以太网设备1

以太网系列1

区别于其他通信接口类型的设备，以太网系列 CAN 卡在本接口库中只进行通道链接以及数据收发。设备相关参数（如波特率、滤波、IP 等）的设置不在本接口库功能范畴。本系列每个通道对应一个网络连接，可理解为同一个设备的多个通道做为不同的单通道设备。

TCP 系列型号包括：CANDTU-NET、CANDTU-NET-400、CANET-TCP、CANWIFI-TCP、CANFDWIFI-TCP

UDP 系列型号包括：CANET-UDP、CANWIFI-UDP、CANFDWIFI-UDP。

1.工作模式

连接的工作模式，如设备作为服务器，二次开发时应设为客户端模式

项	值	说明
path	n/work_mode	n 代表通道号
value	0-客户端 1-服务器	
Get/Set	Set	

注意：

1. 需在 ZCAN_StartCAN 之前设置
2. UDP 系列不设置此参数

2.本地端口

UDP模式下的本地监听端口

项	值	说明
path	n/local_port	n 代表通道号
value	自定义	如 "4001"
Get/Set	Set	

注意：

1. 需在 ZCAN_StartCAN 之前设置
2. TCP 系列不设置此参数

3.IP地址

目标设备的IP

项	值	说明
path	n/ip	n 代表通道号
value	自定义	如 "192.168.0.178"
Get/Set	Set	

注意：

1. 需在 ZCAN_StartCAN 之前设置

4.工作端口

目标设备监听的端口

项	值	说明
path	n/work_port	n 代表通道号
value	自定义	如 "4001"
Get/Set	Set	

注意：

1. 需在 ZCAN_StartCAN 之前设置

5.示例代码

```
// 以下代码以 CANET-2E-U 为例
// 对于本系列，每个通道为一个单独连接，可以将每个通道理解为独立的设备进行操作
#include "stdafx.h"
#include "zlgcan.h"
#include <iostream>
#include <windows.h>
#include <thread>
#define CH_COUNT 2
bool g_thd_run = 1;
// 此函数仅用于构造示例 CAN 报文
void get_can_frame(ZCAN_Transmit_Data& can_data, canid_t id)
{
memset(&can_data, 0, sizeof(can_data));
can_data.frame.can_id = id; // CAN ID
can_data.frame.can_dlc = 8; // CAN 数据长度 8
can_data.transmit_type = 0; // 正常发送
for (int i = 0; i < 8; ++i) { // 填充 CAN 报文 DATA
can_data.frame.data[i] = i;
} }
void thread_task(CHANNEL_HANDLE handle, int ch)
{
std::cout << "chnl: " << std::dec << ch << " thread run, handle:0x" << std::hex << handle
<< std::endl;
ZCAN_Receive_Data data[100] = {};
while (g_thd_run)
{
int count = ZCAN_GetReceiveNum(handle, 0); // 获取 CAN 报文（参数 2: 0 - CAN, 1 - CANFD）
数量
while (g_thd_run && count > 0)
{ int rcount = ZCAN_Receive(handle, data, 100, 10);
for (int i = 0; i < rcount; ++i)
{
std::cout << "CHNL: " << std::dec << ch << " recv can ID: 0x" << std::hex <<
data[i].frame.can_id << std::endl;
}
count -= rcount;
}
Sleep(100);
}
std::cout << "chnl: " << std::dec << ch << " thread exit" << std::endl;
}
int _tmain(int argc, _TCHAR* argv[])
{
UINT dev_type = ZCAN_CANETTCP;
std::thread thd_handle[CH_COUNT];
CHANNEL_HANDLE ch[CH_COUNT] = {};
DEVICE_HANDLE device[CH_COUNT] = {INVALID_DEVICE_HANDLE};
IProperty* prop[CH_COUNT] = {};
// 循环打开、设置、初始化、启动每个通道
for (int i = 0; i < CH_COUNT; ++i) {
char path[64] = {};
char port[2][16] = { "4001", "4002" };
// 打开设备，即使为同一个设备，不同通道的设备索引也不是同一个
device[i] = ZCAN_OpenDevice(dev_type, i, 0);
if (INVALID_DEVICE_HANDLE == device[i]) {
std::cout << "open device failed!" << std::endl;
goto end;
}
// 获取 IProperty 指针，用于配置参数
prop[i] = GetIProperty(device[i]);
if (NULL == prop[i]) {
std::cout << "get property failed" << std::endl;
goto end;
}
// 设置工作模式为客户端
if (0 == prop[i]->SetValue("0/work_mode", "0")) {
std::cout << "set work mode failed" << std::endl;
goto end;
}
// 设置目标 IP 地址
if (0 == prop[i]->SetValue("0/ip", "172.16.9.221")) {
std::cout << "set ip failed" << std::endl;
goto end;
}
// 设置目标端口 通道 0-4001，通道 1-4002
if (0 == prop[i]->SetValue("0/work_port", port[i])) {
std::cout << "set port failed" << std::endl;
goto end;
}
// 初始化通道，在 CANET 系列中初始化不做实际操作，仅用于获取通道句柄
ZCAN_CHANNEL_INIT_CONFIG config;
ch[i] = ZCAN_InitCAN(device[i], 0, &config);
if (INVALID_CHANNEL_HANDLE == ch[i]) {
std::cout << "init channel failed!" << std::endl;
goto end;
}
// 启动 CAN 通道
if (0 == ZCAN_StartCAN(ch[i])) {
std::cout << "start channel failed" << std::endl;
goto end;
}
// 启动 CAN 通道的接收线程
thd_handle[i] = std::thread(thread_task, ch[i],i);
}
// 通道 0 发送 10 帧报文
ZCAN_Transmit_Data trans_data[10] = {};
for (int i = 0; i < 10; ++i){
get_can_frame(trans_data[i], i);
}
int send_count = ZCAN_Transmit(ch[0], trans_data, 10);
std::cout << "send frame: " << std::dec << send_count << std::endl;
system("pause");
end:
g_thd_run = 0;
for (int i = 0; i < CH_COUNT; ++i) {
if (thd_handle[i].joinable())
thd_handle[i].join();
std::cout << "thread exit, close device" << std::endl;
if (NULL != prop[i])
ReleaseIProperty(prop[i]);
if (NULL != device[i])
ZCAN_CloseDevice(device[i]);
}
system("pause");
return 0;
}
```