```
//---------------------------------------------------------------------------

#include <Vcl.Dialogs.hpp>
#pragma hdrstop

#include "UnitDMSockets.h"
#include "UnitFormMain.h"
#include "UnitUtils.h"
#include "UnitThreadWorking.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma classgroup "Vcl.Controls.TControl"
#pragma resource "*.dfm"
TdmSockets *dmSockets;
//---------------------------------------------------------------------------
__fastcall TdmSockets::TdmSockets(TComponent* Owner)
    : TDataModule(Owner)
{
    __read_ini_file();  // четене на параметрите от конфигурационния файл

    for(int i = 0; i < intVolume; i++)
    {
        TClientSocket * cs = new TClientSocket(Owner);
        cs->Port = 37;
        cs->Address = stringIPAddress;

        cs->OnConnect = csOut->OnConnect;
        cs->OnDisconnect = csOut->OnDisconnect;
        cs->OnError = csOut->OnError;
        cs->OnRead = csOut->OnRead;

        vcsOut.push_back(cs);
    }
}
//---------------------------------------------------------------------------
__fastcall TdmSockets::~TdmSockets(void)
{
    __write_ini_file();
}
//---------------------------------------------------------------------------
void __fastcall TdmSockets::__read_ini_file(void)
{
    stringIniFileName = Application->ExeName;
    stringIniFileName = stringIniFileName.SubString(1, stringIniFileName.Length() - 3) + "ini";

    iniFile = new TIniFile(stringIniFileName);
    if(!FileExists(stringIniFileName))
    {
        MessageDlg("Missing configuration file!", mtError, TMsgDlgButtons() << mbOK, 0);
    }

    formMain->Top  = iniFile->ReadInteger(L"FormPos", L"Top", 0);
    formMain->Left = iniFile->ReadInteger(L"FormPos", L"Left", 0);

    formMain->memoLog->Visible = !iniFile->ReadInteger(L"Log", L"Visible", 0);

    stringIPAddress = iniFile->ReadString(L"TimeServer", L"IPAddress", L"127.0.0.1");

    intVolume      = iniFile->ReadInteger(L"Package", L"Volume", 1);
    intDelayToFree = iniFile->ReadInteger(L"Package", L"DelayToFree", 0);
}
//---------------------------------------------------------------------------
void __fastcall TdmSockets::__write_ini_file(void)
{
    try
    {
        iniFile->WriteInteger("FormPos", "Top", formMain->Top);
        iniFile->WriteInteger("FormPos", "Left", formMain->Left);

        iniFile->WriteInteger(L"Log", L"Visible", formMain->memoLog->Visible);
    }
    catch(Exception& e)
    {
        MessageDlg(e.Message, mtError, TMsgDlgButtons() << mbOK, 0);
```

```
    }

    delete iniFile;
}
//---------------------------------------------------------------------------
void __fastcall TdmSockets::csOutConnect(TObject *Sender, TCustomWinSocket *Socket)
{
    int intReqId = GetChannelId(Sender);
    intPendingReq++;

    AddToLog(Socket, "CNC[" + IntToStr(intReqId) + "]");
    if(AllActive())
    {
        formMain->buttonSync->Enabled = false;
    }
}
//---------------------------------------------------------------------------
void __fastcall TdmSockets::csOutDisconnect(TObject *Sender, TCustomWinSocket *Socket)
{
    int intReqId = GetChannelId(Sender);
    intPendingReq--;

    AddToLog(Socket, "DSC[" + IntToStr(intReqId) + "]");
    if(AllInactive())
    {
        intPendingReq = 0;

        formMain->buttonSync->Enabled = true;
        Screen->Cursor = crDefault;
    }
}
//---------------------------------------------------------------------------
void __fastcall TdmSockets::csOutError(TObject *Sender, TCustomWinSocket *Socket,
          TErrorEvent ErrorEvent, int &ErrorCode)
{
    int intReqId = GetChannelId(Sender);
    intPendingReq++;

    AddToLog(Socket, "ERR[" + IntToStr(intReqId) + "][" + IntToStr(ErrorCode) + "]");
    if(AllInactive())
    {
        intPendingReq = 0;

        formMain->buttonSync->Enabled = true;
        Screen->Cursor = crDefault;
    }

    ErrorCode = 0;
}
//---------------------------------------------------------------------------
void __fastcall TdmSockets::csOutRead(TObject *Sender, TCustomWinSocket *Socket)
{
    int intReqId = GetChannelId(Sender);

    unsigned long ulTime;

    int intBytesReceived = Socket->ReceiveBuf(&ulTime, 4);
//  if(intBytesReceived != 4)
//  {   // Грешен отговор
//      String str = "Error: " + IntToStr(intBytesReceived) + " bytes received";
//      AddToLog(str);
//      return;
//  }

    ulTime = ntohl(ulTime);

    String strTime;
    strTime.printf(L"%0X", ulTime);
    AddToLog(Socket, "RPL[" + IntToStr(intReqId) + "]::[" + strTime + "]");

    // СИМУЛАЦИЯ НА ОБРАБОТКА
    // многозадачно обслужване за избягване на сериализацията
    // на паралелните клонове
    //
```

```
    std::thread threadWorking(DoWork, Socket, intReqId, intDelayToFree);
    threadWorking.detach();      // развързване на дъщерната нишка от основната
    ////////////////////////////////////////////////////////////////////////////
}
//---------------------------------------------------------------------------
void __fastcall TdmSockets::SendBatchOfReq(void)
{
    /* TODO : Формиране на пакет заявки */
    if(!AllInactive())
    {
        String str = "There are pending requests. Cannot send batch.";
        AddToLog(str);
        return;
    }

    for(int i = 0; i < vcsOut.size(); i++)
    {
        vcsOut[i]->Open();
    }
}
//---------------------------------------------------------------------------
bool __fastcall TdmSockets::AllActive(void)
{
    bool boolResult = true;

    for(int i = 0; i < vcsOut.size(); i++)
    {
        if(!vcsOut[i]->Active)
        {
            boolResult = false;
            break;
        }
    }

    return boolResult;
}
//---------------------------------------------------------------------------
bool __fastcall TdmSockets::AllInactive(void)
{
    bool boolResult = true;

    for(int i = 0; i < vcsOut.size(); i++)
    {
        if(vcsOut[i]->Active)
        {
            boolResult = false;
            break;
        }
    }

    return boolResult;
}
//---------------------------------------------------------------------------
int __fastcall TdmSockets::GetChannelId(TObject *Sender)
{
    int intId = -1;

    for(int i = 0; i < vcsOut.size(); i++)
    {
        if(vcsOut[i] == Sender)
        {
            intId = i;
            break;
        }
    }

    return intId;
}
//---------------------------------------------------------------------------
```