



UTF2202

用户手册

类别	内容
关键词	UTF2202、六叶树、用户手册
摘要	UTF2202 是六叶树一款免驱的 USB 接口的 SocketCAN 适配器，集成两路 CAN 接口，支持 CAN2.0 模式和 FD 模式，波特率范围:5K-6M。

文档记录

版本	日期	说明
V1.00	2024.11	创建文档



目录

UTF2202	1
用户手册	1
1.功能简介	3
1.1 产品概述	3
1.2 产品外观	3
1.3 参数指标	4
1.4 主要功能	4
1.5 软件支持	4
1.6 典型应用	4
2.接线说明	5
2.1 前面板接口	5
2.2 后面板接口	6
2.3 总线连接	7
3.使用教程	8
3.1 主机连接	8
3.2 软件(can-utils)安装(可选项, 非必须)	9
3.3 参数配置	10
3.4 开启设备	11
3.5 软件(can-utils)使用	11
3.6 关闭设备	13
4.代码接口调用	13
4.1 C++	14
5.免责声明	17
6.联系我们	17



1.功能简介

1.1 产品概述

UTF2202 是六叶树一款免驱的 USB 接口的 SocketCAN 适配器，集成两路 FD 接口，支持 CAN2.0A 和 CAN2.0B 协议和 FD，符合 ISO/DIS 11898-1/2/3 标准，支持 ISO 标准、Non-ISO 标准，采用 USB2.0 接口，与 USB1.1\USB3.0 总线兼容，支持 5Kbps~6Mbps 的波特率，自带匹配电阻，可以通过拨码开关进行控制，支持主流的 linux 操作系统，支持 X86、ARM 架构。

1.2 产品外观





1.3 参数指标

名称	六叶树UTF2202适配器	尺寸	10.0×7.5×2.9cm
调试软件	can-utils 工具(支持命令安装+源码安装)	操作系统	LINUX
应用范围	大陆雷达开发等	二次开发	SOCKET接口支持(提供C++用例)
功能	SOCKETCAN适配器	CAN波特率	5K~6M(支持自定义)
应用平台	X86/64/ARM等	驱动安装	免驱动安装,即插即用
USB接口	USB2.0 高速 480M/S	CAN标准	CAN2.0+FD
隔离类型	电源和信号双隔离	通道数量	2
应用平台	香橙派OrangePi/树莓派/友善之臂/东凌嵌入式/ 瑞芯微/英伟达等。	匹配电阻	内置120欧姆(通过拨码开关控制)
		外观材质	铝合金

1.4 主要功能

- 1.平台通过 USB 接口将 CAN 数据报文发送到 CAN 总线。
- 2.将 CAN 总线上的 CAN 数据报文通过 USB 接口采集到平台。

1.5 软件支持

- CAN 通讯工具 can-utils。

1.6 典型应用

- 机器人控制
- 汽车雷达调试

2.接线说明

2.1 前面板接口



2.1.1 USB1

预留接口，保留。

2.1.2 KEY

功能键，保留。

2.1.3 USB2

通过 USB 线接到电脑 USB 口，起供电和通讯的作用。

2.1.4 LED 指示灯

指示灯	状态	
SYS	常灭:正常	
	闪烁:硬件接收错误，丢包	

CAN0	指示 CAN0 通道状态 常亮:通道开启 常灭:通道关闭 闪烁:CAN 总线有数据正在进行	
CAN1	指示 CAN1 通道状态 常亮:通道开启 常灭:通道关闭 闪烁:CAN 总线有数据正在进行	
ERR	未启用, 保留	

2.2 后面板接口



2.2.1 接线端子

采用间距 5.08 端子。

R		CAN0				CAN1				R	
		CANL	REV	G	CANH	CANL	REV	G	CANH		
CAN0 匹配电阻开关, 拨到下方代表使能 CAN0 通道的匹配电阻 (120 欧姆)		CAN0 通道 L 线	未使用, 保留	GND	CAN0 通道 H 线	CAN1 通道 L 线	未使用, 保留	GND	CAN1 通道 H 线	CAN1 匹配电阻开关, 拨到下方代表使能 CAN1 通道的匹配电阻 (120 欧姆)	

2.2.2 匹配电阻开关

说明:

匹配电阻:已内置一个 120 欧姆的电阻,如下图:

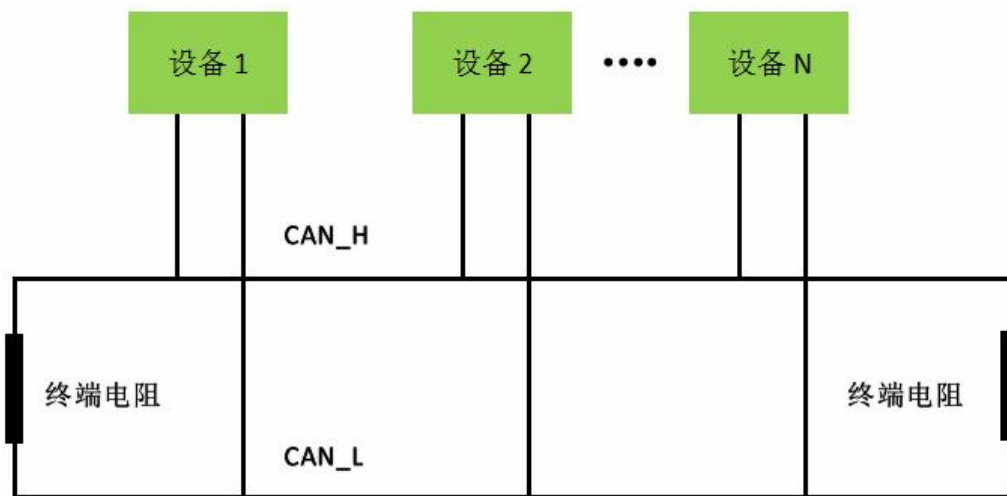


通过拨码开关控制，两个开关同时拨到下方代表开启匹配电阻。



2.3 总线连接

UTF2202 适配器和 CAN-bus 总线连接的时候，仅需要将 CAN_L 连 CAN_L, CAN_H 连 CAN_H 信号。CAN-bus 网络采用直线拓扑结构，总线的 2 个终端需要安装 120 Ω 的终端电阻；如果节点数目大于 2，中间节点不需要安装 120 Ω 的终端电阻。对于分支连接，其长度不应超过 3 米。CAN-bus 总线的连接见下图。



注意：CAN-bus 电缆可以使用普通双绞线、屏蔽双绞线。若通讯距离超过 1Km，应保证线的截面积大于 $\Phi 1.0\text{mm}^2$ ，具体规格应根据距离而定，常规是随距离的加长而适当加大。

3.使用教程

3.1 主机连接

通过配套的 USB 线，将 UTF2202 连接到主机(电脑或开发板)。

注意事项:所有命令都使用管理员或 root 权限执行。

通过“ifconfig -a”查看设备是否被主机识别，如果能看到 can0、can1 接口代表设备已被主机正确识别。



```
root@lys-virtual-machine:~# ifconfig -a
can0: flags=193<UP,RUNNING,NOARP> mtu 72
    RX packets 20 bytes 320 (320.0 B)
    RX errors 0 dropped 1 overruns 0 frame 0
    TX packets 3 bytes 16 (16.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

can1: flags=128<NOARP> mtu 16
    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

ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.102 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::5a80:7aec:5219:14a9 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:e6:57:37 txqueuelen 1000 (Ethernet)
    RX packets 213121 bytes 13753711 (13.7 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 418938 bytes 51663435 (51.6 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 480 bytes 41820 (41.8 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 480 bytes 41820 (41.8 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@lys-virtual-machine:~#
```

这里截图显示can0、can1，设备已被主机正常识别，如果没有显示can0、can1接口，说明当前系统内核可能版本太低或内核编译时没有启用socketcan功能。

3.2 软件(can-utils)安装(可选项，非必须)

can-utils 工具非必须安装，该工具只是用来进行简单的 CAN 数据收发测试，自己开发的 socketcan 应用程序不会依赖该工具，如果安装，也只需要安装一次即可，不用每次系统重启都安装。

安装命令: apt-get install can-utils

```
root@ubuntu:/home/zhumiao# apt-get install can-utils
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  can-utils
0 upgraded, 1 newly installed, 0 to remove and 255 not upgraded.
Need to get 0 B/117 kB of archives.
After this operation, 656 kB of additional disk space will be used.
Selecting previously unselected package can-utils.
(Reading database ... 198003 files and directories currently installed.)
Preparing to unpack .../can-utils_2018.02.0-lubuntu1_amd64.deb ...
Unpacking can-utils (2018.02.0-lubuntu1) ...
Setting up can-utils (2018.02.0-lubuntu1) ...
Processing triggers for man-db (2.9.1-1) ...
```

检查安装结果:

命令:

candump --help



```
root@ubuntu:/home/zhumiao# candump -help
candump - dump CAN bus traffic.
```

```
Usage: candump [options] <CAN interface>+
       (use CTRL-C to terminate candump)
```

Options:

可以输出这些帮助信息，说明安装成功！

```
-t <type>   (timestamp: (a)bsolute/(d)elta/(z)ero/(A)bsolute w date)
-H          (read hardware timestamps instead of system timestamps)
-c          (increment color mode level)
-i          (binary output - may exceed 80 chars/line)
-a          (enable additional ASCII output)
-S          (swap byte order in printed CAN data[] - marked with ``')
-s <level>  (silent mode - 0: off (default) 1: animation 2: silent)
-b <can>    (bridge mode - send received frames to <can>)
-B <can>    (bridge mode - like '-b' with disabled loopback)
-u <usecs>  (delay bridge forwarding by <usecs> microseconds)
-l          (log CAN-frames into file. Sets '-s 2' by default)
-L          (use log file format on stdout)
-n <count>  (terminate after reception of <count> CAN frames)
-r <size>   (set socket receive buffer to <size>)
-D          (Don't exit if a "detected" can device goes down.
-d          (monitor dropped CAN frames)
-e          (dump CAN error frames in human-readable format)
-x          (print extra message infos, rx/tx brs esi)
-T <msecs>  (terminate after <msecs> without any reception)
```

```
Up to 16 CAN interfaces with optional filter sets can be specified
on the commandline in the form: <ifname>[,filter]*
```

3.3 参数配置

注意：不管是使用 **can-utils** 工具还是自己开发的 **socketcan** 应用，使用前，参数配置都是必须的,而且是每次主机重启都必须配置，所以建议将参数配置命令放到主机的系统启动脚本里执行。

波特率参数配置:

```
ip link set can0 type can bitrate 1000000 dbitrate 5000000 fd on fd-non-iso on
```

指令含义:

can0 通道参数配置

仲裁域波特率:1M 对应命令里的 1000000，单位是 Hz

数据域波特率:5M 对应命令里的 5000000，单位是 Hz

FD:FD 模式开启 对应命令里的 fd on，如果不工作在 FD 模式，命令里删掉 fd on 这一项即可

ISO 模式:NON-ISO 对应命令里的 fd-non-iso on，如果要配置成 ISO 模式,on 改为 off 即可。

```
root@lys-virtual-machine:/home/lys# ip link set can0 type can bitrate 1000000 dbitrate 5000000 fd on fd-non-iso on
root@lys-virtual-machine:/home/lys#
```



3.4 开启设备

命令:"ip link set up can0"

指令含义:

使能 can0 通道

```
root@lys-virtual-machine:/home/lys# ip link set up can0
root@lys-virtual-machine:/home/lys#
```

3.5 软件(can-utils)使用

3.5.1 数据发送

//ID 3 位代表标准帧 8 位代表扩展帧 不足的补零

cansend can0 023#1122334455667788

//扩展数据帧

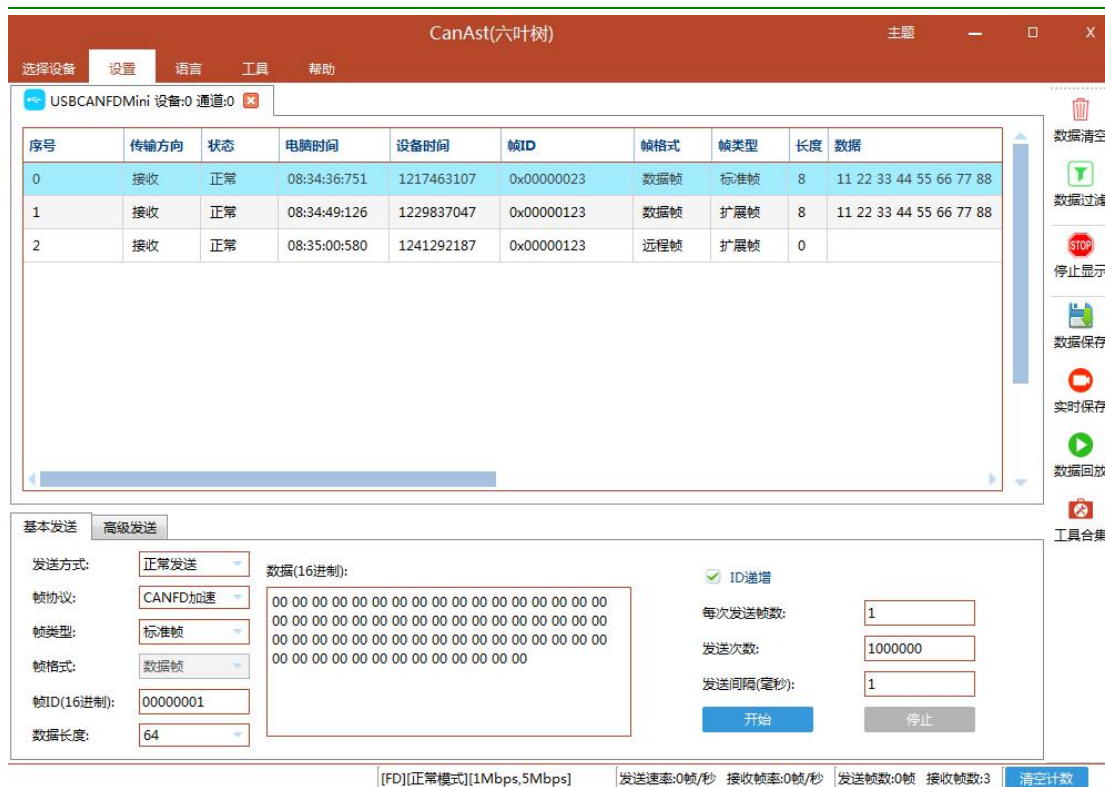
cansend can0 00000123#1122334455667788

//远程帧

cansend can0 00000123#R

命令截图如下:

```
root@lys-virtual-machine:~# cansend can0 023#1122334455667788
root@lys-virtual-machine:~# cansend can0 00000123#1122334455667788
root@lys-virtual-machine:~# cansend can0 00000123#R
root@lys-virtual-machine:~#
```

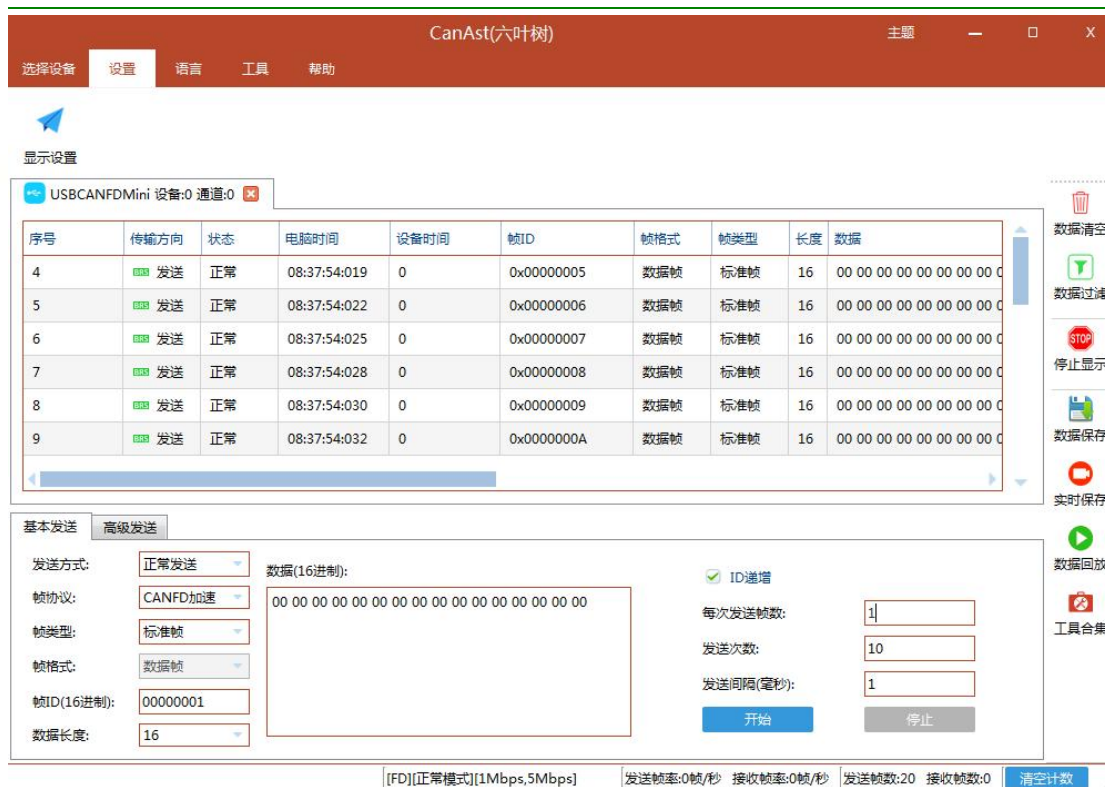


3.5.2 数据接收

//接收数据并打印在窗口

candump can0

```
root@lys-virtual-machine:~# candump can0
can0 001 [16] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
can0 002 [16] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
can0 003 [16] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
can0 004 [16] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
can0 005 [16] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
can0 006 [16] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
can0 007 [16] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
can0 008 [16] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
can0 009 [16] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
can0 00A [16] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
can0 001 [16] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
can0 002 [16] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
can0 003 [16] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
can0 004 [16] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
can0 005 [16] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
can0 006 [16] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
can0 007 [16] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
can0 008 [16] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
can0 009 [16] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
can0 00A [16] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```



3.6 关闭设备

命令:"ip link set down can0"

指令含义:

关闭 can0 通道

```
root@lys-virtual-machine:~# ip link set down can0
root@lys-virtual-machine:~#
```

4.代码接口调用

应用代码可以使用 **SocketCAN** 套接字对设备进行控制，包括数据的接收和发送以及滤波。这里要注意，代码里的 **socketCAN** 套接字是不能控制 CAN 卡的开启和关闭以及波特率的设置，这些都必须通过上面说的“参数配置”配置好。具体调用逻辑流程，可以查看六叶树官方的 **socketCAN** 专栏:http://www.liuyeshu.cn/?page_id=1256,里面有代码用例。

4.1 C++

```
/**
*****
* @file    socketCanTest.cpp
* @author  lys
* @version V1.1.0
* @date    2023
* @brief
*
* @note
* SOCKET CAN收发使用用例
* 准备条件:
* 1.设备驱动已安装好,ifconfig -a 能够查看can0设备
* 2.已通过命令ip link配置好波特率并开启
*    ip link set can0 type can bitrate 500000
*    ip link set up can0
* 用法:
* write:发送can数据,使用结构体can_frame
* read:接收can数据,使用结构体can_frame
* 代码功能:
* 程序运行,初始化了socket,设置了过滤器功能(setsockopt),只接收0x590的报文,发送了1帧0x610
* 报文,接收到0x590的报文接收线程会自动退出,整个程序退出。
* @endverbatim
*****
* 版权:长沙六叶树教育科技有限公司
* 官网:www.liuyeshu.cn
* 更多资料教程下载见官网
*****
修改日期    版本号    修改者    功能描述
2018.11.10  V1.1.0    LYS
-----
*****
*/
#include <iostream>
#include <string.h>
#include <unistd.h>
#include <net/if.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <linux/can.h>
#include <linux/can/raw.h>
#include <thread>
```

```
/**
 * @输入参数: 无
 * @输出参数: 无
 * @返回值: 无
 * @说明: 接收线程函数
 * 接收到1帧数据自动退出
 */
void receiveThread(int socket)
{
    struct can_frame frame;

    std::cout << "Info: read start...\n";

    int nbytes = read(socket, &frame, sizeof(struct can_frame));

    if (nbytes < 0)
    {
        std::cout << "Error: can raw socket read.\n";
        return;
    }

    if (nbytes < sizeof(struct can_frame))
    {
        std::cout << "Error: read incomplete CAN frame.\n";
        return;
    }

    printf("Receive OK\n");

    // 打印接收到的报文数据
    printf("ID=0x%X DLC=%d Data: ", frame.can_id, frame.can_dlc);
    for (uint8_t i = 0; i < frame.can_dlc; ++i)
    {
        printf("0x%02X ", frame.data[i]);
    }
    printf("\n");
}
```

```
int main()
{
    int nbytes;
    struct sockaddr_can socketCan0Addr;
    struct ifreq ifr;
    struct can_frame frame = {0};

    int socketFd = socket(PF_CAN, SOCK_RAW, CAN_RAW);

    // 获取can0的接口索引
    strcpy(ifr.ifr_name, "can0");
    ioctl(socketFd, SIOCGIFINDEX, &ifr);

    socketCan0Addr.can_family = AF_CAN;
    //把can0的接口索引赋值给socketCan0Addr
    socketCan0Addr.can_ifindex = ifr.ifr_ifindex;

    // 把socketCan0Addr绑定到socket上
    bind(socketFd, (struct sockaddr *)&socketCan0Addr, sizeof(socketCan0Addr));

    /*设置过滤, 只接收id是0x590的CAN报文,如果需要接收所有数据
    ,不执行CAN_RAW_FILTER操作即可*/
    struct can_filter rfilter;
    rfilter.can_id = 0x590;
    rfilter.can_mask = CAN_SFF_MASK;
    setsockopt(socketFd, SOL_CAN_RAW, CAN_RAW_FILTER, &rfilter, sizeof(rfilter));

    // 开启接收回复的线程
    std::thread t(receiveThread, socketFd);

    // 准备数据
    frame.can_id = 0x610;
    //扩展帧
    frame.can_id |= CAN_EFF_FLAG;
    frame.can_dlc = 8;
    frame.data[0] = 0x40;
    frame.data[1] = 0x01;
    frame.data[2] = 0x10;
    frame.data[3] = 0x00;
    frame.data[4] = 0x00;
    frame.data[5] = 0x00;
    frame.data[6] = 0x00;
    frame.data[7] = 0x00;

    // 发送数据
    nbytes = write(socketFd, &frame, sizeof(frame));
    if (nbytes != sizeof(frame))
    {
        std::cout << "Send Error\n";
    }

    //等待接收线程结束
    t.join();

    //程序退出
    return 0;
}
```

5. 免责声明

本文档提供有关长沙六叶树教育科技有限公司产品的信息。本文档并未授予任何知识产权的许可，并未以明示或暗示，或以禁止发言或其它方式授予任何知识产权许可。除六叶树教育科技有限公司在其产品的销售条款和条件中声明的责任之外，六叶树概不承担任何其它责任。并且，六叶树电子产品的销售使用不作任何明示或暗示的担保，包括对产品的特定用途适用性、适销性或对任何专利权、版权或其它知识产权的侵权责任等，均不作担保。六叶树可能随时对产品规格及产品描述做出修改，不另行通知。

6. 联系我们



电话:15211065817(业务合作咨询)



邮箱:798746621@qq.com(业务咨询+技术支持)



微信:18673379565(技术支持)



官网:www.liuyeshu.cn(资料下载)



网上商城:<https://shop112408209.taobao.com>(产品购买)

淘宝店铺搜索：“六叶树教育科技”