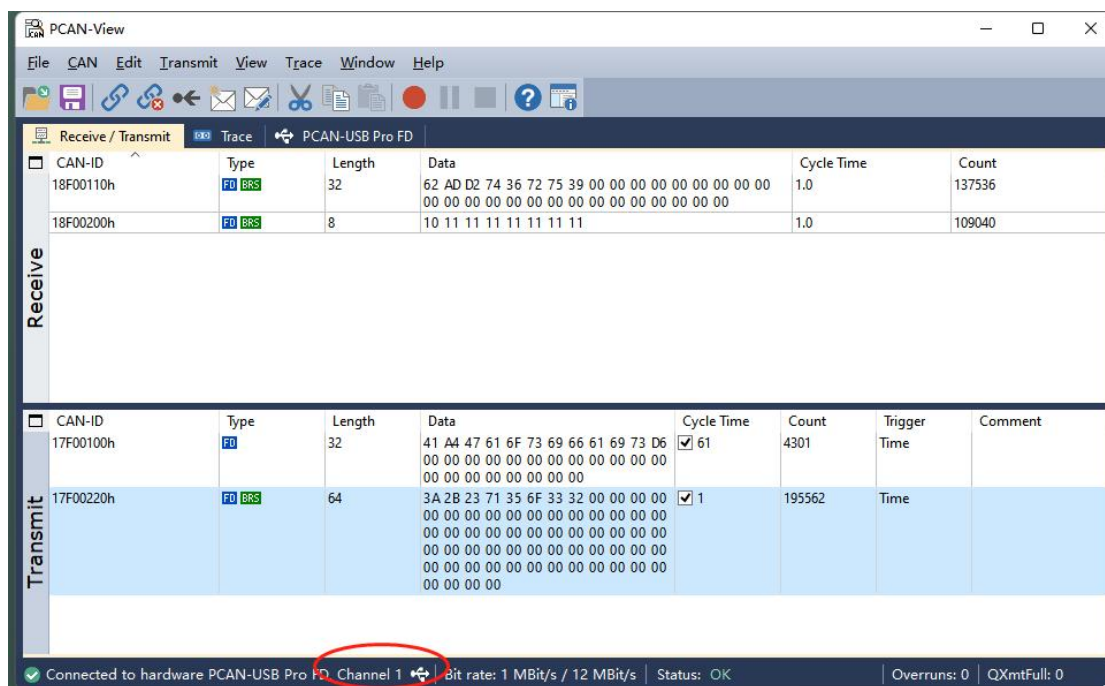


Device ID Value:	Function
11000000	Activated both CAN0 CAN1 120Ω resistor
01000000	Activated Only CAN1 120Ω resistor
10000000	Activated Only CAN0 120Ω resistor
00000000	Deactivated both CAN0,CAN1 120Ω resistor

注意设置完成后需要拔插 USB 线重新连接设备。

看到下图所示 TERM 下两个指示灯常亮，表示内部终端电阻使能。



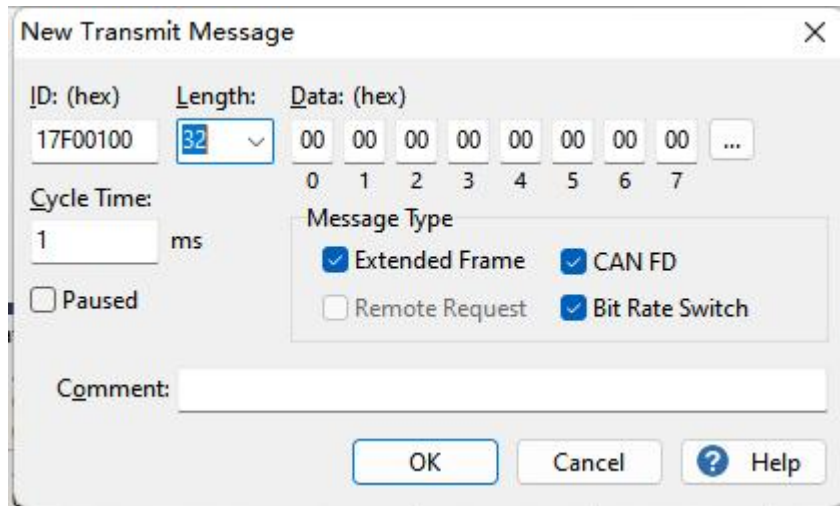


1. 选择菜单命令 **Transmit > New Message** 弹出“New Transmit Message”对话框。



## USB2CANFD-X2

高速 USB2.0 转换双路 CANFD 通讯设备



The "New Transmit Message" dialog box is used to configure a CAN message for transmission. It includes fields for ID (hex), Length, and Data (hex). The ID is set to 17F00100, Length is 32, and Data is 00 00 00 00 00 00 00 00. The Cycle Time is set to 1 ms. The Message Type section has checkboxes for Extended Frame (checked), CAN FD (checked), Remote Request (unchecked), and Bit Rate Switch (checked). There is a Paused checkbox and a Comment field. The bottom has OK, Cancel, and Help buttons.

ID: (hex)	Length:	Data: (hex)
17F00100	32	00 00 00 00 00 00 00 00 ...

Cycle Time: 1 ms

☐ Paused

Comment:

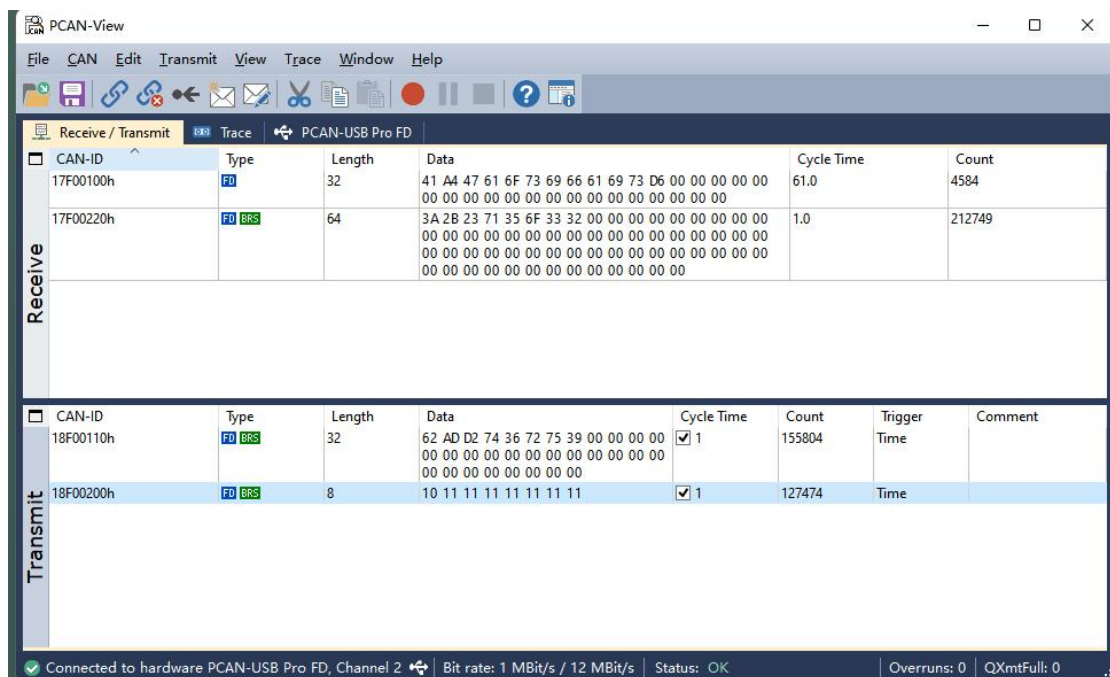
Message Type

☒ Extended Frame ☒ CAN FD

☐ Remote Request ☒ Bit Rate Switch

OK Cancel ? Help

2. 使能（勾选）复选框,CANFD, Extended Frame, Bit Rate Switch(如果不选此项,bus load 会非常高, 而且不选只会使能仲裁段波特率), 输入 ID、数据长度和 CAN 消息数据。
- 3 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。本次输入 1
4. 点击 OK 确认条目。创建的传输消息出现在接收/发送选项卡。
- 5 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。



The PCAN-View interface displays two message lists: Receive and Transmit. The Receive list shows two messages: 17F00100h (Type: FD, Length: 32, Cycle Time: 61.0, Count: 4584) and 17F00220h (Type: FD BRS, Length: 64, Cycle Time: 1.0, Count: 212749). The Transmit list shows two messages: 18F00110h (Type: FD BRS, Length: 32, Cycle Time: 1, Count: 155804, Trigger Time: Time) and 18F00200h (Type: FD BRS, Length: 8, Cycle Time: 1, Count: 127474, Trigger Time: Time). The status bar at the bottom indicates: Connected to hardware PCAN-USB Pro FD, Channel 2, Bit rate: 1 Mbit/s / 12 Mbit/s, Status: OK, Overruns: 0, QXmtFull: 0.

Receive / Transmit	CAN-ID	Type	Length	Data	Cycle Time	Count	Trigger Time	Comment
Receive	17F00100h	FD	32	41 A4 47 61 6F 73 69 66 61 69 73 D6 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	61.0	4584		
	17F00220h	FD BRS	64	3A 2B 23 71 35 6F 33 32 00	1.0	212749		
Transmit	18F00110h	FD BRS	32	62 AD D2 74 36 72 75 39 00	1	155804	Time	
	18F00200h	FD BRS	8	10 11 11 11 11 11 11 11	1	127474	Time	

Connected to hardware PCAN-USB Pro FD, Channel 2 | Bit rate: 1 Mbit/s / 12 Mbit/s | Status: OK | Overruns: 0 | QXmtFull: 0



## 高速 USB2.0 转换双路 CANFD 通讯设备

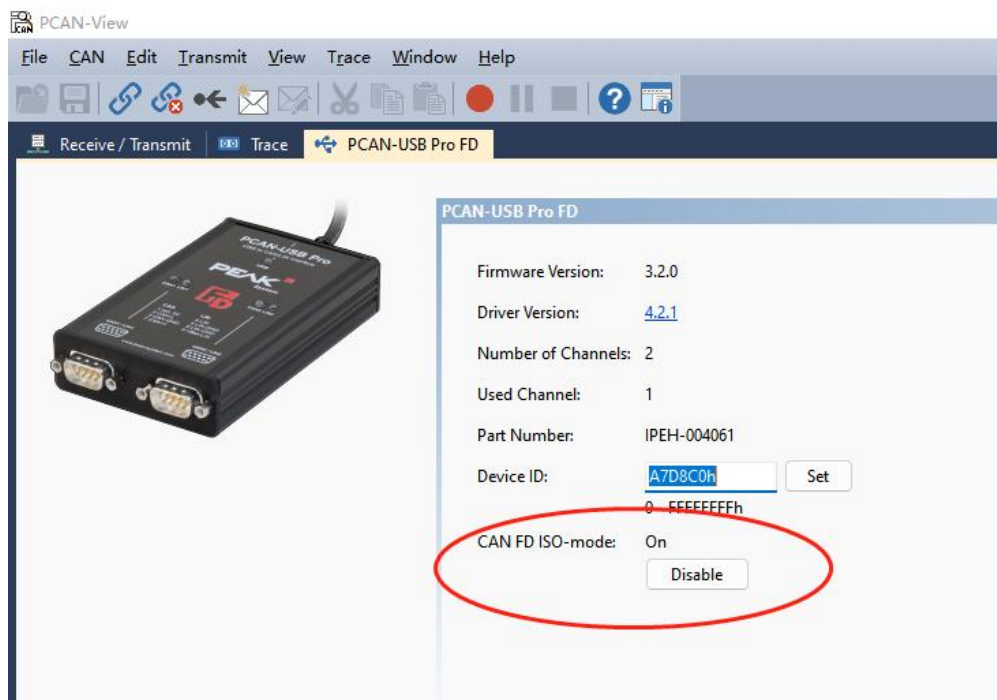
### 3.5 Trace Tab (数据记录选项卡)

[illegible]

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）。

### 3.6 PCAN-USB Pro FD (设备 ID 设置选项卡)



## USB2CANFD-X2

### 高速 USB2.0 转换双路 CANFD 通讯设备

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。

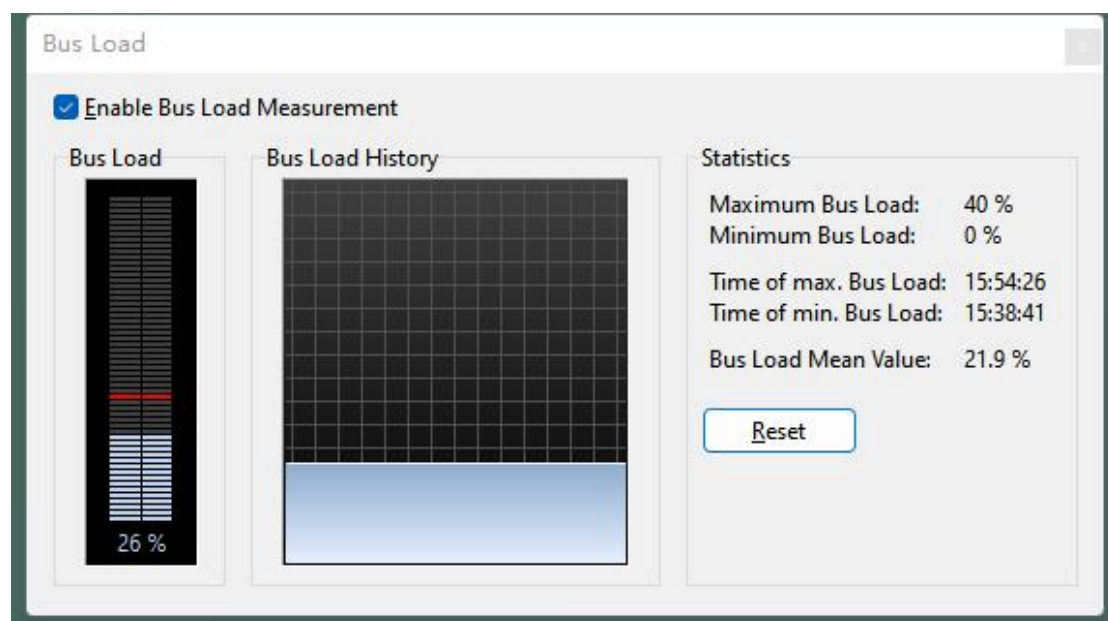
要识别 PCAN-USB Pro FD 适配器，请首先转到用于选择 PCAN-View 硬件的对话框。在"可用的 PCAN 硬件和 PCAN 网络"列表中，您可以右键单击每个 USB 适配器并执行命令"识别"。因此，相应适配器的 LED 会很快闪烁。

USB2CANFD-X2 设备还可以通过 ID 使能内置终端电阻，详细可以参考前面的章节 2.2 章节介绍原理和说明。

### 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

## 3.7 Bus Load Tab（总线负载选项卡）



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

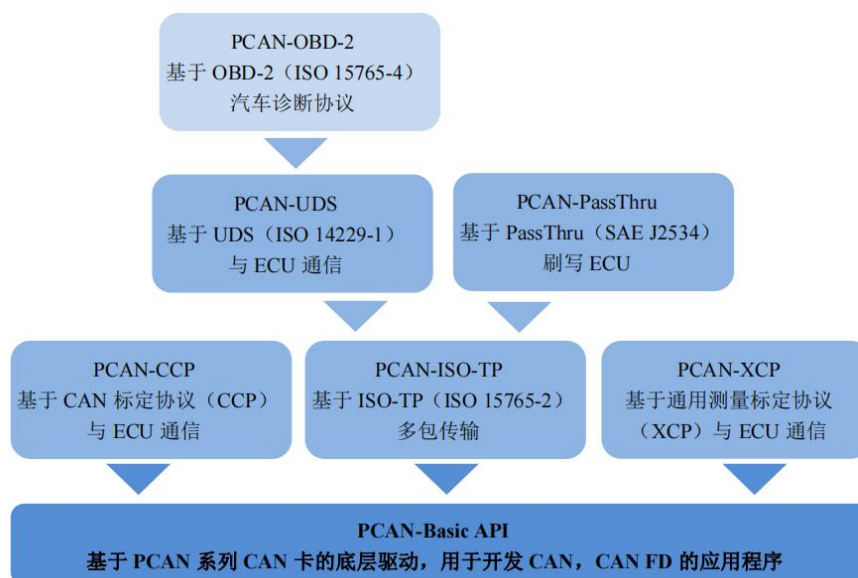
## 3.8 Status Bar（状态栏）

Connected to hardware PCAN-USB Pro FD, Channel 1 | Bit rate: 1 MBit/s / 12 MBit/s | Status: OK | Overruns: 0 | QXmtFull: 0

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。

### 3.9 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 ISO-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：

#### 3.9.1 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

### 3.9.2 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

### 3.9.3 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

### 3.9.4 PCAN-UDS API

UDS (ISO 14229-1) 标准用于统一的诊断服务和定义控制器 (E C U) 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

### 3.9.5 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

### 3.9.6 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不用考虑它的厂家。

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

## 4 Linux(PCAN) PCANVIEW 使用

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>



# USB2CANFD-X2

高速 USB2.0 转换双路 CANFD 通讯设备

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

## 4.1 安装 Linux 字符驱动

### Step1, 开发环境必须安装的安装包

```
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。

## 4.2 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 NC — (S) (S) (S)

④ = ⊥ ⊙ < ⊙ ⊠ 。

## 4.2.1 系统环境要求

按 4.1 章节安装好字符驱动，详细请参考 [Docker 部署指南](#)

[Linux 系统环境要求](#)

## 4.2.2 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://www.peak-system.com/debian/dists/`lsb_release -cs`/peak-system.list -O- | sudo 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：

## USB2CANFD-X2

高速 USB2.0 转换双路 CANFD 通讯设备

```
innomaker@innomaker:~/Downloads/peak-linux-driver-8.13.0$ wget -q http://www.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

```
Reading package lists... Done  
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:  
  pcanview-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) ...
```

## 4.3 收发数据

Step1 连接

参照下图将设备连接到 Linux 电脑，安装跳线帽激活 120 欧姆终端电阻。

Connect hardware to your pc As below, please add on the jumper for 120Ω jumper.

# USB2CANFD-X2

高速 USB2.0 转换双路 CANFD 通讯设备



## Step2 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，

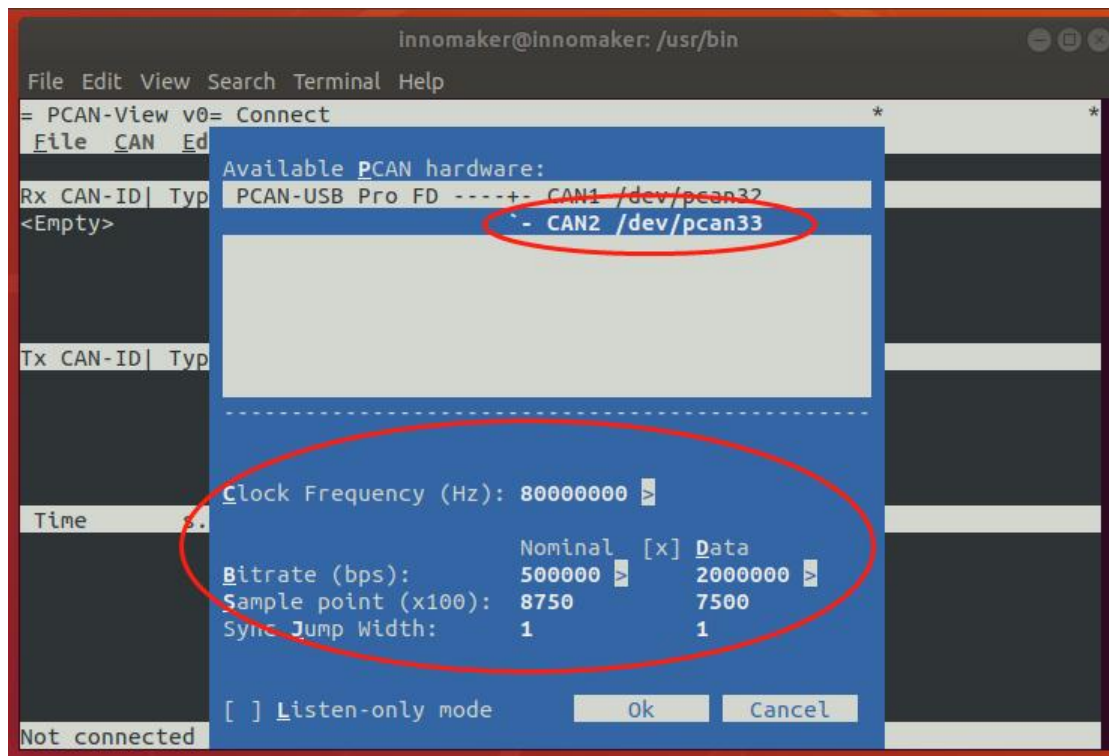
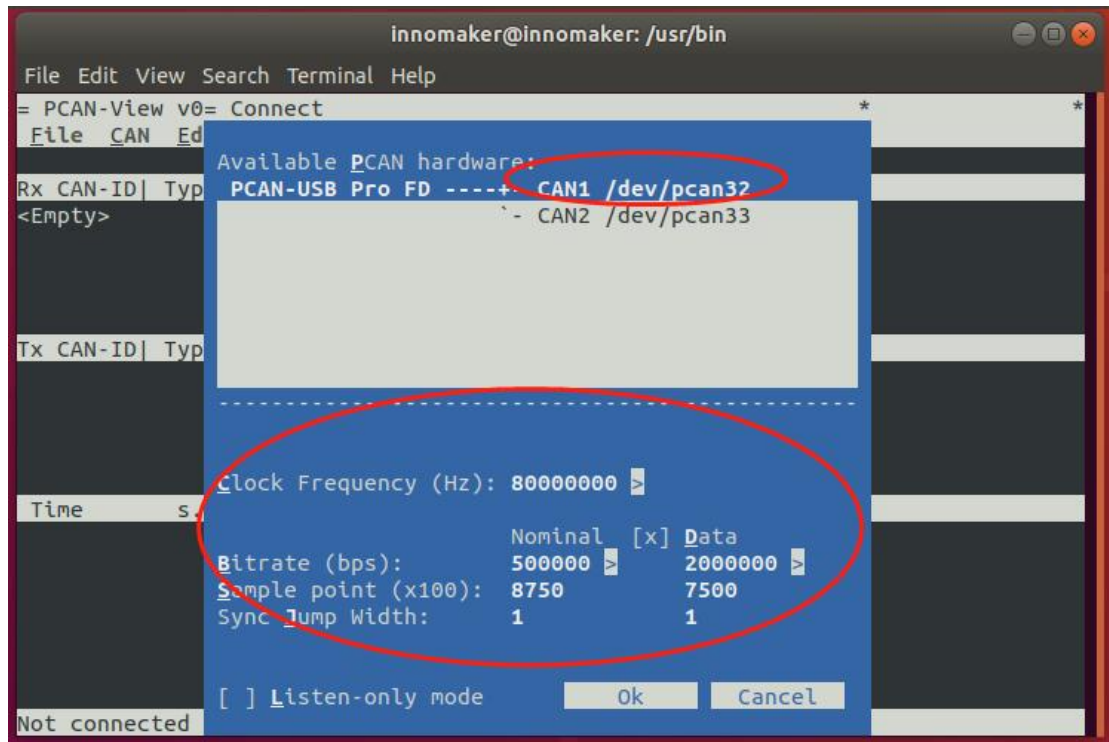
```
cd /usr/bin
```

```
/pcanview
```

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

# USB2CANFD-X2

高速 USB2.0 转换双路 CANFD 通讯设备



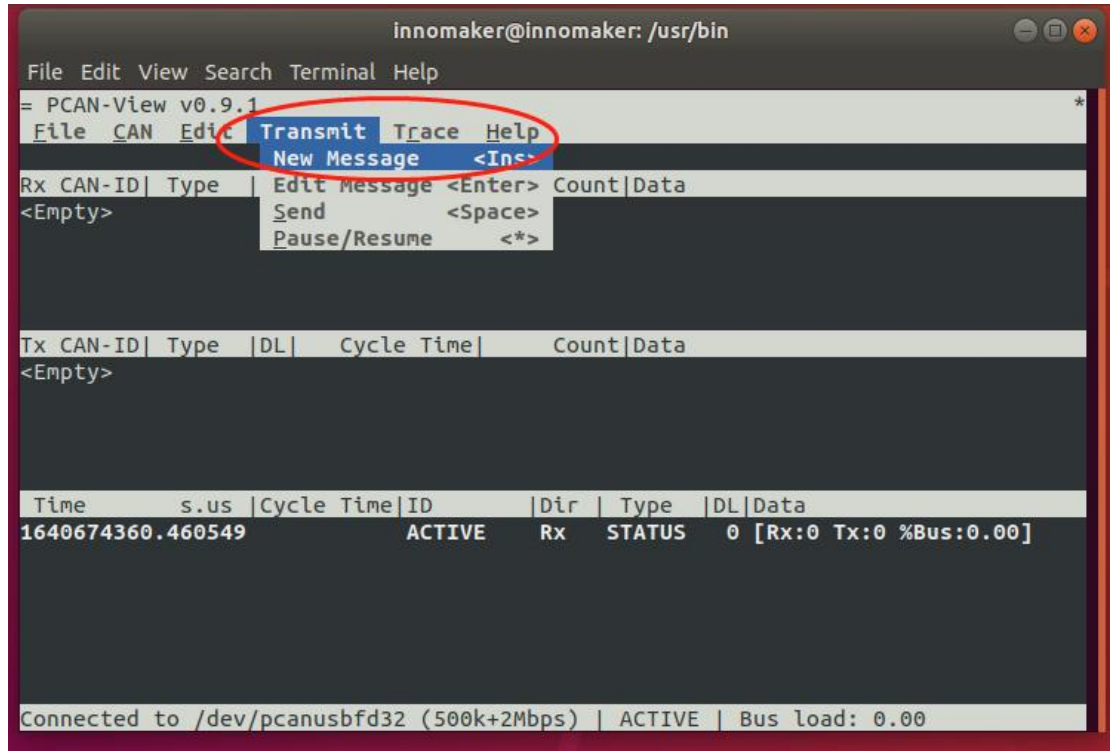
## Step3 发送和接收数据

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

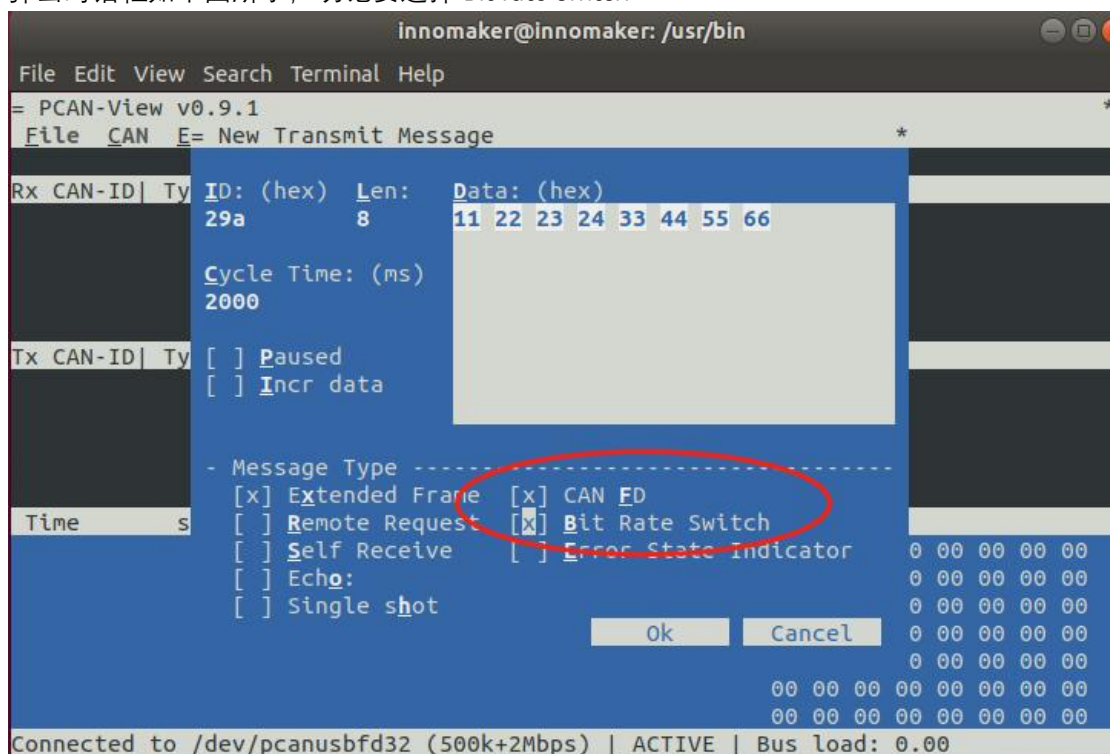


# USB2CANFD-X2

高速 USB2.0 转换双路 CANFD 通讯设备



弹出对话框如下图所示，切记要选择 Bit rate switch



## 高速 USB2.0 转换双路 CANFD 通讯设备

通讯效果如下图所示:

```

innomaker@innomaker: /usr/bin
File Edit View Search Terminal Help
= PCAN-View v0.9.1 *
File CAN Edit Transmit Trace Help

Rx CAN-ID| Type |DL| Cycle Time| Count|Data
0000029ah ..x.fb. 8 2000.898 142 11 22 23 24 33 44 55 66

Tx CAN-ID| Type |DL| Cycle Time| Count|Data
29bh ....fb. 8 * 50 ms 4416 aa bb cc dd ee ff 11 22

Time s.us |Cycle Time|ID |Dir | Type |DL|Data
1640677041~883509 50.000 29bh Tx ....fb. 8 aa bb cc dd ee ff 11 22
1640677041~933509 50.000 29bh Tx ....fb. 8 aa bb cc dd ee ff 11 22
1640677041~983529 50.020 29bh Tx ....fb. 8 aa bb cc dd ee ff 11 22
1640677042~033529 50.000 29bh Tx ....fb. 8 aa bb cc dd ee ff 11 22
1640677042~083530 50.001 29bh Tx ....fb. 8 aa bb cc dd ee ff 11 22
1640677042~133531 50.001 29bh Tx ....fb. 8 aa bb cc dd ee ff 11 22
1640677042~183531 50.000 29bh Tx ....fb. 8 aa bb cc dd ee ff 11 22
Connected to /dev/pcanusbfd33 (500k+2Mbps) | ACTIVE | Bus load: 0.00

```

如果需要发送或者接收 64 位长度的数据，需要将窗口最大化才能看到 Count|Data 结果

[illegible]

## 5 Linux(Socket CAN)










































### CAN-UTILS/C/PYTHON Linux 使用说明

#### 5.1 Linux 版本支持列表

USB2CANFD-X2 设备在以下版本 Linux 中可以直接识别为 SOCKETCAN，可以通过以下命令确认是否可直接识别为 CAN0 或者 CAN1。

`ifconfig -a` 如果没有此命令，需要执行先执行命令安装 `sudo apt-get install net-tools` 或者直接使用以下命令确认。

`dmesg`

	amd64	i386	arm64	armhf	ppc64el
Ubuntu:					
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	see Xenial				
Tumbleweed					
Debian:					

# USB2CANFD-X2

高速 USB2.0 转换双路 CANFD 通讯设备

Wheezy 7.11	■	■	■	■	
Jessie 8.11	■	■	■	■	
Stretch 9.9	■	■	■	■	
Buster 10	■	■	■	■	■
Bullseye 11	■	■	■	■	■

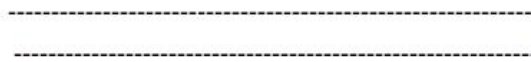
## 5.2 硬件连接

### CAN 0 Channel

CAN\_L(pin 2)

CAN\_H(pin 7)

### Connection



### CAN 1 Channel

CAN\_L(pin 2)

CAN\_H(pin 7)





# USB2CANFD-X2

高速 USB2.0 转换双路 CANFD 通讯设备

此时 LED 指示灯状态如下



## 5.3 CAN-UTILS 使用

### 5.3.1 环境准备安装 CAN-UTILS

键入命令 `ifconfig -a` 以检查 "can0" 和 'can1'设备在系统中是否可用，如果找不到命令 `ifconfig`，请使用命令 `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)
```

输入以下命令安装 can-utils

`sudo apt-get install can-utils`

Note:

此工具是测试 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/>

### 5.3.2 接收和发送数据

初始化 CAN0 和 CAN1 接口。

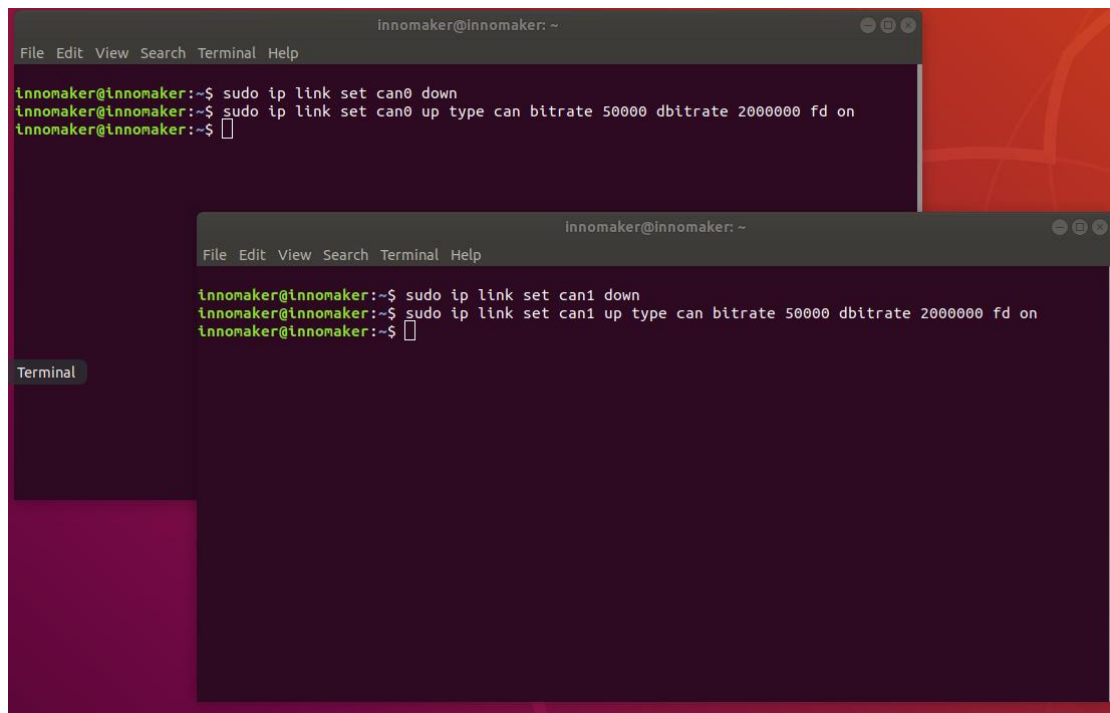
打开两个 Terminal 命令窗口，分别给 CAN0 和 CAN1

```
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
```



<1>然后设置 CAN0 为接收

```
candump can0
```

<2>设置 CAN1 为发送

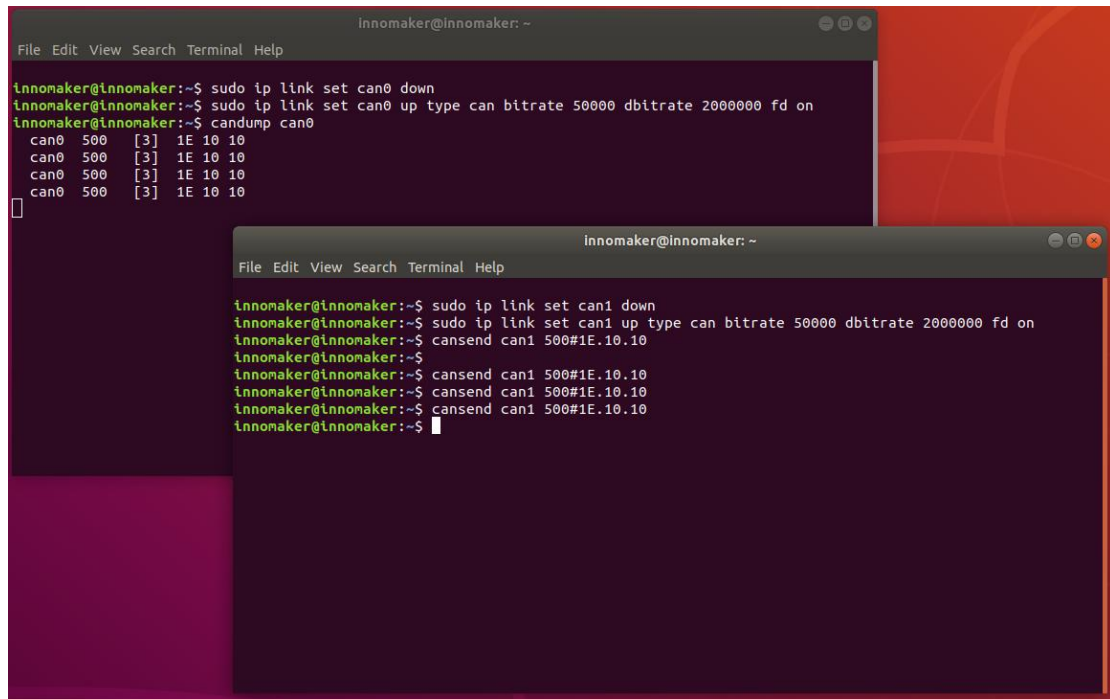
```
cansend can1 500#1E.10.10
```

收发效果如下图



# USB2CANFD-X2

高速 USB2.0 转换双路 CANFD 通讯设备



```
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:~$  
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:~$
```

## 5.4 C 程序使用

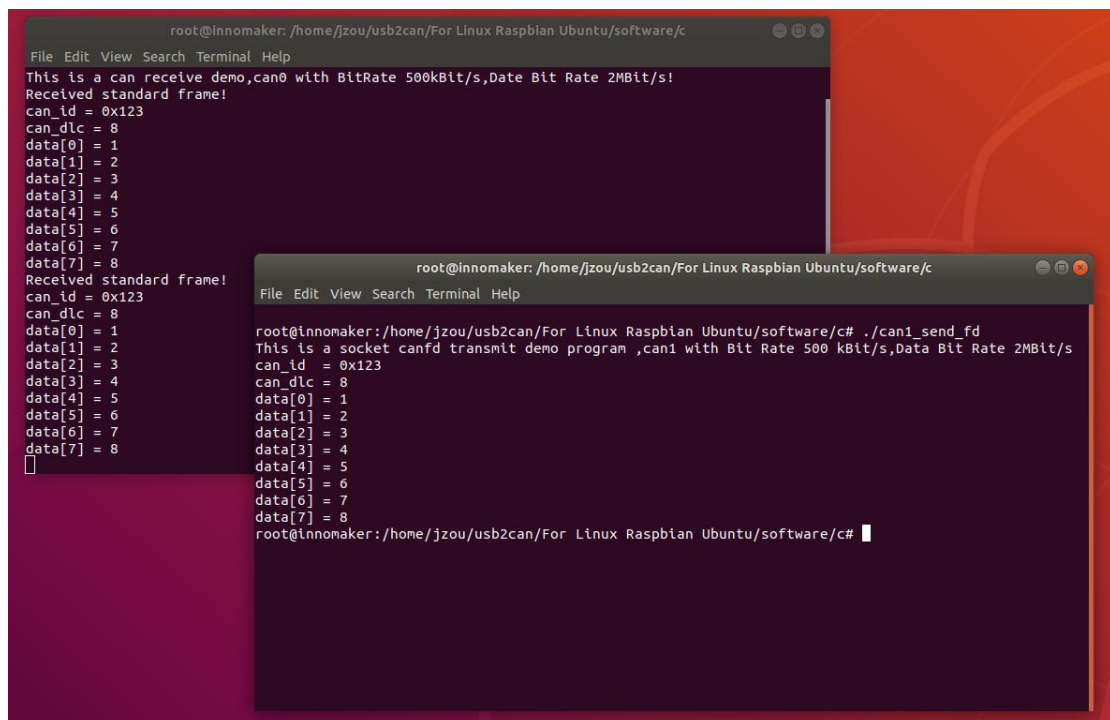
<1>执行以下指令，设置 CAN0 为接收端

```
sudo ./can0_receive_fd
```

<2>执行以下指令，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/jzhou/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/jzhou/usb2can/For Linux Raspbian Ubuntu/software/c#  
root@innomaker: /home/jzhou/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 Bit 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/jzhou/usb2can/For Linux Raspbian Ubuntu/software/c#
```

## 5.5 Python3 程序使用

(1) Check the Python version of your Raspbian. Python 3.7.3 default in 2019-09-26-Raspbian.img. Our Demo can run on any Python3 version.

`python3 -V`

A screenshot of a terminal window on a Raspberry Pi. The window title is 'pi@raspberrypi: ~'. The terminal shows the command 'python3 -V' being executed, and the output is 'Python 3.7.3'. The terminal has a dark background with light green text. The window has a standard Linux desktop environment with icons for a Raspberry Pi, a globe, a folder, and a terminal. The system clock in the top right corner shows 08:16.

(2) If you can't find the Python3 in system. Install the Python3

`sudo apt-get install python3-pip`

(3) Install Python CAN library.

`sudo pip3 install python-can`

(4) Set CAN0 as receiver

`sudo python3 receive.py`

(5) Set CAN1 as sender

`sudo python3 send.py`

## 5.6 C 程序/Python 程序 源码说明（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)

## 5.6.1 C 程序说明（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.

## USB2CANFD-X2

高速 USB2.0 转换双路 CANFD 通讯设备

```
/*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));
```

## 5.6.2 Python 程序说明（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')
```

## 5.6.3 错误帧说明（英文）

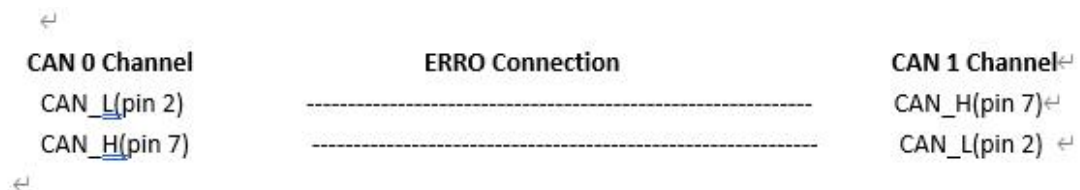
You may receive some error frame marked in red when you use the USB2CANX2-FD module. They will tell you what problem does the USB2CANX2-FD module meet on your CAN Bus.

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.





## USB2CANFD-X2

高速 USB2.0 转换双路 CANFD 通讯设备

SeqID	SystemTime	Channel	Direc...	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 */
```

This Error frame ID = 0x200000000 | 0x00000020|0x00000004

= 0x200000000 | CAN\_ERR\_ACK|CAN\_ERR\_CRTL

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:

## USB2CANFD-X2

高速 USB2.0 转换双路 CANFD 通讯设备

```
/* error status of CAN-controller / data[1] */
#define CAN_ERR_CRTL_UNSPEC      0x00 /* unspecified */
#define CAN_ERR_CRTL_RX_OVERFLOW 0x01 /* RX buffer overflow */
#define CAN_ERR_CRTL_TX_OVERFLOW 0x02 /* TX buffer overflow */
#define CAN_ERR_CRTL_RX_WARNING  0x04 /* reached warning level for RX errors */
#define CAN_ERR_CRTL_TX_WARNING  0x08 /* reached warning level for TX errors */
#define CAN_ERR_CRTL_RX_PASSIVE  0x10 /* reached error passive status RX */
#define CAN_ERR_CRTL_TX_PASSIVE  0x20 /* reached error passive status TX */
                                   /* (at least one error counter exceeds */
                                   /* the protocol-defined level of 127) */
#define CAN_ERR_CRTL_ACTIVE      0x40 /* recovered to error active state */
```

$\text{data}[1] = 0x0C = 0x04 | 0x08 = \text{CAN\_ERR\_CRTL\_RX\_WARNING} | \text{CAN\_ERR\_CRTL\_TX\_WARNING}$   
It means the USB2CAN module can't send/receive data properly and reached warning level.

$\text{data}[1] = 0x30 = 0x10 | 0x20 = \text{CAN\_ERR\_CRTL\_RX\_PASSIVE} | \text{CAN\_ERR\_CRTL\_TX\_PASSIVE}$   
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.