

USBCAN 操作流程 V1.70

- 1、将对应系统的 controlcan 文件夹，拷贝到 linux 桌面。
- 2、libcontrolcan.so、controlcan.h 为二次开发库文件，直接引用即可。内部集成了 USB 驱动，不用单独安装 USB 驱动。
- 3、main.cpp 为二次开发样例，一个简单的测试样例，可以完成数据的收发。
- 4、Makefile 文件，编译文件，编译前需要修改相应的路径：`g++ -o hello_cpp main.cpp /home/ttc/Desktop/controlcan/libcontrolcan.so -lpthread`（按实际路径修改）。
- 5、hello_cpp 为编译后生成的可执行文件。
- 6、Ctrl+Alt+T：打开命令窗口。
- 7、按照 ubuntu 命令.TXT 文档中的命令运行相应的样例。（注意，一定要加权限运行）
- 8、样例功能说明：
 - a. 样例设置波特率：125K
 - b. 接收线程中，CAN1、CAN2 不断循环查询并接收显示数据。
 - c. 主线中 CAN1、CAN2 交替循环发送 5 帧扩展帧数据，ID 递增。等待 10 秒后，关闭接收线程，程序全部退出。
 - d. 可以把 CAN1 与 CAN2 连接起来，实现数据的相互收发测试。
- 9、硬件连接：

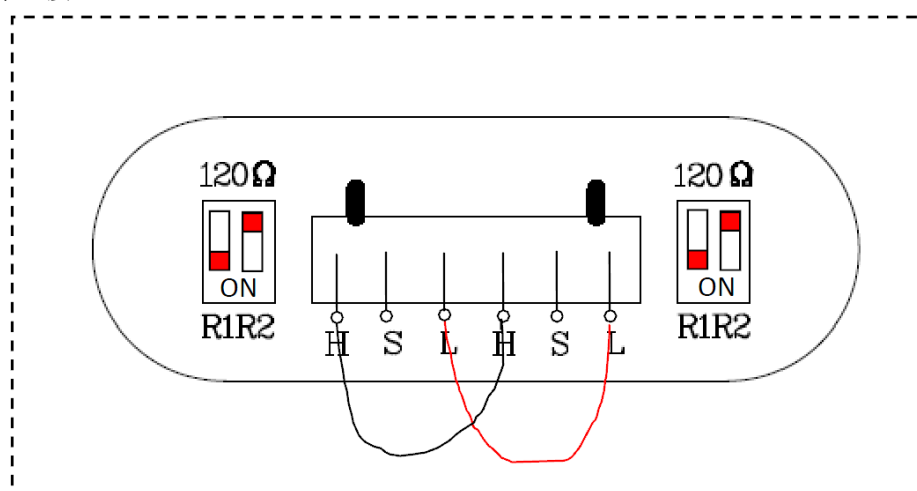


图 2 CANalsyt-II 分析仪（Linux 版、顶配 pro、OBD 通用版）测试接线图

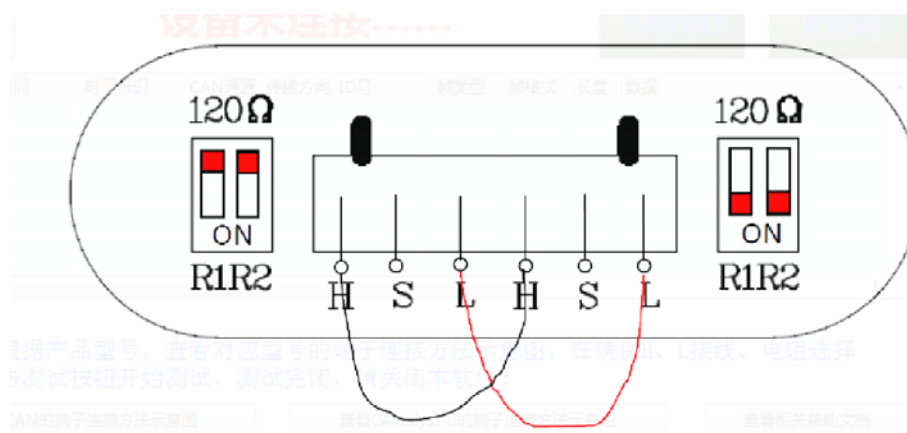


图 3 CANalsyt-II 分析仪（顶配版 带容错 CAN）测试接线图

- 10、测试结果

```
ttc@ubuntu: ~/Desktop/controlcan
ttc@ubuntu:~$ lsusb 确认USB CAN设备是否正常连接
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 003 Device 053: ID 04d8:0053 Microchip Technology, Inc. 目前设备在线
Bus 003 Device 004: ID 0e0f:0003 VMware, Inc. Virtual Mouse
Bus 003 Device 002: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 002: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
ttc@ubuntu:~$ cd Desktop/controlcan/ 定位目录
ttc@ubuntu:~/Desktop/controlcan$ rm hello_cpp 删除原文件
ttc@ubuntu:~/Desktop/controlcan$ make clean && make 重新编译
rm -f *.o hello
#gcc -o hello -L. -lcontrolcan -lpthread -usb main.c
#arm-none-linux-gnueabi-gcc -o hello -L. -L /home/caidunqing/controlcan -lcontrolcan -lpthread main.c
g++ -o hello_cpp main.cpp /home/ttc/Desktop/controlcan/libcontrolcan.so -lpthread
ttc@ubuntu:~/Desktop/controlcan$ sudo ./hello_cpp 一定要加sudo 获取权限运行, 否则操作不了USB设备
[sudo] password for ttc: 输入帐户密码
>>this is hello ! 程序运行正常
>>open deivce success! 打开设备正常
>>Get VCI_ReadBoardInfo success! 读取设备信息正常
>>Serial_Num:01701019000 产品序列号
>>hw_Type:CAN-Linux 设备类型
Index:0000 CAN1 TX ID:0x00000000 Extend Data DLC:0x08 data:0x 00 01 02 03 04 05 06 07 帧长度 数据:
Index:0001 CAN2 TX ID:0x00000001 Extend Data DLC:0x08 data:0x 00 01 02 03 04 05 06 07
Index:0002 CAN1 TX ID:0x00000002 Extend Data DLC:0x08 data:0x 00 01 02 03 04 05 06 07
Index:0003 CAN2 TX ID:0x00000003 Extend Data DLC:0x08 data:0x 00 01 02 03 04 05 06 07
Index:0004 CAN1 TX ID:0x00000004 Extend Data DLC:0x08 data:0x 00 01 02 03 04 05 06 07
Index:0005 CAN2 TX ID:0x00000005 Extend Data DLC:0x08 data:0x 00 01 02 03 04 05 06 07
Index:0006 CAN1 TX ID:0x00000006 Extend Data DLC:0x08 data:0x 00 01 02 03 04 05 06 07
Index:0007 CAN2 RX ID:0x00000000 Extend Data DLC:0x08 data:0x 00 01 02 03 04 05 06 07 TimeStamp:0x065FA867
Index:0008 CAN1 RX ID:0x00000001 Extend Data DLC:0x08 data:0x 00 01 02 03 04 05 06 07 TimeStamp:0x065FA877
Index:0009 CAN2 TX ID:0x00000007 Extend Data DLC:0x08 data:0x 00 01 02 03 04 05 06 07
Index:0010 CAN1 TX ID:0x00000008 Extend Data DLC:0x08 data:0x 00 01 02 03 04 05 06 07
Index:0011 CAN2 TX ID:0x00000009 Extend Data DLC:0x08 data:0x 00 01 02 03 04 05 06 07
Index:0012 CAN2 RX ID:0x00000002 Extend Data DLC:0x08 data:0x 00 01 02 03 04 05 06 07 TimeStamp:0x065FA882
Index:0013 CAN2 RX ID:0x00000004 Extend Data DLC:0x08 data:0x 00 01 02 03 04 05 06 07 TimeStamp:0x065FA89A
Index:0014 CAN1 RX ID:0x00000003 Extend Data DLC:0x08 data:0x 00 01 02 03 04 05 06 07 TimeStamp:0x065FA88E
Index:0015 CAN1 RX ID:0x00000005 Extend Data DLC:0x08 data:0x 00 01 02 03 04 05 06 07 TimeStamp:0x065FA8A5
Index:0016 CAN1 RX ID:0x00000007 Extend Data DLC:0x08 data:0x 00 01 02 03 04 05 06 07 TimeStamp:0x065FA8BD
Index:0017 CAN2 RX ID:0x00000006 Extend Data DLC:0x08 data:0x 00 01 02 03 04 05 06 07 TimeStamp:0x065FA8B1
Index:0018 CAN2 RX ID:0x00000008 Extend Data DLC:0x08 data:0x 00 01 02 03 04 05 06 07 TimeStamp:0x065FA8C8
Index:0019 CAN1 RX ID:0x00000009 Extend Data DLC:0x08 data:0x 00 01 02 03 04 05 06 07 TimeStamp:0x065FA8D4
run thread exit 10秒定时时间到, 程序退出 接收帧带时间标识
ttc@ubuntu:~/Desktop/controlcan$
```

显示列表说明:

- 1) 第一列 Index 为列表序号: 0 开始递增。
- 2) 第二列为通道号: CAN1/CAN2 两通道
- 3) 第三列为帧方向: RX (接收), TX (发送)
- 4) 第四列为 ID
- 5) 第五列为帧格式: Standard (标准帧)、Extend (扩展帧)
- 6) 第六列为帧类型: Data (数据帧)、Remote (远程帧)
- 7) 第七列为帧长度
- 8) 第八列为数据
- 9) 第九列为时间标识: 只有接收帧才有。

注意:

- 1) 运行时, 一定要加 Sudo 获取权限运行, 否则 USB 设备没有权限操作。可参考《USB 权限设置.pdf》, 将 USB 设备权限开放后, 不需要加 Sudo。
- 2) 因为发送与接收在两个线程中, 线程没有同步。所以在显示时, 有可能出现数据错位, 是正常现象。
- 3) 样例只是提供一个简单的调用 so 库的方法供参考, 程序接收, 与发送函数设置在两个线程中, 并且线程没有同步。现实中客户编程中, 发送与接收函数不能同时调用 (不支持多线程), 如果在多线程中, 一定需要互锁。需要客户自行完善代码。
- 4) 测试程序 10s 内会自动退出, 将不会再接收总线上的其它数据。
- 5) 测试接线, 可以参考: 光盘\说明文档目录\5.插件 1: USB-CAN 总线适配器测试.pdf
- 6) 二次开发, 可以参考: 光盘\说明文档目录\4.接口函数库 (二次开发库) 使用说明书.pdf

- 7) 用户需要二次开发，可以自行设计界面，后台代码中，需要修改波特率等参数。
- 8) 调用 `VCI_CloseDevice` 函数后，USB 会有复位操作，相关于插拔 USB 一次，需要 1-2 秒左右的响应时间，所以需要再次 `VCI_OpenDevice`，需要间隔 2 秒以上。
- 9) 不同的芯片平台 `so` 库不一样，需要选择使用对应平台的 `so` 库。如果提供的 `so` 库都使用不了，可以提供对应平台的交叉编译工具，联系技术进行编译对应平台的 `so` 库。